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

Elette Boyle[*] Ran Cohen[†] Aarushi Goel[‡]

October 20, 2023

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).

[*]Reichman University and NTT Research. E-mail: `elette.boyle@runi.ac.il`.
[†]Reichman University. E-mail: `cohenran@runi.ac.il`.
[‡]NTT Research. E-mail: `aarushi.goel@ntt-research.com`.

Contents

1 Introduction

The problem of *Byzantine agreement (BA)* [85, 73] 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 [85, 73] and blockchain protocols (e.g., [31]), to secure multiparty computation (MPC) [96, 57, 7, 30, 88]. 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* [50], 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. [70] 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* [17]), 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 [17, 28, 19], the number of *bits* that must be communicated by each party is large, $\Omega(n)$.[2] BA with *amortized* $\tilde{O}(1)$ per-party communication (and computation) can be achieved [21, 31, 1]; 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 [69, 71].

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 [49] (and furthermore, the connectivity of the underlying communication graph must be $\Omega(n)$ [48, 51]). 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. [61], who showed that without trusted setup assumptions, at least one party must send $\Omega(\sqrt[3]{n})$ messages.[4] But, the bound in [61] applies only to a restricted setting of protocols with *static message filtering*, where

[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., [41, 43].

[2] In fact, the constructions in [17, 28, 19] 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 [61] easily extends to a public setup such as a common reference string.

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 [70] 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, Goldwasser, and Tessaro [17], 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. Boyle et al. [17] 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 [17] 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 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 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., [61, 1]), 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).

[5] A SNARK [81, 10] 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.

- **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 [68], 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.

protocol	rounds	max com. per party	setup	cryptographic assumptions	message filtering	corrupt.	remark
HKK'08 [61]		$\Omega(\sqrt[3]{n})$	crs		static	static	lower bound
KS'09 [68]	$O(1)$	$\tilde{O}(n \cdot \sqrt{n})$	-	-	dynamic	static	
KS'11 [69]	$\mathsf{polylog}(n)$	$\tilde{O}(\sqrt{n})$	-	-	dynamic	adaptive	
KLST'11 [71]	$\mathsf{polylog}(n)$	$\tilde{O}(\sqrt{n})$	-	-	dynamic	static	
BGH'13 [21]	$O(1)$	$\tilde{O}(n)$	-	-	dynamic	static	
BGT'13 [17]	1	$\tilde{O}(n)$	pki	owf	dynamic	static	
CM'19 [31][†]	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 [38][†]	exp $O(1)$	$\tilde{O}(n)$	trusted-pki	vrf	dynamic	adaptive	asynchronous
BKLL'20 [11][†]	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. [(†)] The protocols from [31, 1, 38, 11] 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 $\mathsf{polylog}(n)$ rounds. *Static message filtering* requires honest parties to decide on the parties they will listen to and process their messages before the beginning of each round, whereas *dynamic message filtering* allows this decision to be done during the round and depending on the content of the incoming messages.

[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.

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.

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* [64], *aggregate signatures* [14], or *threshold signatures* [47]. 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] As discussed in the related-work section (Section 1.3), 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., [54, 12]). We provide a more detailed comparison to different signature notions in Section 1.3.

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*.

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 $\mathsf{polylog}(n)$.[10] An SRDS scheme is *t-secure* if it satisfies the above properties even facing t colluding adversarial parties.

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 [17] that

[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 [17].

[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.

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

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. [70]; this establishes a $\mathsf{polylog}(n)$-degree communication tree (which is essentially a sparse overlay network) in which each node is assigned with a committee of $\mathsf{polylog}(n)$ parties. The guarantees are that the $\mathsf{polylog}(n)$-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 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 [17], 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. Every party P_i that receives the signed pair (y, s), evaluates a pseudorandom function (PRF) on the seed s and its identity i to determine a set of (sufficiently random) $\mathsf{polylog}(n)$ parties, and sends the signed (y, s) to every party in that set. A party that receives such a signed pair, can verify that a majority of the parties agree on (y, s) (by the guarantees of multi-signatures) and that it was supposed to receive a message from the sender (by evaluating the PRF on s and the sender's identity). In this case, it can output y and halt.

The protocol from [17] 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 [70] 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 $\mathsf{polylog}(n)$ 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). To address this, we assign $\mathsf{polylog}(n)$ (virtual) identities to every party, one identity for each path from that party to the supreme committee in the communication tree; this ensures that the fraction of signatures that are generated by corrupted parties is equal to the corruption threshold. We refer the reader to Section 4.1 for more details.

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* $\mathsf{polylog}(n)$ *rounds, and where every party communicates* $\mathsf{polylog}(n) \cdot \mathsf{poly}(\kappa)$ *bits.*

[11] The focus of [17] 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 $\mathsf{polylog}(n)$-size set, the so-called *supreme committee*.

We note that our BA protocol is the first to establish a $\mathsf{polylog}(n)$-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 $\tilde{O}(1)$. Thus, we can obtain the following corollaries.

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 $\ell \cdot \mathsf{polylog}(n) \cdot \mathsf{poly}(\kappa)$ bits of communication per party.*
2. **MPC***: Assuming fully homomorphic encryption, a function $f : (\{0,1\}^{\ell_{\mathsf{in}}})^n \to \{0,1\}^{\ell_{\mathsf{out}}}$ can be securely computed with guaranteed output delivery tolerating a static, malicious βn-adversary, such that the total communication complexity (of all parties) is $n \cdot \mathsf{polylog}(n) \cdot \mathsf{poly}(\kappa) \cdot (\ell_{\mathsf{in}} + \ell_{\mathsf{out}})$ 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 $\mathsf{polylog}(n)$-degree communication tree for achieving *full* agreement,[12] 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.

Constructing SRDS. We present two constructions of SRDS, offering a tradeoff between setup assumptions and cryptographic assumptions.

Our first construction is influenced by the "sortition approach" of Algorand [31] 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.[13] Lamport's signatures [72], 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 $\mathsf{polylog}(n)$ 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

[12] If the communication tree is sampled *after* the corruptions have been fixed and is given to the parties as setup, then with overwhelming probability all nodes are *good* (i.e., with more than two-thirds honest majority in the parties assigned). This ensures that the path from all leaf nodes to the root only contains good nodes and hence all parties can communicate with the committee assigned to the root node without disruptions to reach full agreement.

[13] 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.

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) [82] as done in [31]. However, this approach [31] 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, 26, 36, 27, 38, 11, 95, 94] also follow the sortition approach of [31]; however, similar to our first construction, their PKI is assumed to be honestly generated by a trusted third party.

Theorem 1.3 (SRDS from OWF and trusted PKI, informal)**.** *Let $\beta < 1/3$ and assume that OWF exist. Then, there exists a βn-secure SRDS in the trusted-PKI model.*

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. To illustrate the underlying ideas, consider a simplified case where all of the nodes in the almost-everywhere communication tree are honest (each node will essentially be realized by some committee of parties). A naïve construction would be to have all parties sign the message and send the signature to their respective leaf nodes. Every leaf node would then count the number of verified signatures received and send the message and the counter to its parent. In a recursive way, every node would simply add the counters received from its children and send it to its parent. At the end of this process, the root node would get a final count of the total number of verified signatures. This approach completely breaks down, however, if even one node is not honest as it can lie about its count. 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 bad nodes send a valid count of the number of signatures aggregated so far.

Toward 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, every node must prove that they received sufficiently many valid proofs backing up its count. To verify, it is sufficient to check at the root node, whether sufficiently many "base" signatures were aggregated. This approach requires proof systems that support *recursive composition*; for this reason, we use *proof-carrying data* (PCD) systems [32].

A PCD system extends the notion of 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. [9] 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.[14] Extractability assumptions of this kind have been previously considered in, e.g., [93, 45, 58, 20]. 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.

[14]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). We refer the reader to Section 5.2 for the detailed solution.

Theorem 1.4 (SRDS from CRH, SNARKs and bare PKI, informal)**.** *Let $t < n/3$ and assume that CRH and SNARKs with linear extraction exist. Then, there exists a t-secure SRDS in the CRS and bare-PKI model.*

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 (CRS) model—this task is not possible.

We note that the lower bound of Holtby et al. [61] 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 (and then may perform additional sanity checks on the incoming messages from these parties to ensure they are not malformed). The lower bound in [61] shows that *dynamic* filtering, i.e., where in any given round, parties may decide to accept or ignore a message from other parties based on the incoming messages received in that round, is required (at least in the CRS model). We present the first such lower bound in the dynamic-filtering model.

Theorem 1.5 (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 [61]: 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.[15] 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.

At a high level, the idea of our lower bound is to make a linear set of corrupted parties emulate the role of isolated parties during the first part of the protocol (reaching almost-everywhere agreement). This way, the honest parties cannot distinguish between isolated honest parties and faking corrupted parties, and must attempt to communicate the output value to all such parties. However, if each honest party only sends a sublinear number of messages, then with a very high

[15] 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 [19] for a discussion on message filtering in protocols over incomplete graphs.

probability, most isolated honest parties (and faking corrupted parties) only receive messages from a sublinear number of non-isolated parties in the last round. The adversary can use this fact to keep an isolated honest party confused in the following sense. In the last round of an execution with preagreement on 0 (resp., on 1), the adversary sends to this party messages corresponding to an execution with preagreement on 1 (resp., on 0). Without private-coin setup such as PKI, an isolated party cannot distinguish between honest messages in the real execution and fake messages from the simulation.

To carry out this attack, we need to show that there exist parties who receive messages from a "small" set of neighbors in both scenarios: when all parties start with input 0 and when all start with input 1 (otherwise, the adversary may not have a sufficient corruption budget for the attack). Before the protocols begins, the adversary decides on the set of parties to corrupt by first emulating in its head two executions, one with preagreement on 0 and the other with preagreement on 1, where the same linear-size set of parties act as isolated parties. We use a counting argument to show that there exist isolated parties who receive messages from a sublinear set of neighbors in both executions. The adversary targets one of these parties to attack and corrupts all "simulated isolated parties" except for the targeted one, along with the pair of neighbor-sets who communicate with the targeted party in the simulation. We refer the reader to Section 4.2.1 for a formal description and analysis of the attack.

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 1.5) requires some form of private-coin setup. Given this, one of our goals is to minimize the trust assumptions in 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., [24, 66, 67, 25].

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., [76, 1, 26, 36, 27]. 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; this is weaker than a general correlated randomness setup (as used, e.g., in [2] for threshold-signatures setup). 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). In fact, one can consider the model where every party honestly generates and publishes its public key, and corrupted parties can deviate from the protocol only in the online phase.

Necessity of OWF for single-round boost in PKI model. Theorem 1.5 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 pre-image for each public key. In this case, the adversary can carry out the attack for the CRS model, discussed above.

Theorem 1.6 (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.*

We note that this lower bound does not extend to more complex private-coin setups, where the parties receive correlated secret strings that are jointly sampled from some distribution, e.g., setup for information-theoretic signatures. Indeed, given such a setup it is possible to boost almost-everywhere to everywhere agreement in a single round with information-theoretic security and where every party sends $\mathsf{polylog}$ many messages (albeit, each of size $\Omega(n)$) [19]. The reason that the proof approach of Theorem 1.6 does not apply in this case is that when the private keys of two honest parties are correlated, it is unclear how an (even computationally unbounded) adversary that only receives partial information about this correlation can consistently invert the setup information and impersonate honest parties. We leave the feasibility of single-round boost protocols from almost-everywhere to everywhere in the correlated-randomness model, in which every party sends sublinear many *bits* (as opposed to messages), as an interesting open question.

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 assumption [55]). 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, 26, 36, 27]) 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*[16] (see [12, 77] and a discussion in [6]). 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., [77]. 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. We demonstrate challenges toward such an approach, by showing that in some cases this *necessitates* a form of succinct non-interactive arguments.

More specifically, we observe that asserting approval of a multi-signature by sufficiently many parties is inherently equivalent to asserting existence of a large subset of parties $S \subseteq [n]$, such that their corresponding verification keys $\{\mathsf{vk}_i\}_{i\in S}$ satisfy a given function-target relation $f_{\sigma,m}(\{\mathsf{vk}_i\}_{i\in S}) = 1$. (Here m is the message, σ is the multi-signature and $f_{\sigma,m}$ is a function that is derived from the multi-signature verification function.) Such a task can be viewed as a class of "Subset-f" problems on the verification keys $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$, capturing as special cases the standard Subset-Sum and Subset-Product problems with functions $f_\Sigma(\{x_i\}_{i\in S}) = \sum_{i\in S} x_i$ and $f_\Pi(\{x_i\}_{i\in S}) = \prod_{i\in S} x_i$, respectively.

[16]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.

Considering even a generous setup model of trusted PKI, where parties' verification keys $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ are generated independently and honestly, the Subset-f problem begins taking the form of problems where we do not know (or even possibly believe) that the witness $S \subseteq [n]$ can be compressed to $o(n)$ bits. As we show, an SRDS of this form implies a type of average-case non-interactive argument for asserting membership in Subset-f, with *succinct* proof size: namely, a form of succinct non-interactive argument (SNARG) as in [9], with *average-case* soundness guarantees. Although this average-case notion does not directly fall within the negative results of Gentry and Wichs [55], it appears to be a powerful notion, which may be interpreted as a barrier toward this approach to SRDS construction without SNARG-like tools.

Motivated by this, we explore hardness of the Subset-f problem for more general classes of functions f. We show that over rings with appropriate structure (namely, Hadamard product), NP-hardness results for Subset-Sum and Subset-Product can be extended to include (worst-case) Subset-ϕ for all elementary symmetric polynomials ϕ.

We make explicit the above connection for the multi-signature scheme of Lu et al. [77] (LOSSW) in relation to (average-case) Subset-Product, and extend to multi-signature schemes of appropriate structure in relation to the Subset-ϕ problem for elementary symmetric polynomials ϕ. The reduction leverages homomorphism, where a combined signature for a set of parties on message m in the multi-signature scheme corresponds to a valid single-party signature with respect to a specific joint function of the parties' verification keys vk_i; for LOSSW, their product $\mathsf{vk}^* = \prod_i \mathsf{vk}_i$.

Theorem 1.7 (SRDS from multi-signatures requires average-case SNARGs, informal)**.** *Any SRDS based on the LOSSW [77] multi-signature scheme in a natural way (as we define) implies the existence of succinct non-interactive arguments for average-case Subset-Product. This extends to a more general class of multi-signature schemes and Subset-ϕ for elementary symmetric polynomials.*

At a high level, the reduction interprets a (random) Subset-Product instance with target $(x_1, \ldots, x_n, t)$ as a set of uniform *verification keys* $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n, \mathsf{vk}_{n+1} = t^{-1})$ for the multi-signature scheme. Given a satisfying witness $S \subseteq [n]$ with $\prod_{i \in S} x_i = t$ of appropriate size, this translates to knowing a large subset of verification keys for which generating an SRDS on their behalf can be achieved efficiently with respect to the *degenerate* verification key $\mathsf{vk}^* = \prod_{i \in S} \mathsf{vk}_i \cdot t^{-1} = 1$. On the other hand, for uniformly sampled keys *without* such an embedded trapdoor subset, forging such an SRDS will be hard.

We refer the reader to Section 6 for formal definitions of these notions (including SRDS "based on" a multi-signature scheme and average-case SNARGs), as well as further discussion and details of our claims and proofs.

1.3 Additional Related Work

Distributed signatures. Distributed signatures come in many flavors. We compare SRDS to existing notions from the literature.

Threshold signatures [47, 87, 54, 92, 12, 59] can guarantee that a sufficiently large number of parties signed the message, while keeping the signature-length (including all information needed to verify) independent of n. However, threshold signatures require the keys to be generated by a trusted party in a correlated way (e.g., as a Shamir sharing of the signing key), and the signature-reconstruction protocol of existing schemes does not offer succinct aggregation in "small" batches. SRDS imply threshold signatures by having the setup algorithm produce the PKI for the parties, and using the aggregation algorithm to reconstruct a signature.

We note that Libert et al. [75] constructed *fully distributed* threshold signatures that do not require any setup assumptions. However, this scheme is not applicable in our setting, since it requires an interactive key-generation protocol to generate the public and secret keys, and this protocol in turn uses a broadcast channel. In fact, as indicated by our lower bound, some form of private-coin setup is inherently needed for constructing SRDS.

Multi-signatures [64, 83, 12, 6, 77, 15] guarantee that a subset of parties signed the message. Unlike threshold signatures, correlated trusted setup is not needed and a bare PKI suffices; in addition, some of the constructions enable succinct aggregation in "small" batches. *Aggregate signatures* [14, 78, 13, 77, 74, 60] are a similar primitive that allows aggregating signatures on different messages. The main distinction of SRDS is succinctness that enables verification *without* knowing the signing parties. This property is crucial for our BA protocol construction.

Group signatures [29] and *ring signatures* [91] allow any individual party to sign a message on behalf of a set while hiding their identity. This is different than our setting where we need to prove that a majority of the parties signed the message.

Large-scale MPC. The focus of this work is communication complexity of Byzantine agreement protocols; however, Corollary 1.2 demonstrates applications with respect to general secure multiparty computation (MPC). Large-scale MPC was initially studied by Damgård and Ishai [41] and successors (e.g., [42, 43, 44]), in the sense that the amortized per-party work grows only as $\tilde{O}(|C|/n + \mathsf{poly}(n))$, where C is the circuit to be computed. Dani et al. [46] applied the almost-everywhere agreement protocol [70] to achieve MPC with amortized per-party communication of $\tilde{O}(|C|/n + \sqrt{n})$. Using cryptographic assumptions (threshold FHE), Zamani et al. [97] reduced the amortized cost to $\tilde{O}(|C|/n)$. Under comparable assumptions, our results achieve amortized cost of $\tilde{O}(\ell_{\mathsf{in}} + \ell_{\mathsf{out}})$ (where ℓ_{in} and ℓ_{out} stand for the function's input/output length).

The *bottleneck* complexity of MPC was studied in [20], as the maximum communication complexity required by any party within the protocol execution. It was shown that for some n-party functions $f : \{0,1\}^n \to \{0,1\}$, some parties must communicate $\Omega(n)$ bits to compute f, even if security is not required. This result rules out generic MPC with *balanced*, sublinear communication per party, and motivates our MPC results of amortized sublinear communication per party. Note that in [18] load-balanced MPC was achieved, however, amortized over large programs (and in a model that allows each party to have a single use of a broadcast channel).

Communication-efficient BA. Known protocols that break the $\Omega(n^2)$ communication barrier from [49] (for deterministic protocols) follow one of two paradigms. The first is starting with the almost-everywhere agreement protocol of [70] and boost it to full agreement; this approach includes [68, 69, 71, 21, 17], as well as our results. The second is based on the sortition approach from Algorand [31], where only a "small" set of parties are allowed to talk in every round, and includes [31, 1]. The latter approach inherently leads to unbalanced protocols, since parties that are eligible to talk send messages to all other parties.

We note that while [68, 71, 21, 17] and our results hold in the static-corruptions setting, some protocols are resilient to *adaptive* corruptions. Assuming secure data erasures (i.e., where honest parties can erase some parts of their internal states) $\tilde{O}(\sqrt{n})$-balanced BA [69] and $\tilde{O}(1)$-amortized BA [31] can be achieved against adaptive corruptions. In the erasure-free setting, [1] achieved $\tilde{O}(1)$-amortized BA against adaptive corruptions. One of the interesting open questions we pose in Section 1.4 is whether $\tilde{O}(1)$-balanced BA can be achieved in the adaptive setting.

1.4 Open Questions

Our results leave open several interesting questions for followup work.

Our constructions of SRDS offer a trade-off between cryptographic assumptions and setup assumptions (indeed, our lower bound indicates that some form of private-coin setup is needed). Is it possible to get the best of both, i.e., construct SRDS with bare PKI under standard, falsifiable assumptions? This in turn would imply $\tilde{O}(1)$-balanced BA from the corresponding computational assumption and setup. Alternatively, does SRDS in a weak setup model *require* strong computational assumptions: For example, do SRDS with bare PKI *imply* some kind of succinct non-interactive arguments (SNARGs)?

Taking a step back: Is it possible to achieve $\tilde{O}(1)$-balanced BA *unconditionally*? While our SRDS-based approach inherently makes use of computational assumptions (and our lower bound implies this necessity for a *one-shot* boost from almost-everywhere to everywhere agreement in the PKI model), this leaves open the possibility of removing cryptography via an alternative approach.

Can one further extend the lower bound in this work, identifying a minimal required *round complexity* for generically converting from almost-everywhere to everywhere agreement within various setup models?

In the $\tilde{O}(1)$-amortized BA setting, known constructions consider stronger security models. Namely, the protocol in Braud-Santoni et al. [21] is secure against static corruptions (similarly to our protocols); however, no trusted setup assumptions are required. The protocol of Abraham et al. [1] guarantees security against adaptive corruptions; however, it requires a trusted PKI assumption. In contrast, the protocol of King and Saia [69] does not require setup assumptions and is resilient to adaptive corruptions, but it provides suboptimal total communication $\tilde{O}(n\sqrt{n})$. It is interesting to explore if $\tilde{O}(1)$-balanced BA can be achieved without setup or in the adaptive setting.

Regarding the communication model, the vast majority of sub-quadratic BA protocol are defined in the synchronous model. In the asynchronous setting unbalanced sub-quadratic BA in the trusted PKI model was recently proposed [38, 11]. We note that *balanced* sub-quadratic BA is not known even in the partially synchronous model. An interesting question is to expand our techniques beyond the synchronous realm.

Finally, all known BA protocols with $o(n^2)$ total communication follow either the approach of King et al. [70] or of Chen and Micali [31], which are based on electing a polylog-size committee. As such, these protocols only support a non-optimal constant fraction of corruptions. Is it possible to achieve $o(n^2)$ total communication while tolerating the optimal number of corruptions $t < n/2$?

Paper Organization

In Section 2, we provide basic definitions. SRDS are defined in Section 3. Our BA protocol and the lower bounds appear in Section 4. Section 5 presents two constructions of SRDS, and in Section 6, we explore the connection of SRDS based on multi-signatures to succinct non-interactive arguments. Some of the definitions and proofs are deferred to the appendix.

2 Preliminaries

In this section, we present the security model and the definition of Byzantine agreement. Additional definitions of proof-carrying data systems, of Merkle hash proof systems, and of multi-signatures, can be found in Appendix A.

Protocols. All protocols considered in this paper are PPT (probabilistic polynomial time): the running time of every party is polynomial in the (common) security parameter, given as a unary string. For simplicity, we consider Boolean-input Boolean-output protocols, where apart from the common security parameter, each party has a single input bit, and each of the honest parties outputs a single bit. We note that our protocols can be used for agreement on longer strings, with an additional dependency of the communication complexity on the input-string length.

We consider protocols in the PKI model, and we distinguish between two flavors of PKI: a *trusted PKI* and a *bare PKI*. In both settings, a trusted party samples a secret key sk_i and a public key vk_i, for every $i \in [n]$, from some distribution. The adversary is allowed to corrupt parties adaptively based on $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ and learn the secret key associated with every corrupted party. In the bare-PKI model, the adversary can replace the public key of every corrupted party by an arbitrary string vk'_i of its choice, even "after" looking at the public keys of honest parties.

The communication model is *synchronous*, meaning that protocols proceed in rounds. In each round, every party can send a message to every other party over a private channel.[17] It is guaranteed that every message sent in a round will arrive at its destinations by the end of that round. The adversary is *rushing* in the sense that it can use the messages received by corrupted parties from honest parties in a given round to determine the corrupted parties' messages for that round.

Note that an adversary can always blow up the communication complexity of a protocol by flooding honest parties with many bogus messages. It is therefore standard to count only messages that are actually processed by honest parties. We follow the model of [19], who formalized this intuition. Namely, message receival consists of two phases: a *filtering* phase, where incoming messages are inspected according to specific filtering rules defined by the protocol, and some messages may be discarded, followed by a *processing* phase, where the party computes its next-message function based on the remaining non-filtered messages. In practice, the filtering procedure should be "lightweight," and consist of operations like counting messages or verifying validity of a signature.

Byzantine Agreement. Informally, in an n-party, t-resilient Byzantine agreement protocol, the honest parties must agree on one of their input bits, even when t parties collude and actively try to prevent it. We provide two definitions for BA: the first is the standard, property-based definition and the second is based on the real/ideal paradigm.

We start with the property-based definition. This definition captures the core properties required for consensus; namely, *agreement*, *validity*, and *termination*. This is a weaker definition than the simulation-based one, and as such is most suitable for proving lower bounds.

Definition 2.1 (BA, property-based)**.** *Let π be an n-party protocol in which every party P_i has an input bit $x_i \in \{0,1\}$ and outputs a bit $y_i \in \{0,1\}$ at the end of the protocol. The protocol π is an n-party, t-resilient BA protocol (according to the property-based definition) if the following properties are satisfied with all but negligible probability when up to t parties maliciously attack the protocol:*

- ***Agreement.*** *For every pair of honest parties P_i and P_j it holds that $y_i = y_j$.*
- ***Validity.*** *If there exists a bit x such that for every honest party P_i it holds that $x_i = x$, then for every honest party P_i it holds that $y_i = x$.*

[17] We note that using standard techniques, our constructions can be defined over *authenticated* channels (that do not provide privacy) by additionally assuming the existence of key-agreement protocols.

- ***Termination.*** *Every honest party eventually outputs a bit.*

We proceed with the simulation-based definition, which requires the protocol to realize an ideal BA functionality. Roughly speaking, an ideal functionality represents a trusted third party that receives inputs from all the parties and provides them with the correct output. A protocol in the real world (where parties communicate between themselves and no trusted party exists) securely realizes an ideal functionality if every attack in the real protocol can be simulated in the ideal-world computation; since by definition no attack can happen in the ideal world, it is concluded that no attack can happen in the real world as well. We refer the reader to [22, 23, 56] for further details on the real/ideal paradigm. We follow the standard ideal functionality for Byzantine agreement, see, e.g., [84, 34, 39, 35, 37], where the functionality collects the inputs from all parties and counts them; in case of a strong majority of the inputs equals some bit, then this bit is set as the output, and otherwise the adversary gets to choose the output. This definition implies the property-based one and is stronger as it guarantees security under composition; we use this definition for our protocol constructions.

The functionality f_{ba}

The functionality f_{ba} proceeds as follows, running with parties $P_1, \ldots, P_n$ and an adversary $\mathcal{S}$, statically corrupting a subset of parties indexed by a set $\mathcal{I} \subseteq [n]$ of size $|\mathcal{I}| \leq t$.

1. For each $i \in [n] \setminus \mathcal{I}$, party P_i sends a bit $x_i \in \{0, 1\}$ as its input; the functionality sends $(P_i, x_i)_{i \in [n] \setminus \mathcal{I}}$ to the adversary.
2. The adversary sends input bits $(x_i)_{i \in \mathcal{I}}$ for the corrupted parties and a "tie-breaker bit" $\tilde{x}$.
3. Once all parties provided their inputs, if there exists a bit b such that $|\{i \mid x_i = b\}| \geq n - t$, then set $y = b$; otherwise, set $y = \tilde{x}$. Send y to every party.

Figure 1: The Byzantine agreement functionality

Definition 2.2 (BA, simulation-based)**.** *An n-party, t-resilient Byzantine agreement protocol (according to the simulation-based definition) is a protocol π that realizes the BA ideal functionality (defined in Figure 1) tolerating a malicious adversary statically corrupting up to t parties.*

Balanced BA. In this work, we design a balanced Byzantine agreement protocol, with $\tilde{O}(1)$ per-party communication; i.e., for all adversarial strategies, the communication complexity incurred by each honest party is $\mathsf{polylog}(n) \cdot \mathsf{poly}(\kappa)$ with overwhelming probability. One can also consider a slightly relaxed variant, where the per-party communication is $\tilde{O}(1)$ *in expectation*. The relaxed notion is satisfied by committee-based BA protocols such as [31, 1], where each party has a similar probability of being elected in the committee and hence the parties incur a similar per-party communication when given sufficiently many invocations of the protocol. In this work, however, we focus on the former stronger notion that is formalized as follows.

Definition 2.3 (Balanced BA)**.** *Let $\alpha(n, \kappa)$ be a function. An n-party, t-resilient Byzantine agreement protocol has $\alpha(n, \kappa)$-balanced communication complexity, if for every PPT adversary corrupting up to t parties, the communication complexity incurred by each honest party is $\alpha(n, \kappa)$, except for negligible probability.*

3 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.

Below, we define the new signature scheme and the security requirements. Later, in Section 5, we present two constructions offering a tradeoff between cryptographic and setup assumptions: the first assuming one-way functions in the trusted-PKI model and the second assuming CRH and SNARKs with linear extraction in the CRS and bare-PKI model. In Section 6, we show that a natural approach towards constructing SRDS in an untrusted-PKI model has strong connections to succinct average-case argument systems for certain NP-Complete problems.

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

Definition 3.1 (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* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Aggregate}, \mathsf{Verify})$ *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 $m \in \mathcal{M}$, 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.[18] Given a base signature/aggregated signature, let $\mathsf{max}(\sigma)$ denote the function that extracts the maxima associated with σ and $\mathsf{min}(\sigma)$ denote the function that extracts the maxima associated with σ (in case of a base signature, both $\mathsf{max}(\sigma)$ and $\mathsf{min}(\sigma)$ will return the same value).

[18]In Section 5 we will show how this property can be achieved by each of our constructions.

Remark (Notation n). Here, we use n to denote the number of parties in the SRDS scheme. Looking ahead, the effective number of (virtual) parties in the SRDS used in our BA protocol in Section 4 will be larger than the actual (real) 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 4) and our SRDS constructions (Section 5) 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 $\mathsf{Aggregate}$. 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 (possibly randomized) step $\mathsf{Aggregate}_2$ is short.

Looking ahead at the BA protocol in Section 4.1, 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 3.2 (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 $\mathsf{Aggregate}$ algorithm can be decomposed into 2 algorithms $\mathsf{Aggregate}_1$ and $\mathsf{Aggregate}_2$, such that the following hold:*
   - $\mathsf{Aggregate}_1(\mathsf{pp}, \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}, m, \{\sigma_1, \ldots, \sigma_q\}) \to S_{\mathsf{sig}}$*, where S_{sig} is of size $\mathsf{poly}(\log n, \kappa)$ and $\mathsf{Aggregate}_1$ is deterministic.*
   - $\mathsf{Aggregate}_2(\mathsf{pp}, m, S_{\mathsf{sig}}) \to \sigma$*, 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 2 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 [70]. For our application in the BA protocol in Section 4.1, we require this tree to be an "$(n, \mathcal{I})$-almost-everywhere-communication tree" (see Definition 3.3), where n is the number of parties and $\mathcal{I}$ is the set of corrupt parties.[19] 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 4.1. 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 $(n, \mathcal{I})$-almost-everywhere-communication tree, which is a slight variant of the tree described in King et al. [70].

Definition 3.3 ($(n, \mathcal{I})$-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 $(n, \mathcal{I})$-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.*

[19]This tree is a combinatorial object that was first defined by King et al. [70]. 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 4.1.

Definition 3.4 (robustness)**.** *Let $t < n/3$. An SRDS scheme Π is t-robust with a bare PKI (resp., with a trusted PKI) if for* $\mathsf{mode} = \mathsf{b\text{-}pki}$ *(resp.,* $\mathsf{mode} = \mathsf{tr\text{-}pki}$*) and for any (stateful) PPT adversary $\mathcal{A}$ it holds that:*

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

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

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.

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 $|\mathcal{I}| \leq t$ and $\mathcal{A}$ requests to corrupt a party P_i:

    (a) Send sk_i to $\mathcal{A}$ and receive back vk'_i.

    (b) If $\mathsf{mode} = \mathsf{b\text{-}pki}$, 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 $(n, \mathcal{I})$-$\mathsf{almost\text{-}everywhere\text{-}communication\ tree}$ $T = (V, E)$ (as per Definition 3.3), in which 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. 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 2: Robustness experiment for SRDS

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

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

The experiment $\mathsf{Expt}^{\mathsf{forge}}_{\mathsf{mode},\Pi,\mathcal{A}}$ *is defined in Figure 3.*

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 2.

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 3: 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 4, 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 3.6 (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).*

4 Balanced Communication-Efficient Byzantine Agreement

In this section, we consider Byzantine agreement protocols with $\tilde{O}(1)$ communication per party. In Section 4.1, 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 Section 4.2, we show that a similar task cannot be achieved under weaker setup assumptions.

4.1 Balanced Byzantine Agreement from SRDS

We start by showing how to combine succinctly reconstructed distributed signatures (SRDS) with the protocol of [17] to obtain BA with balanced $\tilde{O}(1)$ communication. We prove the following theorem.

Theorem 4.1 (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 (according to Definition 2.2) in a hybrid model for generating the SRDS setup and the relevant PKI, such that:*

- *The round complexity and communication locality are $\mathsf{polylog}(n)$; every party communicates $\mathsf{polylog}(n) \cdot \mathsf{poly}(\kappa)$ 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 4.1 with our SRDS constructions from Section 5, we get the following corollaries.

Corollary 4.2. *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 4.3. *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.*

High-level overview. The protocol is defined in a hybrid model that abstracts the communication tree of [70]. The parties can communicate in a way that mimics almost-everywhere agreement, and the adversary is allowed to isolate a $o(1)$ fraction of the parties. 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 he is assigned to, it is essential to ensure 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 his virtual identities. These virtual IDs are assigned to the parties in such a way that the virtual IDs associated with 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. Looking ahead, this property is necessary for robustness, when using our SRDS construction from Section 5.2.

The protocol starts by invoking $f_{\mathsf{ae\text{-}comm}}$ (defined below) to obtain an almost-everywhere-communication tree where each party is assigned to z leaves. The supreme committee members (parties assigned to the root-node) run Byzantine agreement on their inputs to agree on the output y and run a coin-tossing protocol to agree on a random seed s. The supreme committee then makes use of the communication-tree to distribute these values to all non-isolated parties. The parties then collectively generate an SRDS signature to certify the pair (y, s).

To compute this signature, each party locally signs the received pair of values; this is done using a different virtual identity for every leaf node corresponding to the party. Each signature is sent to all parties assigned to the corresponding leaf node. For each node in the tree, the assigned parties aggregate the received signatures, while making sure that the maxima and minima of virtual IDs associated with the signatures that they aggregate indeed lie within the range associated with the leaf nodes that have a path to the current node, and propagate them to the node's parent in a recursive way until reaching the root, where the final aggregated signature is computed.

Next, the supreme-committee again uses the communication-tree to distribute this aggregated signature to all non-isolated parties. Each non-isolated party evaluates a PRF on the seed s and its identity to determine a set of parties, to which he sends the pair (y, s) along with the signature. Isolated parties can now verify the signature and be convinced about the correct output y. Here correctness crucially relies on the fact that the adversary could not have forged an aggregated SRDS signature on any other value.

In Section 4.1.1, we define the ideal functionalities to be used in the BA protocol, and in Section 4.1.2, we describe the protocol and prove its security. Finally, in Section 4.1.3, we present applications of our protocol to broadcast and MPC.

4.1.1 Functionalities used in the Protocol

We start by describing the functionalities used in our construction.

Almost-everywhere communication. The functionality $f_{\mathsf{ae\text{-}comm}}$ is a reactive functionality that abstracts the properties obtained by the protocol from [70]. 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 $\mathcal{D} \subseteq [n]$. 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 $(n, \mathcal{I})$-almost-everywhere-communication tree defined in Section 3. Specifically, in Definition 3.3, 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 4.4 ($(n, \mathcal{I})$-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 $(n, \mathcal{I})$-almost-everywhere-communication tree (Definition 3.3) 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.*

The original protocol of [70] has an inverse-polynomial error in n; the reason is that the committees are chosen to be $O(\log n)$. Boyle et al. [17] adjusted the protocol to have committees of poly-logarithmic size, thus obtaining poly-logarithmic locality with a negligible error in n. Note that the security parameter κ is not used in this protocol, so the locality is independent of κ.

As observed in [17], 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. [70] securely realizes $f_{\mathsf{ae\text{-}comm}}$ in the authenticated-channels model tolerating a computationally unbounded, malicious adversary statically corrupting βn parties, for a constant $\beta < 1/3$. Every invocation requires $\mathsf{polylog}(n)$ rounds, and every party sends and processes $\mathsf{polylog}(n)$ bits. Throughout all invocations, every party sends to, and processes messages received from $\mathsf{polylog}(n)$ other parties.

The functionality $f_{\mathsf{ae\text{-}comm}}$

The n-party reactive functionality $f_{\mathsf{ae\text{-}comm}}$ proceeds as follows:

- **First invocation:** Upon receiving an init message from each party, the functionality asks the adversary for a communication tree $T = (V, E)$ and does the following:
  1. Verify that T is an n-party almost-everywhere-communication tree with respect to the set of corrupted parties $\mathcal{I}$ (otherwise, output $\perp$ to all parties).
  2. Let $\mathcal{D}$ be the set of isolated parties in T and let $\mathcal{C}$ be the set of parties assigned to the root.
  3. The functionality sends to each P_i for $i \in [n]$ its local view in the tree, consisting of:
     - All the nodes that P_i is assigned to (and the parties assigned to them).
     - All the parent and children nodes (and the parties assigned to them) of the nodes that P_i is assigned to.
- **Subsequent invocations:** Every party P_i with $i \in \mathcal{C}$ provides a message m_i. If more than 2/3 of the parties in $\mathcal{C}$ provided the same message m, send m to the adversary and receive back $\{\hat{m}_j\}_{j \in \mathcal{D}}$. For every $i \notin \mathcal{D}$ deliver m to P_i and for every $j \in \mathcal{D}$ deliver $\hat{m}_j$ to P_j.

Figure 4: The almost-everywhere communication functionality

Byzantine agreement. We consider the standard Byzantine agreement functionality f_{ba} as defined in Section 2 (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 [52] realizes f_{ba} over authenticated channels tolerating a computationally unbounded, malicious adversary statically corrupting $t < n/3$ parties using $t + 1$ rounds and $\mathsf{poly}(n)$ communication complexity. An immediate corollary is that for $n' = \mathsf{polylog}(n)$, the n'-party BA functionality f_{ba} can be instantiated using $\mathsf{polylog}(n)$ rounds and $\mathsf{polylog}(n)$ 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. [33] 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 [52], $n' = \mathsf{polylog}(n)$ parties can agree on a random κ-bit string in $\mathsf{polylog}(n)$ rounds and $\mathsf{polylog}(n) \cdot \mathsf{poly}(\kappa)$ communication.

Signature aggregation. The signature-aggregation functionality $f_{\mathsf{aggr\text{-}sig}}$ (formally described in Figure 5) 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.

Note that the inputs to the aggregation procedure are not private, so if the aggregation algorithm $\mathsf{Aggregate}_2$ is deterministic (for example, in the OWF-based SRDS construction in Section 5.1) the parties simply need to agree on the common set of input signatures S_{sig} and locally run $\mathsf{Aggregate}_2$ to obtain the same aggregated signature. To agree on S_{sig}, each party broadcasts its input signatures and filters-out invalid signatures by running the deterministic algorithm $\mathsf{Aggregate}_1$. However, if the algorithm $\mathsf{Aggregate}_2$ is randomized, it may be the case that security relies on keeping the random coins hidden from the parties. For this reason, after the parties agree on S_{sig}, we use an MPC protocol to compute the aggregated signature and realize $f_{\mathsf{aggr\text{-}sig}}$.

The functionality $f_{\mathsf{aggr\text{-}sig}}(\mathcal{P})$

The n'-party functionality $f_{\mathsf{aggr\text{-}sig}}$, running with parties $\mathcal{P} = \{P_1, \ldots, P_{n'}\}$ and the adversary, is parametrized by the public parameters pp and proceeds as follows.

1. Every party P_i sends $(m_i, S_{\mathsf{sig}_i})$ as input, where S_{sig_i} is a set of signatures.
2. If at least 2/3 of the parties provided the same message m and the same set S_{sig}, then compute

$$\sigma \leftarrow \mathsf{Aggregate}_2(\mathsf{pp}, m, S_{\mathsf{sig}}).$$

   Else, let the adversary choose σ.
3. Finally, deliver σ to every party P_i.

Figure 5: The signature-aggregation functionality

Assuming the existence of one-way functions, the protocol of Damgård and Ishai [40] can be used to realize the n'-party functionality $f_{\mathsf{aggr\text{-}sig}}$, for $n' = \mathsf{polylog}(n)$, 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 $\mathsf{polylog}(n) \cdot \mathsf{poly}(\kappa)$ communication. In our construction, this functionality is used by the parties assigned to a node (in the almost-everywhere communication-tree obtained from $f_{\mathsf{ae\text{-}comm}}$) for aggregating signatures received from parties assigned to their children. From Definition 4.4, we know that each node only has $\log(n)$ child nodes and each node is assigned $\mathsf{polylog}(n)$ parties. Therefore, $f_{\mathsf{aggr\text{-}sig}}$ is only used for aggregating at most $\mathsf{polylog}(n)$ signatures. Note that in [40] a broadcast channel is also required and the resulting protocol is constant round. For $n' = \mathsf{polylog}(n)$ the broadcast can be realized by a deterministic protocol, e.g., from [52], and the resulting protocol has $\mathsf{polylog}(n)$ rounds and $\mathsf{polylog}(n) \cdot \mathsf{poly}(\kappa)$ communication.

4.1.2 The Byzantine Agreement Protocol

Having defined the ideal functionalities, we are now ready to present our BA protocol in Figure 6. To reduce the security of π_{ba} to that of the SRDS scheme, we will show that by *robustness* every honest party will receive an accepting signature on (y, s), and by unforgeability, no party will receive an accepting signature on a different value. Before proceeding to the proof, we discuss a subtlety in the reduction.

Recall that robustness of an SRDS scheme ensures that an adversary who after the *setup and corruption* phase is allowed to choose a message m, and the order of aggregation (using a directed rooted tree T), cannot prevent the honest parties from successfully signing m. Note that if in π_{ba}, an adversary can prevent the honest parties from signing (y, s), then we can derive the corresponding tree and partially aggregated signatures of the corrupted parties to break the robustness of the SRDS scheme. Note that here, in the robustness game, we will assume that the total number of parties are $n \cdot z$ (i.e., each virtual party in the Byzantine agreement protocol is a real party in the SRDS game) and $(n, \mathcal{I})$-almost-everywhere-communication tree with repeated parties T used in the Byzantine agreement protocol is transformed into an $(n \cdot z, \{(i,j)\}_{i \in \mathcal{I}, j \in [z]})$-almost-everywhere-communication tree by augmenting it with a level 0 comprising of $n \cdot z$ nodes (representing the $n \cdot z$ parties in the SRDS game), and adding an edge between each of these nodes and the leaf node that it (i.e., the party that they represent) is assigned to.

Lemma 4.5. *Let $\beta < 1/3$ and assume the existence of PRF and βn-secure SRDS in the bare-PKI model (resp., trusted-PKI model). Then, protocol π_{ba} is a βn-resilient BA protocol in the $(f_{\mathsf{ae\text{-}comm}}, f_{\mathsf{ba}}, f_{\mathsf{ct}}, f_{\mathsf{aggr\text{-}sig}})$-hybrid model such that:*

- *The round complexity and the locality of the protocol are $\mathsf{polylog}(n)$; the number of bits communicated by each party is $\mathsf{polylog}(n) \cdot \mathsf{poly}(\kappa)$.*
- *The adversary can adaptively corrupt the parties based on the public setup of the SRDS, i.e., pp and $\{\mathsf{vk}_{1,1}, \ldots, \mathsf{vk}_{n,z}\}$ before the onset of the protocol. For bare PKI, the adversary can additionally replace the corrupted parties' public keys.*

The proof of Lemma 4.5 can be found in Appendix B.1.

4.1.3 Applications

We point out a few applications of our BA protocol.

Broadcast with balanced polylog communication. Consider a single a run of the protocol (on dummy inputs). The communication graph forms a tree with stronger properties than Definition 4.4, achieving *everywhere* agreement of all parties on the supreme committee, such that every party sends only $\tilde{O}(1)$ throughout the protocol constructing it. Having established the communication tree, it is possible to run a simple broadcast protocol in the PKI model. The sender signs his input bit and sends it up to the supreme committee, which in turn sends the signed bit to all other parties. If fact, since the communication tree is reusable, after multiple executions (with different senders) the communication will grow in a proportional way only to the number of bits that have been broadcasted. In particular, note that the SRDS PKI is only needed for a single run of the protocol (to establish the communication tree) and is not needed afterwards.

Protocol π_{ba}

- **Common Input:** An SRDS scheme and a PRF family $\mathcal{F} = \{F_s\}_{s\in\{0,1\}^\kappa}$ mapping elements of $[n]$ to subsets of $[n]$ of size $\mathsf{polylog}(n)$.
- **Private Input:** Every party 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 $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa, 1^{n\cdot z})$. Every party P_i locally computes $(\mathsf{vk}_{i,j}, \mathsf{sk}_{i,j}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ for every $j \in [z]$. The public output consists of pp and the set of public keys $\mathsf{vk} = \{\mathsf{vk}_{i,j}\}_{i\in[n],j\in[z]}$. We assume that there exists a mapping $\mathsf{idmap} : [n] \times [z] \to [n \cdot z]$ 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 $(f_{\mathsf{ae\text{-}comm}}, f_{\mathsf{ba}}, f_{\mathsf{ct}}, f_{\mathsf{aggr\text{-}sig}})$-hybrid model.
- **The Protocol:**

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

[a] To ensure that the corrupt parties do not broadcast very long messages, we assume that the parties broadcast each element in $S_{\mathsf{sig}}^{i,\ell,1}$ one-by-one and each party is only allowed to initiate polylogarithmic number of broadcasts.

Figure 6: Byzantine agreement with balanced $\mathsf{polylog}$ communication

Corollary 4.6. *Let $\beta < 1/3$ be a constant. Assuming βn-secure SRDS schemes, there exists an n-party binary broadcast protocol tolerating a malicious adversary that can statically corrupt βn of the parties, such that the communication locality of ℓ executions is $\mathsf{polylog}(n)$, and the round complexity and the number of bits each party communicates is $\ell \cdot \mathsf{polylog}(n) \cdot \mathsf{poly}(\kappa)$.*

MPC with amortized polylog communication overhead. Following the MPC protocol from [17], the supreme committee can run among themselves a protocol establishing an encryption key of a public-key encryption scheme where the decryption key is secret shared among the committee members, and broadcast the public key. Every party encrypts its input and sends it up the tree to the supreme committee that run an MPC protocol for decrypting all ciphertexts and compute the function. Using FHE-based MPC that minimize the communication (e.g., [4]), we obtain the following corollary.

Corollary 4.7. *Let $\beta < 1/3$ be a constant. Assuming βn-secure SRDS and FHE schemes, every n-party functionality $f : (\{0,1\}^{\ell_{\mathsf{in}}})^n \to \{0,1\}^{\ell_{\mathsf{out}}}$ can be securely computed tolerating a malicious adversary that can statically corrupt βn parties, such that communication locality and round complexity are $\mathsf{polylog}(n)$, and amortized communication complexity is $(\ell_{\mathsf{in}} + \ell_{\mathsf{out}}) \cdot \mathsf{polylog}(n) \cdot \mathsf{poly}(\kappa)$.*

4.2 Lower Bound on Balanced Byzantine Agreement

In the previous section, we showed how to extend almost-everywhere agreement to full agreement in one round. The minimal setup assumptions used were a bare PKI and CRS. In Section 4.2.1, we show the some form of private-coin setup is necessary for this task.[20] In Section 4.2.2, we show that in the PKI model, where the public/private keys of each party are independently sampled, cryptographic assumptions are further needed.

4.2.1 Lower Bound on Balanced Byzantine Agreement in CRS Model

We denote by $f^*_{\mathsf{ae\text{-}comm}}$ a weakened version of the functionality $f_{\mathsf{ae\text{-}comm}}$ (from Figure 4) that enables communication between almost all of the parties, except for an isolated set $\mathcal{D}$ that is randomly chosen by the functionality, rather than by the adversary. We note that this notion is non-standard and is not achieved by existing protocols for almost-everywhere agreement. The purpose of this adjustment is to provide a stronger lower bound, as the adversary's capabilities are more restricted. In fact, we only require that with some inverse-polynomial probability, there exists a single isolated party that is chosen by the functionality.

Theorem 4.8 (Theorem 1.5, restated)**.** *Let π be a βn-resilient Byzantine agreement protocol in the $(f_{\mathsf{crs}}, f^*_{\mathsf{ae\text{-}comm}})$-hybrid model, for $\beta < 1$. Assume that π has two parts: the first consists of a polynomial number of rounds where communication is via $f^*_{\mathsf{ae\text{-}comm}}$, and the second consists of a single round over point-to-point channels. Then, there exists a party that sends $\Theta(n)$ messages in the last round.*

Proof. By classical results [85, 51], BA protocols cannot tolerate one-third of corrupted parties, even in the CRS model; therefore, we can assume that $\beta < 1/3$. Let π be a protocol in the $(f_{\mathsf{crs}}, f^*_{\mathsf{ae\text{-}comm}})$-hybrid model that invokes $f^*_{\mathsf{ae\text{-}comm}}$ for polynomially many rounds followed by a

[20]We note that, our lower bound easily extends to the random oracle model, for the sake of simplicity we prove it merely with a CRS setup.

single point-to-point round, and assume that the number of messages sent by every party in the last round is $o(n)$. We will construct an adversarial strategy that violates the *validity* of π with noticeable probability.

Choosing the corrupted set. Given the common reference string crs, the adversary starts by deciding on the set of corrupted parties. The adversary chooses a random subset $\mathcal{J} \subseteq [n]$ of size $\beta n/2$ and simulates two executions of π inside its head.

- In the first execution, all parties have input bit 0 where every party P_j with $j \in \mathcal{J}$ is corrupted and does not send any message throughout the protocol. For every $j \in \mathcal{J}$, denote the set of parties that sends messages to P_j in the last point-to-point round by $\mathcal{C}_j^0$ and record the messages as $\{\hat{m}_{i\to j}^0\}_{i\in\mathcal{C}_j^0}$.
- In the second execution, all parties have input bit 1 where every party P_j with $j \in \mathcal{J}$ is corrupted and does not send any message throughout the protocol. For every $j \in \mathcal{J}$, denote the set of parties that sends messages to P_j in the last point-to-point round by $\mathcal{C}_j^1$ and record the messages as $\{\hat{m}_{i\to j}^1\}_{i\in\mathcal{C}_j^1}$.

In each of the virtual executions described above, from the joint view of all parties P_i with $i \notin \mathcal{J}$, every party P_j with $j \in \mathcal{J}$ could be an isolated honest party, so they must join forces and send messages to every such P_j. Note that it could be that some parties in $\mathcal{J}$ receive a linear number of messages, e.g., if every party P_i with $i \notin \mathcal{J}$ sends a message to the same party P_j for some $j \in \mathcal{J}$. However, as each party sends only $o(n)$ messages in this step, the number of such parties cannot be too large; in particular, there must be a party who receives $o(n)$ messages in *both* of the above executions.

Claim 4.9. *There exists $j \in \mathcal{J}$ such that $|\mathcal{C}_j^0 \cup \mathcal{C}_j^1| \in o(n)$.*

Proof. Consider the first virtual execution, where all honest parties start with input 0. Denote by $\mathcal{J}' = \{j \in \mathcal{J} \mid |\mathcal{C}_j^0| \in \Theta(n)\}$ the set of parties that receive a linear number of messages from $\{P_i\}_{i\notin\mathcal{J}}$ (i.e., receive $\delta(n)$ messages for some $\delta \in \Theta(n)$). If $|\mathcal{J}'| \in \Theta(n)$, i.e., there are linear many parties that receive a linear number of messages, it must be that the number of messages sent from $\{P_i\}_{i\notin\mathcal{J}}$ to $\{P_j\}_{j\in\mathcal{J}}$ is quadratic. This will contradict to the assumption that every party in $\{P_i\}_{i\notin\mathcal{J}}$ only sends a sublinear number of messages. Therefore, $|\mathcal{J}'| \in o(n)$, and it holds that $|\mathcal{C}_j^0| \in o(n)$ for a majority of $j \in \mathcal{J}$. By an analogue argument, also in the second virtual execution, where all honest parties start with input 1, it holds that $|\mathcal{C}_j^1| \in o(n)$ for a majority of $j \in \mathcal{J}$. Hence, there exists $j \in \mathcal{J}$ for which $|\mathcal{C}_j^0 \cup \mathcal{C}_j^1| \in o(n)$. □

The adversary proceeds by choosing uniformly at random $i^* \in \mathcal{J}$. If it holds that $|\mathcal{C}_{i^*}^0 \cup \mathcal{C}_{i^*}^1| \geq \beta n/2$, the adversary aborts the attack and halts. By Claim 4.9, the adversary does not abort with probability at least $1/n$. Next, the adversary chooses a random subset $\mathcal{I} \subseteq [n] \setminus \{i^*\}$ of size βn, such that $\mathcal{J} \cup \mathcal{C}_{i^*}^0 \cup \mathcal{C}_{i^*}^1 \setminus \{i^*\} \subseteq \mathcal{I}$. Denote by $\mathcal{E}$ the event where the adversary does not abort and that party P_{i^*} is isolated by $f_{\mathsf{ae\text{-}comm}}^*$ with respect to the set of corrupted parties $\mathcal{I}$ as defined above. By the definition of $f_{\mathsf{ae\text{-}comm}}^*$ and by Claim 4.9, this event happens with inverse-polynomial probability. The attack defined below will be analyzed conditioned on the event $\mathcal{E}$.

The attack. We proceed by defining a series of hybrid experiments to contradict the *validity* of π. For the first claim, we define the adversarial strategy $\mathcal{A}_1$, where the corrupted parties are P_i with $i \in \mathcal{J}$. The parties in $\mathcal{J} \setminus \{i^*\}$ do not send messages throughout the protocol, whereas party P_{i^*} does not send any message during the first part of the protocol, but in the last round sends messages as an honest party with input 0 that was isolated in the first part.

Claim 4.10. *Consider an execution of π with $\mathcal{A}_1$, where all parties start with input bit* 1*. Then, all honest parties output* 1 *with all but negligible probability.*

Proof. The claim follows immediately by the *validity* property of π. □

For the second claim, we define the adversarial strategy $\mathcal{A}_2$, where the set of corrupted parties is $\mathcal{I}$. The parties in $\mathcal{J} \setminus \{i^*\}$ do not send messages throughout the protocol, and the parties in $\mathcal{I} \setminus \mathcal{J}$ play honestly on input 1, except that in the last round, the set of parties in $\mathcal{C}^0_{i^*}$ additionally sends the messages $\{\hat{m}^0_{i \to i^*}\}_{i \in \mathcal{C}^0_{i^*}}$ to P_{i^*}.

Claim 4.11. *Consider an execution of π with $\mathcal{A}_2$, where party P_{i^*} starts with input bit* 0 *and all other parties with input bit* 1*. Then, conditioned on $\mathcal{E}$, all honest parties (including P_{i^*}) output* 1 *with all but negligible probability.*

Proof. Conditioned on $\mathcal{E}$, the view of all honest parties other than P_{i^*}, is identically distributed as in Claim 4.10. It follows that every honest party but P_{i^*} will output 1 except for negligible probability. By *agreement*, P_{i^*} will also output 1 except for negligible probability. □

Next, consider the adversarial strategy $\mathcal{A}_3$, where the set of corrupted parties is $\mathcal{I}$. The parties in $\mathcal{J} \setminus \{i^*\}$ do not send messages throughout the protocol, and the parties in $\mathcal{I} \setminus \mathcal{J}$ play honestly on input 0, except that in the last round, the set of parties in $\mathcal{C}^1_{i^*}$ additionally sends the messages $\{\hat{m}^1_{i \to i^*}\}_{i \in \mathcal{C}^1_{i^*}}$ to P_{i^*}.

Claim 4.12. *Consider an execution of π with $\mathcal{A}_3$ where all parties starts with input bit* 0*. Then, conditioned on the event $\mathcal{E}$, all honest parties output* 1 *with noticeable probability.*

Proof. We will show that, conditioned on $\mathcal{E}$, the view of P_{i^*} in this scenario will be distributed as in previous scenario with noticeable probability; hence, by Claim 4.11, party P_{i^*} will output 1 with the same probability. By *agreement* so will all other honest parties.

To analyze the view of P_{i^*} in the first scenario (where all parties outside of $\mathcal{J}$ start with input 1), let $\mathcal{B}_1$ be the set of honest parties that send messages to P_{i^*} in the last round. Denote by $\{m^1_i\}_{i \in \mathcal{B}_1}$ the messages sent by these parties to P_{i^*}. The view of P_{i^*} consists of his input bit 0, his random coins, the crs, the messages $\{\hat{m}^0_{i \to i^*}\}_{i \in \mathcal{C}^0_{i^*}}$, and messages $\{m^1_i\}_{i \in \mathcal{B}_1}$.

To analyze the view of P_{i^*} in the second scenario (where all parties outside of $\mathcal{J}$ start with input 0), let $\mathcal{B}_0$ be the set of honest parties that send messages to P_{i^*} in the last round. Denote by $\{m^0_i\}_{i \in \mathcal{B}_0}$ the messages sent by these parties to P_{i^*}. The view of P_{i^*} consists of his input bit 0, his random coins, the crs, the messages $\{\hat{m}^1_{i \to i^*}\}_{i \in \mathcal{C}^1_{i^*}}$, and messages $\{m^0_i\}_{i \in \mathcal{B}_0}$.

Recall that by Claim 4.9, when running two *independent* executions of π in which the parties in $\mathcal{J}$ do not talk till the last round, the first where every P_j with $j \in [n] \setminus \mathcal{J}$ starts with 0 and the second when every such P_j starts with 1, there exists $j^* \in \mathcal{J}$ such that P_{j^*} receives $o(n)$ messages in both executions with probability at least $1/n$. Since the executions in the first and second scenarios are independent of each other and also of the two virtual executions run in the head of the adversary, it holds that there exists a party P_{j^*} with $j^* \in \mathcal{J}$ that receives $o(n)$ messages in each of the four executions with probability at least $1/n^2$. Since i^* is chosen uniformly at random in $\mathcal{J}$, it holds that the sizes of $\mathcal{C}^0_{i^*}$, $\mathcal{C}^1_{i^*}$, $\mathcal{B}_0$, and $\mathcal{B}_1$ are all is $o(n)$ with probability at least $1/n^3$. In this case, it holds that the pair of sets $\{\hat{m}^1_{i\to i^*}\}_{i\in\mathcal{C}^1_{i^*}}$ and $\{m^0_i\}_{i\in\mathcal{B}_0}$ is identically distributed as $\{\hat{m}^0_{i\to i^*}\}_{i\in\mathcal{C}^0_{i^*}}$ and $\{m^1_i\}_{i\in\mathcal{B}_1}$, and the view of P_{i^*} is identically distributed in both the first and second scenarios. $\square$

Since by assumption, the event $\mathcal{E}$ occurs with inverse-polynomial probability, the attack succeeds with inverse-polynomial probability. This concludes the proof of Theorem 4.8. $\square$

4.2.2 Lower Bound on Balanced Byzantine Agreement in PKI Model

We proceed to prove the second lower bound, showing that in the trusted PKI model, where each party receives an independently sampled pair of public/private keys, one-way functions are necessary for extending almost-everywhere agreement to full agreement in a single communication round. Note that a lower bound in the trusted PKI model readily implies a lower bound in weaker PKI models.

Theorem 4.13 (Theorem 1.6, restated)**.** *Let π be a βn-resilient Byzantine agreement protocol in the trusted PKI and $f^*_{\mathsf{ae\text{-}comm}}$-hybrid model, for $\beta < 1$. Assume that π has two parts: the first consists of a polynomial number of rounds where communication is via $f^*_{\mathsf{ae\text{-}comm}}$, and the second consists of a single round over point-to-point channels. Then, if one-way functions do not exist, there exists a party that sends $\Theta(n)$ messages in the last round.*

At a high level, the proof of the theorem considers an adversary that receives the public keys $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ of the PKI setup, where each vk_i is sampled with a secret sk_i *independently* of other keys. Under the assumption that one-way functions do not exist, with noticeable probability the adversary can find a corresponding secret key $\widetilde{\mathsf{sk}}_i$ (i.e., a pre-image) for every vk_i, and then carry out the attack from Section 4.2.1. This intuition, however, is not sufficient for proving the theorem, since the distribution of randomly generated keys $\{(\mathsf{vk}_i, \mathsf{sk}_i)\}_{i\in[n]}$ may be different than the distribution of the inverted keys $\{(\mathsf{vk}_i, \widetilde{\mathsf{sk}}_i)\}_{i\in[n]}$. In this case, the simulated messages generated by the adversary when emulating the executions in its head may be different than those generated in the real protocol, and so honest parties can tell them apart.

To overcome this subtlety, recall that Impagliazzo and Luby [62] showed that the existence of *distributional one-way functions* (functions for which it is hard to sample a uniform pre-image) implies the existence of one-way functions. Stated differently, if one-way functions do not exist, then for any polynomial $p(\cdot)$ and any polynomial-time computable function f, there exists a PPT algorithm Inv such that, for infinitely many n, the following distributions are $1/p(n)$-statistically close:

- $\{(x, f(x)) \mid x \leftarrow \{0,1\}^n\}$.
- $\{(\mathsf{Inv}(f(y)), y) \mid x \leftarrow \{0,1\}^n, y = f(x)\}$.

In this case, we say that Inv inverts f with $1/p(n)$-statistical closeness. In case the distributions are identically distributed we call the inverter *perfect* and denote it by PInv.

Proof of Theorem 4.13. Without loss of generality, in the following we consider $n = \kappa$. The trusted PKI setup can be modeled by a trusted party that for every $i \in [n]$ samples uniformly random $r_i \in \{0,1\}^n$, computes a polynomial-time function $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n) = f_{\mathsf{pki}}(r_1, \ldots, r_n)$, where for every $i \in [n]$, $\mathsf{vk}_i = f^i_{\mathsf{pki}}(r_i)$ for some function f^i_{pki}. The trusted party outputs to each party P_i the random coins r_i along with $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$. Denote by $1/p(n)$ the success probability of the attack in the proof of Theorem 4.8 and let Inv be the inverter algorithm for f_{pki} that is guaranteed to exist by [62] with $1/2p(n)$-statistical closeness under the assumption that one-way functions do not exist.

Let π be a protocol in the trusted PKI and $f^*_{\mathsf{ae\text{-}comm}}$-hybrid model that invokes $f^*_{\mathsf{ae\text{-}comm}}$ for polynomially many rounds followed by a single point-to-point round, and assume that the number of messages sent by every party in the last round is $o(n)$. Following the lines of the proof of Theorem 4.8, we will construct an adversarial strategy that violates the *validity* of π with non-negligible probability.

Choosing the corrupted set. Initially, the adversary receives the public keys $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ from the trusted party modeling the trusted PKI, and computes $(\tilde{r}_1, \ldots, \tilde{r}_n) \leftarrow \mathsf{Inv}(\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$. Next, the adversary chooses a random subset $\mathcal{J} \subseteq [n]$ of size $\beta n/2$ and simulates two executions of π inside its head.

- In the first execution, every party P_i has input bit 0 and receives $\tilde{r}_i$ and $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ from the trusted PKI. Every party P_j with $j \in \mathcal{J}$ is corrupted and does not send any message throughout the protocol. For every $j \in \mathcal{J}$, denote the set of parties that sends messages to P_j in the last point-to-point round by $\mathcal{C}^0_j$ and record the messages as $\{\hat{m}^0_{i \to j}\}_{i \in \mathcal{C}^0_j}$.
- In the second execution, every party P_i has input bit 1 and receives $\tilde{r}_i$ and $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ from the trusted PKI. Every party P_j with $j \in \mathcal{J}$ is corrupted and does not send any message throughout the protocol. For every $j \in \mathcal{J}$, denote the set of parties that sends messages to P_j in the last point-to-point round by $\mathcal{C}^1_j$ and record the messages as $\{\hat{m}^1_{i \to j}\}_{i \in \mathcal{C}^1_j}$.

Claim 4.14. *There exists $j \in \mathcal{J}$ such that $|\mathcal{C}^0_j \cup \mathcal{C}^1_j| \in o(n)$, except for probability $1/2p(n)$.*

Proof. Consider a perfect inverter PInv for f_{pki}. In that case for every $j \in \mathcal{J}$, the simulated messages by the adversary $\{\hat{m}^0_{i \to j}\}_{i \in \mathcal{C}^0_j}$ (resp., $\{\hat{m}^1_{i \to j}\}_{i \in \mathcal{C}^1_j}$) are identically distributed as the messages that P_j receives in the last round in an honest execution where all parties in $[n] \setminus \mathcal{J}$ have input 0 (resp., 1) and parties in $\mathcal{J} \setminus \{j\}$ are corrupted and do not send messages. Therefore, by an identical argument to Claim 4.9, there exists $j \in \mathcal{J}$ such that $|\mathcal{C}^0_j \cup \mathcal{C}^1_j| \in o(n)$.

The claim follows since Inv is an inverter with $1/2p(n)$-statistical closeness. $\square$

The adversary proceeds by choosing uniformly at random $i^* \in \mathcal{J}$, and as before, if $|\mathcal{C}^0_{i^*} \cup \mathcal{C}^1_{i^*}| \geq \beta n/2$, the adversary aborts the attack and halts. By Claim 4.14 the adversary does not abort with probability at least $1/n - 1/2p(n)$ (recall that by the proof of Theorem 4.8, $1/p(n) \leq 1/n$; hence, $1/n - 1/2p(n) > 0$). Next, the adversary chooses a random subset $\mathcal{I} \subseteq [n] \setminus \{i^*\}$ of size βn, such that $\mathcal{J} \cup \mathcal{C}^0_{i^*} \cup \mathcal{C}^1_{i^*} \setminus \{i^*\} \subseteq \mathcal{I}$. Denote by $\mathcal{E}$ the event where the adversary does not abort and that party P_{i^*} is isolated by $f^*_{\mathsf{ae\text{-}comm}}$ with respect to the set of corrupted parties $\mathcal{I}$ as defined above. By the definition of $f^*_{\mathsf{ae\text{-}comm}}$ and by Claim 4.14, this event happens with inverse-polynomial probability.

The rest of the proof proceeds exactly as in the proof of Theorem 4.8, with the only difference that the statistical distance of the PKI private keys in the protocol and those simulated by the adversary is bounded by $1/2p(n)$. Since the attack in the proof of Theorem 4.8 succeeds with probability $1/p(n)$, it holds that $1/p(n) - 1/2p(n)$ is noticeable. $\square$

5 Constructions of SRDS

In Section 5.1, we present an SRDS scheme with trusted PKI based on OWF, and in Section 5.2, an SRDS scheme with bare PKI based on proof-carrying data and CRH.

5.1 SRDS from One-Way Functions

Theorem 5.1 (Theorem 1.3, restated)**.** *Let $\beta < 1/3$ be a constant. Assuming the existence of one-way functions, there exists a βn-secure SRDS scheme in the trusted PKI model.*

The main building block in our construction is an augmented version of digital signatures with the ability to obliviously sample a verification key without knowing the signing key. Note that by assuming secure erasures, or a trusted party that does not reveal the key-generation coins, our construction can be based on any digital signatures scheme.

Definition 5.2 (signatures with oblivious key generation)**.** *A digital signature scheme* ($\mathsf{DS.KeyGen}$, $\mathsf{DS.Sign}$, $\mathsf{DS.Verify}$) *has* oblivious key generation *if there exists an algorithm* $\mathsf{DS.OKeyGen}$ *that on input the security parameter* 1^κ *outputs a key* ovk*, such that the following hold:*

- ***Indistinguishability.*** *The distribution of* vk*, where* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{DS.KeyGen}(1^\kappa)$*, should be computationally indistinguishable from* ovk*, where* $\mathsf{ovk} \leftarrow \mathsf{DS.OKeyGen}(1^\kappa)$*.*
- ***Obliviousness.*** *A PPT adversary $\mathcal{A}$ can win the following game with negligible probability:*
  1. *Challenger computes* $\mathsf{ovk} = \mathsf{DS.OKeyGen}(1^\kappa; r)$ *and sends* (ovk, r) *to* $\mathcal{A}$*.*
  2. *$\mathcal{A}$ responds with a pair (m, σ), and wins if* $\mathsf{DS.Verify}(\mathsf{ovk}, m, \sigma) = 1$*.*

Claim 5.3. *Assuming the existence of one-way functions, there exists a one-time digital signature scheme with oblivious key generation.*

Proof Sketch. Recall the one-time signatures of Lamport [72] for ℓ-bit messages. Given a one-way function f, the signing key consists of 2ℓ random κ-bits strings $x_1^0, x_1^1, \ldots, x_\ell^0, x_\ell^1$ and the verification key is $y_1^0, y_1^1, \ldots, y_\ell^0, y_\ell^1$, where $y_i^b = f(x_i^b)$. A signature on a message $m = (m_1, \ldots, m_\ell)$ is $\sigma = (x_1^{m_1}, \ldots, x_\ell^{m_\ell})$. To verify a signature $\sigma = (\sigma_1, \ldots, \sigma_\ell)$, check for each $i \in [\ell]$ if $f(\sigma_i) = f(x_i^{m_i})$.

By instantiating the one-way function with a length-doubling pseudorandom generator G, we can define the oblivious key-generation algorithm by sampling 2ℓ random 2κ-bit strings. Indistinguishability follows from the pseudorandomness of G, and obliviousness from its one-wayness. $\square$

Note that via standard transformations (e.g., [56, Sec. 6.4]) the one-time signature construction above can be extended to multi-message signatures with oblivious key generation.

Overview of the construction. Our construction makes use of a digital signature scheme with oblivious key generation (Definition 5.2). For each party toss a biased coin that outputs heads with probability ℓ/n, for some $\ell = \omega(\log(n))$. If the output is heads, sample standard signature keys $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{DS.KeyGen}(1^\kappa)$; otherwise, obliviously sample $\mathsf{vk}_i \leftarrow \mathsf{DS.OKeyGen}(1^\kappa)$. A signature on a message m can be computed only by parties with a valid signing key. The aggregation algorithm concatenates these valid signatures.[21] Verification of a signature requires counting how many valid signatures were signed on the message. Since each signature in this construction encodes the index associated with the corresponding verification key and each aggregate/partially aggregate signature is essentially a concatenation of the base signatures, it is easy to see that in this construction, given a signature/aggregate signature, the maxima and minima associated with it can be easily determined.

The construction of the SRDS scheme is formally described in Figure 7 and the proof of Theorem 5.1 can be found in Appendix C.1.

SRDS from OWF

Let $(\mathsf{DS.KeyGen}, \mathsf{DS.OKeyGen}, \mathsf{DS.Sign}, \mathsf{DS.Verify})$ be a signature scheme with oblivious key generation, let $\alpha(n, \kappa) \in \mathsf{poly}(\log n, \kappa)$, and let $\ell = \log^c(n)$ for some constant $c > 1$.

- $\mathsf{Setup}(1^\kappa, 1^n)$: Output $\mathsf{pp} = 1^\kappa$.
- $\mathsf{KeyGen}(\mathsf{pp})$: Toss a biased coin that outputs heads with probability ℓ/n. If the outcome is heads, compute $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{DS.KeyGen}(1^\kappa)$; else compute $\mathsf{vk} \leftarrow \mathsf{DS.OKeyGen}(1^\kappa)$ and set $\mathsf{sk} = \perp$. Output $(\mathsf{vk}, \mathsf{sk})$.
- $\mathsf{Sign}(\mathsf{pp}, i, \mathsf{sk}, m)$: If $\mathsf{sk} \neq \perp$, compute $\mathsf{sig} \leftarrow \mathsf{DS.Sign}(\mathsf{sk}, m)$ and set $\sigma = \{(i, m, \mathsf{sig})\}$; otherwise, set $\sigma = \perp$. Output σ.
- $\mathsf{Aggregate}_1(\mathsf{pp}, \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}, m, \{\sigma_1, \ldots, \sigma_q\})$: Initialize $S_{\mathsf{sig}} = \emptyset$. For every $i \in [q]$:
  - Parse σ_i as a set of signature-tuples $\{(i_j, m_{i_j}, \mathsf{sig}_{i_j})\}_{i_j \in S_i}$ (for some set S_i).
  - For every $(i_j, m_{i_j}, \mathsf{sig}_{i_j}) \in \sigma_i$, check whether $m = m_{i_j}$ and $\mathsf{DS.Verify}(\mathsf{vk}_{i_j}, m, \mathsf{sig}_{i_j}) = 1$ and $\nexists (i_j, \cdot, \cdot, \cdot) \in S_{\mathsf{sig}}$. If so, set $S_{\mathsf{sig}} = S_{\mathsf{sig}} \cup \{(i_j, m, \mathsf{sig}_{i_j})\}$.

  Denote by $\|\sigma\|$ the bit length of σ. If $\sum_{\sigma \in S_{\mathsf{sig}}} \|\sigma\| \leq \alpha(n, \kappa)$, output S_{sig}; else, output $\perp$.
- $\mathsf{Aggregate}_2(\mathsf{pp}, m, S_{\mathsf{sig}})$: If $\sum_{\sigma \in S_{\mathsf{sig}}} \|\sigma\| \leq \alpha(n, \kappa)$, output S_{sig}; else, output $\perp$.
- $\mathsf{Verify}(\mathsf{pp}, \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}, m, \sigma)$: Initialize $S_{\mathsf{ver}} = \emptyset$.
  - Parse σ as a set of signature-tuples $\{(i, m_i, \mathsf{sig}_i)\}_{i \in S}$ (for some set S).
  - For each $(i, m_i, \mathsf{sig}_i) \in \sigma$, if $m = m_i$ and $\mathsf{DS.Verify}(\mathsf{vk}_i, m_i, \mathsf{sig}_i) = 1$ and $\nexists (i, \cdot, \cdot) \in S_{\mathsf{ver}}$, set $S_{\mathsf{ver}} = S_{\mathsf{ver}} \cup \{(i, m_i, \mathsf{sig}_i)\}$.
  - If $\sum_{\sigma \in S_{\mathsf{ver}}} \|\sigma\| \leq \alpha(n, \kappa)$ and $|S_{\mathsf{ver}}| > \ell'/3$, where $\ell' = \ell/2$, output 1; else, output 0.

Figure 7: Succinctly reconstructed distributed signatures from one-way functions

[21] Since this aggregation process is deterministic, decomposing the algorithm is redundant – we represent it by two algorithms for completeness, to make the syntax compatible with the BA protocol in Section 4.1.

5.2 SRDS from CRH and SNARKs

The construction in Section 5.1 was in the trusted PKI model. In this section, we show how to construct SRDS in the bare PKI, albeit under stronger cryptographic assumptions. Namely, we consider CRH and SNARKs with $\mathsf{linear\ extraction}$, where the size of the extractor is linear in the size of the prover (i.e., $|\mathbb{E}_{\mathcal{P}^*}| \leq c \cdot |\mathcal{P}^*|$ for some constant c). An extractability assumption of this kind has been considered in [93, 45, 58, 20].

Theorem 5.4 (Theorem 1.4, restated)**.** *Let $t < n/3$. Assuming the existence of CRH, digital signatures, and SNARKs with linear extraction, there exists a t-secure SRDS scheme in the CRS model with a bare PKI.*

The construction of the SRDS scheme is formally described in Figure 8 and the proof of Theorem 5.4 can be found in Appendix C.2.

Overview of the construction. As discussed in the Introduction, a PCD system allows for propagation of information up the tree in a succinct and publicly verifiable way. Having the parties locally sign the message and keep track of the number of verified signatures aggregated so far via the PCD system, seems to capture most of our requirements for SRDS. However, in order to prevent an adversary from aggregating fake signatures or multiple copies of the same signature, we need to devise a mechanism of verifying the base signatures in the compliance predicate.

One approach is to hard-wire all verification keys into the compliance predicate and verify each base-level signature. However, this will blow-up the size of the predicate to $O(n)$ and, as a result, the PCD-prover algorithm will run in time $O(n)$. In this case, the scheme will no longer be succinct, as the algorithm $\mathsf{Aggregate}_2$ internally runs the PCD prover. Indeed, recall that in the BA protocol (in Section 4.1.2) $\mathsf{Aggregate}_2$ is executed via an MPC protocol; hence, its complexity must be $\tilde{O}(1)$.

To get around this barrier, we use a *Merkle tree* to hash all the verification keys; a Merkle tree enables a long string (here, the list of *all* verification keys $(\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$) to be hashed to a short value in a committing way, such that one can prove inclusion of the key vk_i in the input string by providing an "opening" to vk_i in low complexity (here, logarithmic in n) (see Appendix A.3 for details). Each incoming and outgoing PCD transcript will now contain this hash value H_{vk}. The base-level transcript will also consist of:

1. The signature γ_i and corresponding verification key $k_i = \mathsf{vk}_i$.
2. A Merkle proof p_i certifying that k_i is the i^{th} verification key in the computation of H_{vk}.

The compliance predicate, in this case, will verify:

1. The signature γ_i with respect to the k_i.
2. That k_i is properly hashed in the Merkle tree.

To prevent an adversary from using a different H_{vk} value, we add an additional check in the compliance predicate that the value of H_{vk} is consistent in all the incoming and outgoing transcripts. Finally, to prevent an adversary from potentially aggregating multiple copies of the same base signature, we encode a maxima max and a minima min of the indices of the keys used to sign the base signatures in each transcript of the PCD proof.

We proceed to give a more detailed overview of our construction. Each base signature (and aggregate signature) corresponds to a "truncated" PCD transcript and a corresponding proof. For base signatures, this proof is set to $\perp$ and in the remaining aggregated signatures, this proof corresponds to a PCD proof. Each truncated transcript $z' = (m, c, \mathsf{max}, \mathsf{min}, \gamma)$ consists of a message m over which the signature is computed, a counter c to keep a count of the number of distinct keys used to sign this signature, a maxima max and minima min of the indices of the keys that signed the message, and a value γ that in the base case is a signature on m corresponding to $\mathsf{vk}_{\mathsf{max}}$ and in all other cases is set to $\perp$.

Each party starts by locally signing the message using its signing key sk_i and preparing z'_i. The algorithm $\mathsf{Aggregate}_1$ collects base signatures and/or partially aggregated signatures, checks for their validity and prepares their corresponding PCD transcripts. For base signatures (where $z' = (m, 1, i, i, \gamma)$ and $\pi = \perp$), $\mathsf{Aggregate}_1$ checks that γ and m verify with respect to vk_i; if so, it prepares a Merkle proof p for vk_i and the PCD transcript is set to $z = z||(H_{\mathsf{vk}}, \mathsf{vk}_i, p)$. For partially aggregated signatures (where $z' = (m, c, \mathsf{max}, \mathsf{min}, \perp)$ and $\pi \neq \perp$), it completes the transcript by setting $z = z'||(H_{\mathsf{vk}}, \perp, \perp)$ and runs the PCD verification algorithm on (z, π). The algorithm $\mathsf{Aggregate}_2$ computes the outgoing transcript that is compliant with the valid incoming PCD transcripts and computes a PCD proof certifying this, i.e., that it is based on c *distinct and valid* individual signatures. Finally, to verify (z', π), set $z = z'||(H_{\mathsf{vk}}, \perp, \perp)$, verify the PCD (z, π), and count the total number of keys used for signing this signature.

Since each signature/aggregate signature in this construction essentially consists of a transcript and a proof and the transcript encodes information about maxima and minima associated with the signature, it is easy to see that in this construction, given a signature/aggregate signature, the maxima and minima associated with it can be easily determined.

6 Connection with Succinct Arguments

In Section 5, we showed how to construct SRDS with a strong setup assumption (trusted PKI) from OWF, and with relatively weak setup assumptions (bare PKI) at the expense of strong, non-falsifiable, cryptographic assumptions (SNARKs with linear extractors). A natural approach towards constructing SRDS that balances the cryptographic and setup assumptions, is to augment a multi-signature scheme with some method of convincing the verifier that sufficiently many parties contributed to the signing process. Indeed, multi-signatures are known to exist under standard falsifiable assumptions in the *registered PKI* model [77]. In this model each party locally generates its own keys (as with bare PKI) but to publish its verification key, the party must prove knowledge of the corresponding secret key, see [12, 77] and a discussion in [6].

In this section, we discuss challenges toward such an approach, by showing that in some cases this *necessitates* some form of succinct non-interactive arguments. We begin in Section 6.1 by formalizing the notion of SNARGs for average-case instances of a language, and formalizing the notion of SRDS "based on" multi-signatures. Next, in Section 6.2, we show that any SRDS based on LOSSW multi-signatures imply SNARGs for average-case instances of the Subset-Product problem. Finally, in Section 6.3, we explore hardness of various Subset-f problems and their connection to SRDS based on more general multi-signature schemes.

SRDS from CRH and SNARKs

Let $(\mathsf{DS.KeyGen}, \mathsf{DS.Sign}, \mathsf{DS.Verify})$ be a digital signature scheme, let $(\mathsf{PCD.Gen}, \mathsf{PCD.Prover}, \mathsf{PCD.Verify})$ be a publicly verifiable proof-carrying data (PCD) system for *logarithmic*-depth *polynomial*-size compliance predicates $\mathtt{C}$, and let $(\mathsf{Merkle.Setup}, \mathsf{Merkle.Hash}, \mathsf{Merkle.Proof}, \mathsf{Merkle.Verify})$ be the Merkle hash proof system corresponding to a hash function H. Let $\alpha(n, \kappa) \in \mathsf{poly}(\log n, \kappa)$.

- $\mathsf{Setup}(1^\kappa)$: Sample $\mathsf{seed} \leftarrow \mathsf{Merkle.Setup}(1^\kappa)$ and PCD keys corresponding to a compliance predicate $\mathtt{C}$ (defined below), as $(\sigma_{\mathsf{pcd}}, \tau_{\mathsf{pcd}}) \leftarrow \mathsf{PCD.Gen}(1^\kappa, \mathtt{C})$.

  The predicate $\mathtt{C}$: Given an input vector $\vec{z}_{\mathsf{in}}$ of length ℓ, such that for $j \in [\ell]$ the j^{th} entry of $\vec{z}_{\mathsf{in}}$ is of the form $\vec{z}_{\mathsf{in}}[j] = (m_{\mathsf{in},j}, c_{\mathsf{in},j}, \mathsf{max}_{\mathsf{in},j}, \mathsf{min}_{\mathsf{in},j}, \gamma_{\mathsf{in},j}, H_{\mathsf{vk},j}, k_{\mathsf{in},j}, p_{\mathsf{in},j})$, and output data of the form $z_{\mathsf{out}} = (m_{\mathsf{out}}, c_{\mathsf{out}}, \mathsf{max}_{\mathsf{out}}, \mathsf{min}_{\mathsf{out}}, \gamma_{\mathsf{out}}, H_{\mathsf{vk,out}}, k_{\mathsf{out}}, p_{\mathsf{out}})$, the predicate $\mathtt{C}(\vec{z}_{\mathsf{in}}, z_{\mathsf{out}})$ equals 1 iff:

  1. For every $j \in [\ell]$, it holds that $H_{\mathsf{vk},j} = H_{\mathsf{vk,out}}$.
  2. For every $j \in [\ell]$, if it is a base level (i.e., if $\mathsf{max}_{\mathsf{in},j} = \mathsf{min}_{\mathsf{in},j}$ and $\gamma_{\mathsf{in},j} \neq \perp$), then $\mathsf{DS.Verify}(k_{\mathsf{in},j}, m_{\mathsf{in},j}, \gamma_{\mathsf{in},j}) = 1$ and $\mathsf{Merkle.Verify}(\mathsf{seed}, (\mathsf{max}_{\mathsf{in},j} || k_{\mathsf{in},j}), H_{\mathsf{vk,out}}, p_{\mathsf{in},j}) = 1$.
  3. It holds that $\mathsf{min}_{\mathsf{in},\ell} \leq \mathsf{max}_{\mathsf{in},\ell}$ and for every $j \in [\ell - 1]$ that $\mathsf{min}_{\mathsf{in},j} \leq \mathsf{max}_{\mathsf{in},j} < \mathsf{min}_{\mathsf{in},j+1}$, i.e., max of an input is greater than or equal to its min and less than the min of the next input.
  4. min of the output transcript is equal to the min of the first input, i.e., $\mathsf{min}_{\mathsf{out}} = \mathsf{min}_{\mathsf{in},1}$.
  5. max of the output transcript is equal to the max of the last input, i.e., $\mathsf{max}_{\mathsf{out}} = \mathsf{max}_{\mathsf{in},\ell}$.
  6. c_{out} stores a count of the number of signatures aggregated so far, i.e., $c_{\mathsf{out}} = \sum_{j \in [\ell]} c_{\mathsf{in},j}$.

  The output is $\mathsf{pp} = (1^\kappa, \sigma_{\mathsf{pcd}}, \tau_{\mathsf{pcd}}, \mathsf{seed})$.

- $\mathsf{KeyGen}(\mathsf{pp})$: Parse $\mathsf{pp} = (1^\kappa, \sigma_{\mathsf{pcd}}, \tau_{\mathsf{pcd}}, \mathsf{seed})$, compute $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{DS.KeyGen}(1^\kappa)$, output $(\mathsf{vk}, \mathsf{sk})$.

- $\mathsf{Sign}(\mathsf{pp}, i, \mathsf{sk}_i, m_i)$: Compute $\gamma_i \leftarrow \mathsf{DS.Sign}(\mathsf{sk}_i, m_i)$, set $z' = (m_i, 1, i, i, \gamma_i)$, and output $\sigma = (z', \perp)$.

- $\mathsf{Aggregate}_1(\mathsf{pp}, \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}, m, \{\sigma_1, \ldots, \sigma_q\})$: Parse $\mathsf{pp} = (1^\kappa, \sigma_{\mathsf{pcd}}, \tau_{\mathsf{pcd}}, \mathsf{seed})$. Compute $H_{\mathsf{vk}} = \mathsf{Merkle.Hash}(\mathsf{seed}, (1||\mathsf{vk}_1), \ldots, (n||\mathsf{vk}_n))$ and set $S_{\mathsf{sig}} = \{H_{\mathsf{vk}}\}$. For each $i \in [q]$ do the following:

  – Parse $\sigma_i = (z'_i, \pi_i)$ and $z'_i = (m_i, c_i, \mathsf{max}_i, \mathsf{min}_i, \gamma_i)$

  – For base level (where $\mathsf{max}_i = \mathsf{min}_i$, $m = m_i$, $\pi_i = \perp$, $\gamma_i \neq \perp$ and $\mathsf{DS.Verify}(\mathsf{vk}_{\mathsf{max}_i}, m_i, \gamma_i) = 1$), compute $p_i = \mathsf{Merkle.Proof}(\mathsf{seed}, (1||\mathsf{vk}_1), \ldots, (n||\mathsf{vk}_n), (\mathsf{max}_i||\mathsf{vk}_{\mathsf{max}_i}))$, prepare the transcript $z_i = z'_i||(H_{\mathsf{vk}}, \mathsf{vk}_{\mathsf{max}_i}, p_i)$ and set $S_{\mathsf{sig}} = S_{\mathsf{sig}} \cup \{(z_i, \pi_i)\}$.

  – Else, set $z_i = z'_i||(H_{\mathsf{vk}}, \perp, \perp)$ and check whether $\mathsf{PCD.Verify}(\tau_{\mathsf{pcd}}, z_i, \pi_i) = 1$ and $m = m_i$. If so, set $S_{\mathsf{sig}} = S_{\mathsf{sig}} \cup \{(z_i, \pi_i)\}$.

  If $\|S_{\mathsf{sig}}\| \leq \alpha(n, \kappa)$,[a] output S_{sig} ; else, output $\perp$.

- $\mathsf{Aggregate}_2(\mathsf{pp}, m, S_{\mathsf{sig}})$: Parse $\mathsf{pp} = (1^\kappa, \sigma_{\mathsf{pcd}}, \tau_{\mathsf{pcd}}, \mathsf{seed})$ and set $c_{\mathsf{out}} = 0$. Parse $S_{\mathsf{sig}} = \{H_{\mathsf{vk}}, \ldots\}$. For each $\sigma_i \in S_{\mathsf{sig}} \setminus \{H_{\mathsf{vk}}\}$, parse $\sigma_i = (z'_i, \pi_i)$ and $z'_i = (m_i, c_i, \mathsf{max}_i, \mathsf{min}_i, \gamma_i)$ and set $c_{\mathsf{out}} = c_{\mathsf{out}} + c_i$. Let $(z_{\mathsf{in},1}, \pi_{\mathsf{in},1})$ be the first element in S_{sig} where $z_{\mathsf{in},1} = (\cdot, \cdot, \cdot, \mathsf{min}_{\mathsf{in},1}, \cdot)$. Set $\mathsf{min}_{\mathsf{out}} = \mathsf{min}_{\mathsf{in},1}$. Similarly, denote $u = |S_{\mathsf{sig}}|$ and let $(z_{\mathsf{in},u}, \pi_{\mathsf{in},u})$ be the last element in S_{sig} where $z_{\mathsf{in},u} = (\cdot, \cdot, \mathsf{max}_{\mathsf{in},u}, \cdot, \cdot)$. Set $\mathsf{max}_{\mathsf{out}} = \mathsf{max}_{\mathsf{in},u}$, set $z'_{\mathsf{out}} = (m, c_{\mathsf{out}}, \mathsf{max}_{\mathsf{out}}, \mathsf{min}_{\mathsf{out}}, \perp)$, and set $z_{\mathsf{out}} = (z'_{\mathsf{out}}, H_{\mathsf{vk}}, \perp, \perp)$. Compute $\pi_{\mathsf{out}} \leftarrow \mathsf{PCD.Prover}(\sigma_{\mathsf{pcd}}, S_{\mathsf{sig}}, \mathsf{linp} = \perp, z_{\mathsf{out}})$ and output $\sigma = (z'_{\mathsf{out}}, \pi_{\mathsf{out}})$.

- $\mathsf{Verify}(\mathsf{pp}, \{\mathsf{vk}_i\}_{i \in [n]}, m, \sigma)$: Parse $\mathsf{pp} = (1^\kappa, \sigma_{\mathsf{pcd}}, \tau_{\mathsf{pcd}}, \mathsf{seed})$, $\sigma = (z', \pi)$, $z' = (m', c, \mathsf{max}, \mathsf{min}, \gamma_i)$. Compute $H_{\mathsf{vk}} = \mathsf{Merkle.Hash}((1||\mathsf{vk}_1), \ldots, (n||\mathsf{vk}_n); \mathsf{seed})$ and $z = z'||(H_{\mathsf{vk}}, \perp, \perp)$. If $m' = m$, $\mathsf{PCD.Verify}(\tau_{\mathsf{pcd}}, z, \pi) = 1$, $c \geq n/3$, and $\|\sigma\| \leq \alpha(n, \kappa)$, output 1; else, output 0.

[a] $\|S_{\mathsf{sig}}\|$ stands for the bit length of S_{sig}.

Figure 8: Succinctly reconstructed distributed signatures from CRH and SNARKs

6.1 Average-Case SNARGs and SRDS Based on Multi-signatures

Average-case SNARGs. We consider a notion of SNARGs for *average-case* instances of an NP language $\mathcal{L}$. This constitutes a weaker primitive than standard SNARGs (as per [10]), which requires soundness against worst-case instances, and may be viewed as a variant of the notion for *cryptographically hard languages* considered in [16]. An average-case SNARG for a language $\mathcal{L}$ is parametrized by an efficiently sampleable distribution $\mathcal{D}_{\mathsf{yes}}$ over the instance-witness pairs in $\mathcal{L}$, and an efficiently sampleable distribution $\mathcal{D}_{\mathsf{no}}$ over instances outside of $\mathcal{L}$. In a similar way to regular SNARGs, average-case SNARGs consist of setup, prover, and verification algorithms. Intuitively, given any instance-witness pair (x, w) in $\mathcal{D}_{\mathsf{yes}}$, the prover algorithm should output a verifying succinct proof with overwhelming probability. At the same time, it should be hard for an adversary to compute a verifying proof for a *random* instance x from $\mathcal{D}_{\mathsf{no}}$.

Definition 6.1 (average-case SNARG for $(\mathcal{D}_{\mathsf{yes}}, \mathcal{D}_{\mathsf{no}})$)**.** *Let $\mathcal{L}$ be an NP language associated with a relation $R_{\mathcal{L}}$, and let $\mathcal{D}_{\mathsf{yes}}$ and $\mathcal{D}_{\mathsf{no}}$ be efficiently sampleable distributions over $(x, w) \in R_{\mathcal{L}}$ and $x \notin \mathcal{L}$, respectively. A succinct non-interactive argument system Π for average-case $\mathcal{L}$, parametrized by the distributions $(\mathcal{D}_{\mathsf{yes}}, \mathcal{D}_{\mathsf{no}})$, is defined by PPT algorithms $(\mathsf{S.Setup}, \mathsf{S.Prove}, \mathsf{S.Verify})$ as follows:*

- $\mathsf{S.Setup}(1^\kappa, 1^n) \to \mathsf{crs}$. *On input the security parameter κ and the instance size n, the setup algorithm outputs a common reference string* crs.
- $\mathsf{S.Prove}(\mathsf{crs}, x, w) \to \pi$. *On input the* crs *and an instance-witness pair $(x, w) \in R_{\mathcal{L}}$, the prover algorithm outputs a proof π.*
- $\mathsf{S.Verify}(\mathsf{crs}, x, \pi) \to b$. *On input the* crs*, an instance x, and a proof π, the verification algorithm outputs a bit $b \in \{0, 1\}$.*

We require the argument system to satisfy the following properties:

1. **Succinctness:** $|\pi| = \mathsf{poly}(\log n, \kappa)$ for all $(x, w) \leftarrow \mathcal{D}_{\mathsf{yes}}(1^n)$.
2. **Completeness:** For any instance-witness pair (x, w) in the support of $\mathcal{D}_{\mathsf{yes}}$, it holds that
$$\Pr\left[\mathsf{S.Verify}(\mathsf{crs}, x, \pi) = 1 \;\middle|\; \mathsf{crs} \leftarrow \mathsf{S.Setup}(1^\kappa, 1^n), \pi \leftarrow \mathsf{S.Prove}(\mathsf{crs}, x, w)\right] \geq 1 - \mathsf{negl}(n, \kappa).$$
3. **Average-case soundness:** For any non-uniform PPT prover $\mathcal{P}^*$, it holds that
$$\Pr\left[\mathsf{S.Verify}(\mathsf{crs}, x, \pi) = 1 \;\middle|\; \mathsf{crs} \leftarrow \mathsf{S.Setup}(1^\kappa, 1^n), x \leftarrow \mathcal{D}_{\mathsf{no}}(1^n), \pi \leftarrow \mathcal{P}^*(\mathsf{crs}, x)\right] \leq \mathsf{negl}(n, \kappa).$$

SRDS based on multi-signatures. We consider implications of SRDS constructions based on an underlying multi-signature scheme (see Appendix A.4) in the following sense. While rigorously specifying the notion is rather involved, at a high level, such a scheme is one that satisfies three natural properties:

1. **Structure:** The aggregate SRDS signature is a pair $(\sigma_{\mathsf{ms}}, \pi)$, where σ_{ms} is a multi-signature and π is some (small) auxiliary information (of size $\tilde{O}(1)$).
2. **Completeness:** Given a valid multi-signature σ_{ms} on a message m corresponding to a sufficiently large subset of keys $\{\mathsf{vk}_i\}_{i \in S}$, together with knowledge of the subset S, it is easy to compute a valid SRDS signature certifying m.

3. **Soundness:** Given a set of honestly generated verification keys, it is difficult to output a verifying SRDS signature $(\sigma_{\mathsf{ms}}, \pi)$ on a message m such that the multi-signature σ_{ms} does not verify on m against any sufficiently large subset of keys.

In order to prove that sufficiently many parties agree on a message m, it suffices to certify that there exists an s-size subset of parties (where s is sufficiently large) who agree on the same message m. Therefore, moving forward for SRDS based on multi-signatures, we only focus on proving that exactly s parties agree on a particular message. We now formalize SRDS based on multi-signatures.

Definition 6.2 (SRDS based on multi-signatures)**.** *An SRDS scheme* $\Pi = (\mathsf{Setup}(1^\kappa, 1^n), \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Aggregate}, \mathsf{Verify})$ *with bare PKI, is based on a multi-signature scheme* $(\mathsf{MS.Setup}, \mathsf{MS.KeyGen}, \mathsf{MS.Sign}, \mathsf{MS.Verify}, \mathsf{MS.Combine}, \mathsf{MS.MVerify})$ *if there exists* $s(n) \in \Theta(n)$*, for which the following hold:*

- **Structure.** The SRDS has the following structure:
  - $\mathsf{Setup}(1^\kappa, 1^n)$: Outputs public parameters of the form $\mathsf{pp_{srds}} = (\mathsf{pp_{ms}}, \mathsf{pp}_2)$, where $\mathsf{pp_{ms}} \leftarrow \mathsf{MS.Setup}(1^\kappa)$ and pp_2 are (potentially) additional public parameters.
  - $\mathsf{KeyGen}(\mathsf{pp_{srds}})$: Parses $\mathsf{pp_{srds}} = (\mathsf{pp_{ms}}, \mathsf{pp}_2)$ and outputs $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{MS.KeyGen}(\mathsf{pp_{ms}})$.
  - $\mathsf{Aggregate}$: Any SRDS σ output by $\mathsf{Aggregate}$ is of the form $(\sigma_{\mathsf{ms}}, \pi) \in \mathcal{X}_{\mathsf{ms}} \times \{0,1\}^{\mathsf{poly}(\log n, \kappa)}$, where $\sigma_{\mathsf{ms}} \in \mathcal{X}_{\mathsf{ms}}$ is a multi-signature (in the support of $\mathsf{MS.Combine}$).
- **Completeness.** There exists a PPT algorithm P, such that with overwhelming probability (in (n, κ)) over honestly sampled $\mathsf{pp_{srds}} = (\mathsf{pp_{ms}}, \mathsf{pp}_2) \leftarrow \mathsf{Setup}(1^\kappa, 1^n)$ and independently sampled verification keys $(\mathsf{vk}_i, \cdot) \leftarrow \mathsf{KeyGen}(\mathsf{pp_{srds}})$ for $i \in [n]$, the following holds:

  Let $m \in \mathcal{M}$, let $S \subseteq [n]$ of size $s(n)$, and let $\sigma_{\mathsf{ms}} \in \mathcal{X}_{\mathsf{ms}}$ be a multi-signature satisfying $\mathsf{MS.MVerify}(\mathsf{pp_{ms}}, \mathsf{vk}_1, \ldots, \mathsf{vk}_n, S, m, \sigma_{\mathsf{ms}}) = 1$. Then, with overwhelming probability (in (n, κ)) over the auxiliary information $\pi \leftarrow \mathsf{P}(\mathsf{pp_{srds}}, \mathsf{vk}_1, \ldots, \mathsf{vk}_n, S, m, \sigma_{\mathsf{ms}})$, it holds that $\mathsf{Verify}(\mathsf{pp_{srds}}, \mathsf{vk}_1, \ldots, \mathsf{vk}_n, m, (\sigma_{\mathsf{ms}}, \pi)) = 1$.
- **Soundness.** Every non-uniform polynomial-time adversary $\mathcal{A}$ wins the following experiment with at most negligible probability (in (n, κ)):
  1. The challenger samples $\mathsf{pp_{srds}} = (\mathsf{pp_{ms}}, \mathsf{pp}_2) \leftarrow \mathsf{Setup}(1^\kappa, 1^n)$ and for every $i \in [n]$, sets $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp_{srds}})$.
  2. The challenger gives $\mathcal{A}$ the values $(\mathsf{pp_{srds}}, \mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ and get back $(m, (\sigma_{\mathsf{ms}}, \pi))$.
  3. The adversary wins the game if and only if $\mathsf{Verify}(\mathsf{pp_{srds}}, \mathsf{vk}_1, \ldots, \mathsf{vk}_n, m, (\sigma_{\mathsf{ms}}, \pi)) = 1$ and, in addition, there does not exist a subset $S \subseteq [n]$ of size $s(n)$, such that $\mathsf{MS.MVerify}(\mathsf{pp_{ms}}, \mathsf{vk}_1, \ldots, \mathsf{vk}_n, S, m, \sigma_{\mathsf{ms}}) = 1$.

  (Observe that the output of this experiment is not necessarily efficiently computable.)

Note that for our purposes it will suffice to consider soundness against an adversary $\mathcal{A}$ who does not have access to a subset of keys $\{\mathsf{sk}_i\}_{i \in S}$ or to a signing oracle. This is a weaker requirement than a comparable soundness guarantee when given corrupted secret keys, which means a barrier against such primitive is stronger.

6.2 Multi-signatures of Lu et al. [77] and Subset-Product

We proceed to show that any SRDS based on the multi-signature scheme of Lu et al. [77] ("LOSSW") as defined above implies SNARGs for a natural average-case version of Subset-Product. Intuitively, an LOSSW multi-signature σ_{ms} for a set of parties $S \subseteq [n]$ is equivalent to a single signature under the *product* of the verification keys $\prod_{i\in S} \mathsf{vk}_i$. In turn, existence of a large set of approving parties S for σ_{ms} is equivalent to existence of a large set of verification keys $\{\mathsf{vk}_i\}_{i\in S}$ for which $\prod_{i\in S} \mathsf{vk}_i$ takes a particular desired value determined by σ_{ms}.

The multi-signature scheme of Lu et al. [77], is based on the *bilinear computational Diffie-Hellman* (BCDH) assumption, parametrized by a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G}$ and $\mathbb{G}_T$ are multiplicative cyclic groups of order p. In Appendix A.5, we formally describe the LOSSW multi-signature scheme; their scheme roughly works as follows:

Construction 6.3 (LOSSW Multi-signatures [77])**.**

- $\mathsf{MS.Setup}(1^\kappa)$*: The setup algorithm outputs public parameters* $\mathsf{pp}_{\mathsf{ms}} = (\mathbb{G}, \mathbb{G}_T, p, g, e)$*.*
- $\mathsf{MS.KeyGen}(1^\kappa)$*: The key-generation algorithm outputs a signing key* $\mathsf{sk} \in \mathbb{Z}_p$*, and the corresponding verification key* $\mathsf{vk} \in \mathbb{G}_T$*, computed as* $e(g,g)^{\mathsf{sk}}$*.*
- $\mathsf{MS.Sign}(\mathsf{pp}_{\mathsf{ms}}, \mathsf{sk}, m)$*: There is a deterministic function* f_{msg} *that takes the public parameters* $\mathsf{pp}_{\mathsf{ms}}$ *and the message* $m \in \mathcal{M}$ *as input and outputs an element in* $\mathbb{G}$ *(see Appendix A.5 for full specification). Given this* f_{msg}*, a signature on* m *with secret key* sk *is generated by sampling* $r \leftarrow \mathbb{Z}_p$ *and computing* $\sigma = (\mathsf{sig}_1, \mathsf{sig}_2)$ *as follows:*
$$\mathsf{sig}_1 = g^{\mathsf{sk}} \cdot (f_{\mathsf{msg}}(\mathsf{pp}_{\mathsf{ms}}, m))^r \quad \textit{and } \mathsf{sig}_2 = g^r.$$
- $\mathsf{MS.Combine}(\mathsf{pp}_{\mathsf{ms}}, \mathsf{vk}_1, \ldots, \mathsf{vk}_n, \{\sigma_i\}_{i\in S}, m)$*: Given a set of individual signatures* $\{\sigma_i\}_{i\in S}$ *on a message* $m \in \mathcal{M}$*, the combine function parses each* σ_i *as* $(\mathsf{sig}_1^{(i)}, \mathsf{sig}_2^{(i)})$ *and computes:*
$$\mathsf{sig}_1 = \prod_{i\in S} \mathsf{sig}_1^{(i)} \quad \textit{and } \mathsf{sig}_2 = \prod_{i\in S} \mathsf{sig}_2^{(i)}.$$
The output is the combined multi-signature $\sigma_{\mathsf{ms}} = (\mathsf{sig}_1, \mathsf{sig}_2)$*.*

  Remark. *Recall that* $\mathsf{sig}_1^{(i)}$ *and* $\mathsf{sig}_2^{(i)}$ *for each* $i \in S$*, is of the form*
$$\mathsf{sig}_1^{(i)} = g^{\mathsf{sk}_i} \cdot (f_{\mathsf{msg}}(\mathsf{pp}_{\mathsf{ms}}, m))^{r_i} \quad \textit{and } \mathsf{sig}_2^{(i)} = g^{r_i}$$
for some $r_i \in \mathbb{Z}_p$*. Therefore,*
$$\mathsf{sig}_1 = g^{\mathsf{sk}^*} \cdot (f_{\mathsf{msg}}(\mathsf{pp}_{\mathsf{ms}}, m))^{r^*} \quad \textit{and } \mathsf{sig}_2 = g^{r^*},$$
where $\mathsf{sk}^* = \sum_{i\in S} \mathsf{sk}_i$ *and* $r^* = \sum_{i\in S} r_i$*. Note that the multi-signature* $\sigma_{\mathsf{ms}} = (\mathsf{sig}_1, \mathsf{sig}_2)$ *can now be viewed as an individual signature on* m *corresponding to secret key* sk^* *and randomness* r^**.*
- $\mathsf{MS.MVerify}(\mathsf{pp}_{\mathsf{ms}}, \mathsf{vk}_1, \ldots, \mathsf{vk}_n, S, m, \sigma_{\mathsf{ms}})$*: Given a message* m*, a multi-signature* σ_{ms} *and the corresponding set* S *of verification keys, the verification algorithm outputs 1 if and only if*
$$e(\mathsf{sig}_1, g) \cdot e(\mathsf{sig}_2, f_{\mathsf{msg}}(\mathsf{pp}_{\mathsf{ms}}, m))^{-1} = \prod_{i\in S} \mathsf{vk}_i.$$
Note that the same algorithm can be used to verify individual signatures (with $S = \{i\}$*).*

Average-case subset-product problem. We proceed to show a connection between any SRDS based on LOSSW multi-signatures, and the following average-case version of the Subset-Product problem.

Definition 6.4 (average-case subset-product problem)**.** *Let $s = s(n)$ be an integer and let $\mathbb{G}$ be a multiplicative group. Given an instance $x = (a_1, \ldots, a_n, t) \in \mathbb{G}^{n+1}$, the $(s, \mathbb{G})$-Subset-Product problem is the problem of deciding if there exists a subset $S \subseteq [n]$ of size $|S| = s$, such that $\prod_{i \in S} a_i = t$. All such instances are said to be in the $(s, \mathbb{G})$-Subset-Product language $\mathcal{L}_\times$.*

We consider the average-case version of this problem characterized by the following two distributions:

1. *$\mathcal{D}_{\mathsf{yes}}(1^n) \to (x, w)$: For $i \in [n]$, sample $a_i \in \mathbb{G}$ uniformly at random. Sample a set $S \subseteq [n]$ of size s uniformly at random. Set $t = \prod_{i \in S} a_i$. Output $x = (a_1, \ldots, a_n, t)$ and $w = S$.*

2. *$\mathcal{D}_{\mathsf{no}}(1^n) \to x$: For $i \in [n]$, sample $a_i \in \mathbb{G}$ uniformly at random. Sample a target $t \in \mathbb{G}$ uniformly at random. Output $x = (a_1, \ldots, a_n, t)$.*

Note that for appropriate parameter regimes, $\mathcal{D}_{\mathsf{no}}$ yields instances ($x \notin \mathcal{L}_\times$) with high probability. For example, consider $s = n/2$ and $\mathbb{G} = \mathbb{Z}_M^*$ for $M = 2^{4n}$: the probability that there exists a subset S such that $\prod_{i \in S} a_i$ is equal to a randomly chosen value t is approximately 2^{-2n}.

Remark. Subset-Product is a well-studied problem, with known NP-hardness results in the worst case, and conjectured hardness in the average-case version considered above.

- **Hardness of worst-case subset product:** For $\mathbb{G} = \mathbb{Z}_M^*$ and $s \in \Theta(n)$, the hardness of *worst-case* $(s, \mathbb{G})$-Subset-Product depends on the density $n/\log M$ of the instance. For $M = 2^{\Theta(n)}$ (i.e., $n/\log M \in \Theta(1)$), there exists $s \in \Theta(n)$ for which the $(s, \mathbb{Z}_M^*)$-Subset-product is NP-complete [65, 53, 63, 79].[22]

- **Hardness of average-case subset product:** The average-case version of Subset-Product is thought to be computationally hard when $n/\log M$ is constant or even $O(1/\log n)$ [63], with the best known algorithms requiring at least $2^{\Omega(n)}$ time. Hardness of distinguishing between distributions $\mathcal{D}_{\mathsf{yes}}$ and $\mathcal{D}_{\mathsf{no}}$ as above is used (in an indirect way) as a computational hardness assumption in an assortment of cryptographic systems [63, 3, 89, 90, 86, 79].[23]

LOSSW-based SRDS implies SNARGs for average-case subset-product. We now show that an SRDS scheme based on the LOSSW multi-signature scheme implies the existence of SNARGs for average-case Subset-Product over the target group $\mathbb{G}_T$ for the underlying bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Intuitively, the construction takes the following form.

Given an instance $x = (a_1, \ldots, a_n, t) \in \mathbb{G}_T^{n+1}$ (coming from either $\mathcal{D}_{\mathsf{yes}}$ or $\mathcal{D}_{\mathsf{no}}$), we will interpret $a_1, \ldots, a_n$ and $a_{n+1} = t^{-1}$ as $n+1$ *verification keys* $\{\mathsf{vk}_i\}_{i \in [n+1]}$ for the SRDS scheme. The succinct

[22] The Subset-Sum problem for arbitrary subset-sizes in this parameter regime was amongst the initial 21 problems that were shown to be NP-complete by Karp [65]. There exists a generic reduction to reduce any instance of 3-SAT to an instance of the Subset-Sum problem for arbitrary subset-sizes. A slight modification to this reduction shows that there exists some $s \in \Theta(n)$ for which $(s, \mathbb{Z}_M)$-Subset-Sum problem is also NP-complete. There also exists a generic reduction from any instance of the $(s, \mathbb{Z}_M)$-Subset-Sum problem, where $\mathbb{Z}_M$ is an additive group of order M, to an instance of the $(s, \mathbb{G}_M)$-Subset-Product problem, where $\mathbb{G}_M$ is a cyclic multiplicative group of order M (with efficient exponentiation).

[23] This follows from the constructions in [63, 3, 89, 90, 86, 79] based on the Subset-Sum problem.

proof certifying that x is in the language will be an SRDS signature on a message $m \in \mathcal{M}$ with respect to the set of parties $S \cup \{n+1\}$ for which $\prod_{i \in S} a_i = t$. The scheme is succinct by construction; the required completeness and average-case soundness properties will hold as follows:

- *Completeness:* If x was generated as $(x, w) \leftarrow \mathcal{D}_{\mathsf{yes}}$ for $w = S \subseteq [n]$, then by definition $\prod_{i \in S} a_i = t$ and consequently $t^{-1} \cdot \prod_{i \in S} a_i = 1$. Knowledge of the corresponding set of verification keys thus enables the prover to generate a valid LOSSW multi-signature under these keys, using the *trivial* $\mathsf{sk}^* = 0$ for $\mathsf{vk}^* = \prod_{i \in S \cup \{n+1\}} \mathsf{vk}_i = 1$. By the completeness of the LOSSW-based SRDS (Definition 6.2), the prover can then translate this multi-signature to a valid SRDS.

- *Average-case Soundness:* On the other hand, if x was generated as $x \leftarrow \mathcal{D}_{\mathsf{no}}$, then since t and consequently t^{-1} is uniform conditioned on $a_1, \ldots, a_n$, the resulting verification keys $\{\mathsf{vk}_i\}_{i \in [n+1]}$ are *jointly uniform*. Thus (for appropriate parameters n and $|\mathbb{G}_T|$), soundness of the argument system holds from the soundness property of the LOSSW-based SRDS (see Definition 6.2).

Lemma 6.5. *Assume there exists an SRDS scheme based on the LOSSW multi-signature scheme, where* $\mathsf{Setup}(1^\kappa, 1^n)$ *generates* $\mathsf{pp}_{\mathsf{ms}} = (\mathbb{G}, \mathbb{G}_T, p, g, e)$*, as per Definition 6.2, with* $n / \log |\mathbb{G}_T| < 1$*. Let* $0 < \alpha < 1$ *be a constant and let* $s(n) = \alpha \cdot n$*. Then, there exist SNARGs for average-case* $(s(n), \mathbb{G}_T)$*-Subset-Product (as defined in Definition 6.4).*

Proof. We construct average-case SNARGs for $(s, \mathbb{G}_T)$-Subset-Product using an SRDS scheme based on the LOSSW multi-signature scheme as per Definition 6.2. Recall that the LOSSW multi-signature scheme is parametrized by a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G}$ and $\mathbb{G}_T$ are multiplicative cyclic groups of order p. Let $\mathcal{M}$ be the message domain of the LOSSW multi-signature scheme.

1. $\mathsf{S.Setup}(1^\kappa, 1^n)$: Run the setup of the SRDS scheme $\mathsf{pp}_{\mathsf{srds}} = (\mathsf{pp}_{\mathsf{ms}}, \mathsf{pp}_2) \leftarrow \mathsf{Setup}(1^\kappa, 1^n)$ and output $\mathsf{crs} = \mathsf{pp}_{\mathsf{srds}}$.

2. $\mathsf{S.Prove}(\mathsf{crs}, x, w)$: Given an average-case yes instance-witness pair, $(x, w) \leftarrow \mathcal{D}_{\mathsf{yes}}(1^n)$ of the form $x = (a_1, \ldots, a_n, t)$ and $w = S$, proceed as follows:

   - Parse $\mathsf{crs} = (\mathsf{pp}_{\mathsf{ms}}, \mathsf{pp}_2)$ and interpret the set $\{a_1, \ldots, a_n, t^{-1}\}$ as a set of $(n+1)$ verification keys $\{\mathsf{vk}_1, \ldots, \mathsf{vk}_{n+1}\}$. Note that $\prod_{i \in S'} \mathsf{vk}_i = 1$ for $S' = S \cup \{n+1\}$.
   - Since in the LOSSW multi-signature scheme, (aggregate) verification key $\mathsf{vk}^* = \prod_{i \in S'} \mathsf{vk}_i$ corresponds to a valid signing key $\mathsf{sk}^* = \sum_{i \in S'} \mathsf{sk}_i$, where $\mathsf{vk}^* = e(g, g)^{\mathsf{sk}^*}$, it holds that if $\mathsf{vk}^* = 1$, then $\mathsf{sk}^* = 0$. Choose an arbitrary $m \in \mathcal{M}$ and sample $r \leftarrow \mathbb{Z}_p$. Compute an LOSSW signature $\sigma_{\mathsf{ms}} = (\mathsf{sig}_1, \mathsf{sig}_2)$ on m with respect to $\mathsf{vk}^* = 1$, where
   $$\mathsf{sig}_1 = g^0 \cdot (f_{\mathsf{msg}}(\mathsf{pp}_{\mathsf{ms}}, m))^r \quad \text{and} \quad \mathsf{sig}_2 = g^r.$$
   - Use the algorithm P (that is guaranteed to exist by Definition 6.2) to compute the auxiliary information $\pi \leftarrow \mathsf{P}(\mathsf{pp}_{\mathsf{srds}}, \mathsf{vk}_1, \ldots, \mathsf{vk}_{n+1}, S', m, \sigma_{\mathsf{ms}})$ from the signature σ_{ms} and the set $S' \subseteq [n+1]$.
   - Finally output $(m, \sigma_{\mathsf{ms}}, \pi)$.

3. $\mathsf{S.Verify}(\mathsf{crs}, x, (m, \sigma_{\mathsf{ms}}, \pi))$: Parse $x = (a_1, \ldots, a_n, t)$ and proceed as follows:

- Parse $\mathsf{crs} = (\mathsf{pp}_{\mathsf{ms}}, \mathsf{pp}_2)$ and interpret the set $\{a_1, \ldots, a_n, t^{-1}\}$ as a set of $(n+1)$ verification keys $\{\mathsf{vk}_1, \ldots, \mathsf{vk}_{n+1}\}$.
- Run the verification algorithm of the LOSSW multi-signature scheme with respect to combined verification key $\mathsf{vk}^* = 1$, i.e., parse $\sigma_{\mathsf{ms}} = (\mathsf{sig}_1, \mathsf{sig}_2)$ and check if

$$e(\mathsf{sig}_1, g) \cdot e(\mathsf{sig}_2, f_{\mathsf{msg}}(\mathsf{pp}_{\mathsf{ms}}, m))^{-1} = 1.$$

In other words, compute

$$b' = \begin{cases} 1, & \text{if } e(\mathsf{sig}_1, g) \cdot e(\mathsf{sig}_2, f_{\mathsf{msg}}(\mathsf{pp}_{\mathsf{ms}}, m))^{-1} = 1 \\ 0, & \text{otherwise} \end{cases}.$$

- Run the verification algorithm of the SRDS scheme on $(\sigma_{\mathsf{ms}}, \pi)$ with respect to m:

$$b \leftarrow \mathsf{Verify}(\mathsf{pp}_{\mathsf{srds}}, \mathsf{vk}_1, \ldots, \mathsf{vk}_{n+1}, m, (\sigma_{\mathsf{ms}}, \pi)).$$

- Output $b \wedge b'$.

We now argue succinctness, completeness, and average-case soundness for this construction:

Succinctness. Succinctness follows from the succinctness of the SRDS scheme.

Completeness. Given any average-case yes instance-witness pair $(x, w) \leftarrow \mathcal{D}_{\mathsf{yes}}(1^n)$, with $x = (a_1, \ldots, a_n, t)$ and $w = S$, it holds that $\prod_{i \in S} a_j = t$ or equivalently $t^{-1} \cdot \prod_{i \in S} a_j = 1$. Let $S' = S \cup \{n+1\}$. Recall that in the LOSSW scheme

$$\prod_{i \in S'} \mathsf{vk}_i = \prod_{i \in S'} e(g, g)^{\mathsf{sk}_i} = e(g, g)^{\sum_{i \in S'} \mathsf{sk}_i},$$

where sk_i is the secret key associated with vk_i. It follows that if $\prod_{i \in S'} \mathsf{vk}_i = 1$ then $\sum_{i \in S'} \mathsf{sk}_i = 0$. Hence, $\sigma_{\mathsf{ms}} = \left((f_{\mathsf{msg}}(\mathsf{pp}_{\mathsf{ms}}, m))^r, g^r \right)$ is a valid multi-signature on m with respect to $\{\mathsf{sk}_i\}_{i \in S'}$. Since the multi-signature verifies $\mathsf{MS.MVerify}(\mathsf{pp}_{\mathsf{ms}}, \mathsf{vk}_1, \ldots, \mathsf{vk}_{n+1}, S', m, \sigma_{\mathsf{ms}}) = 1$, completeness of SRDS based on a multi-signature scheme (see Definition 6.2) implies that the output of P, given this signature and S', will be a valid SRDS signature.

Average-case soundness. Recall that each of the values $(a_1, \ldots, a_n, t)$ in $x \leftarrow \mathcal{D}_{\mathsf{no}}(1^n)$ are sampled uniformly at random. Since t is a randomly sampled value, so is t^{-1}. Therefore, the verification keys $\{\mathsf{vk}_1, \ldots, \mathsf{vk}_n, \mathsf{vk}_{n+1}\}$, where $\mathsf{vk}_i = a_i$ for $i \in [n]$ and $\mathsf{vk}_{n+1} = t^{-1}$, are uniformly distributed over $\mathbb{G}_T^{n+1}$. Since by assumption, $n / \log |\mathbb{G}_T| < 1$, it holds with overwhelming probability (bounded by $2^{n+1} / |\mathbb{G}_T|$) that there does not exist a subset $S' \subseteq [n+1]$ of size $s+1$, such that $\prod_{i \in S'} \mathsf{vk}_i = 1$.

Given $(m, \sigma_{\mathsf{ms}}, \pi)$, we check if: (1) σ_{ms} is a valid multi-signature on m with respect to $\mathsf{vk}^* = 1$, and (2) if $(\sigma_{\mathsf{ms}}, \pi)$ is a valid SRDS on m. Recall that given a multi-signature $\sigma_{\mathsf{ms}} = (\mathsf{sig}_1, \mathsf{sig}_2)$, a message m, the public parameters $\mathsf{pp}_{\mathsf{ms}}$, and a set of verification keys $\{\mathsf{vk}_i\}_{i \in S}$, the verification algorithm of the LOSSW multi-signature scheme checks if

$$e(\mathsf{sig}_1, g) \cdot e(\mathsf{sig}_2, f_{\mathsf{msg}}(\mathsf{pp}_{\mathsf{ms}}, m))^{-1} = \prod_{i \in S} \mathsf{vk}_i.$$

In other words, given a valid multi-signature σ_{ms} on a message m, there exists a unique aggregate verification key $\prod_{i \in S} \mathsf{vk}_i$ for which σ_{ms} verifies. Therefore, if check (1) goes through, then $\mathsf{vk}^* = 1$ is the only aggregate verification key for which σ_{ms} is a valid multi-signature on m. As argued earlier, with a high probability there does not exist a subset $S' \subseteq [n+1]$ such that $\Pi_{i\in S'}\mathsf{vk}_i = 1$. Also, from the soundness of SRDS based on a multi-signature scheme (Definition 6.2), we know that if there does not exist a subset $S' \subseteq [n+1]$ of size $s+1$, such that σ_{ms} is a valid multi-signature on m with respect to $\{\mathsf{vk}_i\}_{i\in S'}$, then the probability of an adversary computing a valid SRDS $(\sigma_{\mathsf{ms}}, \pi)$ on a message m is negligible. Soundness now follows from the soundness of SRDS based on a multi-signature scheme. □

6.3 General Multi-Signatures and the Subset-f Problem

Although the proof of Lemma 6.5 depends on the specific LOSSW multi-signature scheme, the overall approach only depends on certain properties of that scheme; in particular, there is no inherent reliance on the structure of *multiplication* of keys and Subset-Product. Motivated by this observation, in this section, we start by exploring hardness of *Subset-f problems* for a more general class of functions f, focusing on the class of *elementary symmetric polynomials* ϕ_ℓ. We begin by demonstrating (worst-case) NP-hardness results for Subset-ϕ_ℓ. We then abstract out the properties used in Lemma 6.5 (deemed "SNARG-compliance"), and show that existence of SRDS based on a SNARG-compliant multi-signature scheme implies existence of SNARGs for corresponding Subset-ϕ_ℓ problems.

The subset-f problem. We first define the following analogous variant of average-case Subset-Product problem for more general functions f. We restrict our attention to the natural setting of symmetric functions f; one can extend to arbitrary f, e.g., given a canonical ordering of inputs.

Definition 6.6 (average-case subset-f)**.** *Let $s = s(n)$ be an integer, let R be a ring, and let $f : R^s \to R$ an efficiently computable symmetric function. Given an instance $x = (a_1, \ldots, a_n, t) \in R^{n+1}$, the (s, R)-Subset-f $\mathsf{problem}$ is the problem of deciding if there exists a subset $S \subseteq [n]$ of size $|S| = s$, such that $f((a_i)_{i\in S}) = t$. Such instances are said to be in the (s, R)-Subset-f language $\mathcal{L}_f$.*

We consider the average-case version of this problem characterized by the following two distributions:

1. *$\mathcal{D}_{\mathsf{yes}}(1^n) \to (x, w)$: For each $i \in [n]$, sample $a_i \in R$ uniformly at random. Sample a set $S \subseteq [n]$ of size s uniformly at random. Set $t = f((a_i)_{i\in S})$. Output $x = (a_1, \ldots, a_n, t)$, $w = S$.*
2. *$\mathcal{D}_{\mathsf{no}}(1^n) \to x$: For each $i \in [n]$, sample $a_i \in R$ uniformly at random. Sample a target $t \in R$ uniformly at random. Output $x = (a_1, \ldots, a_n, t)$.*

We also consider a variant of the (s, R)-Subset-f problem, where the instance does not include the size of the subset, i.e., given an instance $x = (a_1, \ldots, a_n, t) \in R^{n+1}$, the R-Subset-f $\mathsf{problem}$ is the problem of deciding if there exists a subset $S \subseteq [n]$ of any size such that $f((a_i)_{i\in S}) = t$.

Note that Subset-f is within NP for any function f describable by a polynomial-size circuit. For appropriate parameter regimes, the hardness of Subset-f problems depends on the function f. In Theorem 6.9, we show that for rings (of appropriate size) with Hadamard product, Subset-f for all *elementary symmetric polynomials* f is NP-complete.

NP-hardness of subset-ϕ_ℓ. Recall that Hadamard product (also known as entry-wise product) takes two vectors of the same dimension and produces another vector of matching dimension where the i^{th} element of the resulting vector is a product of the i^{th} elements of the two input vectors.

Definition 6.7 (Hadamard product). *Let $\mathbb{F}$ be a field and let $\vec{a} = (a_1 \ldots, a_n), \vec{b} = (b_1 \ldots, b_n) \in \mathbb{F}^n$ be vectors of length n. The Hadamard product of $\vec{a}$ and $\vec{b}$ is the vector $\vec{a} \odot \vec{b} = (a_1 b_1, \ldots, a_n b_n) \in \mathbb{F}^n$.*

We now define elementary symmetric polynomials.

Definition 6.8 (elementary symmetric polynomials). *Let $n \in \mathbb{N}$ and $\ell \in [n]$. The elementary symmetric polynomial $\phi_\ell(x_1, \ldots, x_n)$ is defined as:*

$$\phi_\ell(x_1, \ldots, x_n) = \sum_{1 \le j_1 < \ldots < j_\ell \le n} x_{j_1} \cdot \ldots \cdot x_{j_\ell}.$$

In the following theorem, we show that for certain rings R that admit Hadamard product, and any elementary symmetric polynomial ϕ_ℓ, the (s, R)-Subset-ϕ_ℓ problem is NP-complete. In particular, we show this for suitably sized rings of the form $R = \mathbb{F}^n$, where for $\ell = 2$, the characteristic of the field must be at least 63 and for $\ell > 2$, the characteristic of the field must be at least $\ell + 2$.

Theorem 6.9. *There exists $s(n) \in \Theta(n)$ such that, for any field $\mathbb{F}$ with $\mathsf{char}(\mathbb{F}) \ge \max(\ell + 2, 63)$, any ring $R = \mathbb{F}^n$ of size $|R| = 2^{\Theta(n)}$ with Hadamard product, and any elementary symmetric polynomial ϕ_ℓ, the (s, R)-Subset-ϕ_ℓ problem is NP-complete.*

We next present a high-level overview of the proof; the full proof can be found in Appendix D.1. We start with a recap of the proof for NP-completeness of subset sum by Karp [65].

NP-completeness of subset sum. The proof for NP-completeness of $\mathbb{Z}_M$-Subset-Sum [65][24] shows a polynomial-time reduction from 3-SAT. At a high level, the reduction proceeds as follows: Given a 3-SAT instance with N variables $\{x_i\}_{i \in [N]}$ and m clauses $\{C_j\}_{j \in [m]}$, define a $\mathbb{Z}_M$-Subset-Sum instance with the following $2(N + m)$ numbers, each with $N + m$ digits, for $M \ge 10^{N+m}$:

1. For each input $i \in [N]$, define two numbers v_i and v'_i. The i^{th} least significant digit of both these numbers is set to 1. If $x_i \in C_j$, then the $(N + j)^{\text{th}}$ least significant digit of v_i is set to 1, else if $\neg x_i \in C_j$, then the $(N + j)^{\text{th}}$ least significant digit of v'_i is set to 1. The remaining digits in both these numbers are set to 0.

2. For each clause $j \in [m]$, define two numbers c_j^1 and c_j^2. The $(N + j)^{\text{th}}$ least significant digit of both these numbers is set to 1 and all the remaining digits are set to 0.

3. The target number t is also an $(N + m)$-digit number in which the first N digits are set to 1, while the remaining digits are set to 3.

Intuitively, given a satisfying assignment for the 3-SAT instance, the corresponding witness for the $\mathbb{Z}_M$-Subset-sum instance includes the following: For each $i \in [N]$, it includes v_i if $x_i = 1$, and v'_i if $x_i = 0$. For each $j \in [m]$, it includes any one of c_j^1 or c_j^2 if there are two literals with value 1 in the j^{th} clause, and both c_j^1 or c_j^2 if there is only one literal with a value of 1 in the j^{th} clause.

[24]Recall that this is a variation of the $(s, \mathbb{Z}_M)$-Subset-sum problem, where the instance does not include the size of the subset, as defined in Definition 6.6.

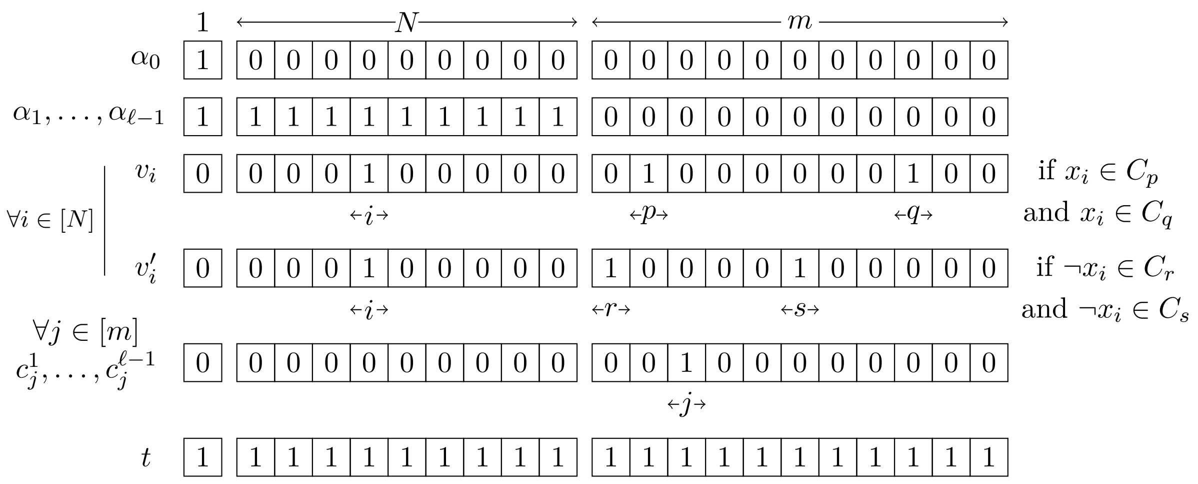

Figure 9: Reducing an instance of 3-SAT with N variables $\{x_i\}_{i\in[N]}$ and m clauses $\{C_j\}_{j\in[m]}$ to an instance of R-Subset-ϕ_ℓ for $\ell \geq 3$ with $n = \ell + 2N + (\ell - 1)m$ elements in R, where $R = \mathbb{F}^{1+N+m}$. Here, 0 (resp., 1) values inside the vectors refer to the 0 (resp., 1) element of $\mathbb{F}$.

Proof sketch of Theorem 6.9. We extend this reduction to show that R-Subset-ϕ_ℓ for $\ell > 1$ is also NP-complete, where R is a ring of appropriate size with Hadamard product. Each of the a_i (for $i \in [n]$) elements and the target value t in an instance of R-Subset-ϕ_ℓ is an element in R and thereby a vector of elements in $\mathbb{F}$. Unlike simple addition, since ϕ_ℓ is a sum of products, if (any) k^{th} entry in the target value is a non-zero element in $\mathbb{F}$, the solution to a yes instance of R-Subset-ϕ_ℓ must consist of at least ℓ elements with non-zero k^{th} entries. Therefore, depending on ℓ, we need to define additional elements in the reduction. We give an overview of our reduction from any 3-SAT instance to R-Subset-ϕ_ℓ for $\ell \geq 3$; the special case of $\ell = 2$ requires a slight modification that is addressed in Appendix D.1. In a similar way to Subset-Sum, this reduction can also be adjusted to show that there exists $s \in \Theta(n)$, for which (s, R)-Subset-ϕ_ℓ problem is also NP-complete, which is sketched in Appendix D.1.

Given a 3-SAT instance with N variables $\{x_i\}_{i\in[N]}$ and m clauses $\{C_j\}_{j\in[m]}$, define a R-Subset-ϕ_ℓ instance with $\ell + 2N + (\ell - 1)m$ elements, where $R = \mathbb{F}^{1+N+m}$. As shown in Figure 9, each of these elements is a vector of $1 + N + m$ elements in the field $\mathbb{F}$ and are defined as follows:

- An element $\alpha_0 \in R$, whose first entry is 1. All the remaining entries in α_0 correspond to 0.
- For each $k \in [\ell - 1]$, define $\alpha_k \in R$, whose first $N + 1$ entries correspond to 1, and the remaining entries correspond to 0.
- For each $i \in [N]$, define two elements $v_i \in R$ and $v'_i \in R$. The $(1 + i)^{\text{th}}$ entry of both these numbers is set to 1. If $x_i \in C_j$, then the $(1 + N + j)^{\text{th}}$ entry of v_i is set to 1, else if $\neg x_i \in C_j$, then the $(1 + N + j)^{\text{th}}$ entry of v'_i is set to 1. All the remaining entries correspond to 0.
- For each $j \in [m]$ and $k \in [\ell-1]$, define element $c_j^k \in R$. The $(1+N+j)^{\text{th}}$ entry in c_j^k corresponds to 1 and the remaining entries correspond to 0.
- The target element t is also a vector of $1 + N + m$ elements in $\mathbb{F}$, with all its entries set to 1.

Now, given a satisfying assignment for the 3-SAT instance, the corresponding witness for the R-Subset-ϕ_ℓ instance includes the following: It includes α_0 and each α_k for $k \in [\ell-1]$. For each $i \in [N]$, it includes v_i if $x_i = 1$, and v_i' if $x_i = 0$. For each $j \in [m]$, it includes any $\ell - 3$ of the elements c_j if all three literals in the j^{th} clause have value 1, else if any two literals have value 1 then it includes any $\ell - 2$ of the elements c_j and if only one of the literals has value 1 then all the $\ell - 1$ elements c_j are included in the witness. This guarantees that the value 1 appears precisely ℓ times in the column of each satisfied clause, so that ϕ_ℓ will evaluate to the target value 1 in these positions.

Similarly for soundness, a valid witness S for the R-Subset-ϕ_ℓ instance must include $a_0, \ldots, a_{\ell-1}$ in order to get ℓ times the value 1 in the first column. Apart from $a_1, \ldots, a_{\ell-1}$, the only other elements that have the value 1 in the next N columns are v_i and v_i'. For each $i \in [N]$, if both v_i and v_i' are included in the set S, a total of $\ell + 1$ elements in S will have value 1 in the $(i+1)^{\text{th}}$ column. The $(i+1)^{\text{th}}$ entry in the result obtained by applying ϕ_ℓ over such a set is $\ell + 1$. Since the characteristic of the field $\mathbb{F}$ is at least $\ell + 2$, we know that $\ell + 1 \neq 1$. Therefore, S can either contain v_i (implying $x_i = 1$) or v_i' (implying $\neg x_i = 1$) for each $i \in [N]$, but not both. For each of the last m columns, S can contain some or all of the elements c_j (for each $j \in [m]$). But since this set of c_j elements can only contribute at most $\ell - 1$ times the value 1 in the $(1 + N + j)^{\text{th}}$ column, we need at least one of the v or v' elements to contribute a 1 value to that column, in order to get a non-zero $(1 + N + j)^{\text{th}}$ entry in the result of ϕ_ℓ. This guarantees at least one variable with a value of 1 in each clause. We give a full proof of completeness and soundness for this reduction Appendix D.1. $\square$

SNARG-compliant multi-signatures and subset-ϕ_ℓ. We now identify the properties of the LOSSW multi-signature scheme used in Lemma 6.5 to provide the connection with average-case SNARGs. Roughly, these properties are:

- Verification keys are sampled independently and uniformly from the key-space of the multi-signature scheme. This property is important for arguing soundness in Lemma 6.5.

- The verification algorithm with keys $\{\mathsf{vk}_i\}_{i \in S}$, is equivalent to the verification algorithm with a single *aggregate* key $\mathsf{vk}_{\mathsf{agg}} = \prod_{i \in S} \mathsf{vk}_i$. In other words, there exists a *key-aggregation function* f_{agg} (e.g., $f_{\mathsf{agg}} = \prod$ in the LOSSW multi-signature scheme), such that the verification algorithm can be decomposed into first applying f_{agg} over the set of keys to obtain an aggregate key $\mathsf{vk}_{\mathsf{agg}}$ and then running some residual function $\mathsf{MS.Verify}_{\mathsf{agg\text{-}key}}$ to perform the remaining verification with respect to $\mathsf{vk}_{\mathsf{agg}}$.

- Given a valid multi-signature σ_{ms} on a message m, there exists a *unique and well-defined* aggregate key vk for which the residual function $\mathsf{MS.Verify}_{\mathsf{agg\text{-}key}}$ (as defined in the previous bullet) outputs 1. Moreover, this aggregate key is easy to compute. For example, for LOSSW, this property is crucially used for arguing soundness in Lemma 6.5.

- And finally, there exist degenerate keys $\mathsf{sk}_{\mathsf{deg}}$ and $\mathsf{vk}_{\mathsf{deg}}$ (e.g., $\mathsf{sk}_{\mathsf{deg}} = 0 \in \mathbb{G}$ and $\mathsf{vk}_{\mathsf{deg}} = 1 \in \mathbb{G}_T$ in the LOSSW multi-signature scheme) that allow forging a multi-signature on any message. This property is used in the completeness argument in Lemma 6.5.

We call multi-signature schemes that satisfy these properties as *SNARG-compliant* multi-signature schemes. We formally define this notion in Definition D.1 in Appendix D.2. Finally, by using the properties of a SNARG-compliant multi-signature scheme, we are able to prove a generalized version of Lemma 6.5. Namely, we show in Lemma D.2 that an SRDS scheme based on a SNARG-compliant multi-signature scheme with key-aggregation function $f_{\mathsf{agg}} = \phi_\ell$, implies SNARGs for average-case Subset-ϕ_ℓ.

Acknowledgements. 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 was done in part while the author was at Northeastern University and supported by NSF grant 1646671. A. Goel's work was done in part while visiting the FACT Center at IDC Herzliya, Israel, and while the author was at Johns Hopkins University, supported in part by an NSF CNS grant 1814919, NSF CAREER award 1942789 and Johns Hopkins University Catalyst award.

Bibliography

[1] I. Abraham, T. H. Chan, D. Dolev, K. Nayak, R. Pass, L. Ren, and E. Shi. Communication complexity of Byzantine agreement, revisited. In *Proceedings of the 38th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 317–326, 2019.

[2] I. Abraham, S. Devadas, D. Dolev, K. Nayak, and L. Ren. Synchronous Byzantine agreement with expected $\mathrm{O}(1)$ rounds, expected $\mathrm{o}(\mathrm{n}^2)$ communication, and optimal resilience. In *Financial Cryptography and Data Security*, pages 320–334, 2019.

[3] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC)*, pages 284–293, 1997.

[4] G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs. 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)*, pages 483–501, 2012.

[5] B. Barak and O. Goldreich. Universal arguments and their applications. *SIAM Journal on Computing*, 38(5):1661–1694, 2008.

[6] M. Bellare and G. Neven. 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)*, pages 390–399, 2006.

[7] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–10, 1988.

[8] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. Scalable zero knowledge via cycles of elliptic curves. In *33rd Annual International Cryptology Conference (CRYPTO), part II*, pages 276–294, 2014.

[9] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. Recursive composition and bootstrapping for SNARKs and proof-carrying data. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, pages 111–120, 2013.

[10] N. Bitansky, R. Canetti, A. Chiesa, S. Goldwasser, H. Lin, A. Rubinstein, and E. Tromer. The hunting of the SNARK. *Journal of Cryptology*, 30(4):989–1066, 2017.

[11] E. Blum, J. Katz, C. Liu-Zhang, and J. Loss. Asynchronous Byzantine agreement with subquadratic communication. In *Proceedings of the 18th Theory of Cryptography Conference (TCC), part I*, pages 353–380, 2020.

[12] A. Boldyreva. 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)*, pages 31–46, 2003.

[13] A. Boldyreva, C. Gentry, A. O'Neill, and D. H. Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS)*, pages 276–285, 2007.

[14] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *22nd International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 416–432, 2003.

[15] D. Boneh, M. Drijvers, and G. Neven. Compact multi-signatures for smaller blockchains. In *24th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT), part II*, pages 435–464, 2018.

[16] D. Boneh, Y. Ishai, A. Sahai, and D. J. Wu. Quasi-optimal SNARGs via linear multi-prover interactive proofs. In *37th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), part III*, pages 222–255, 2018.

[17] E. Boyle, S. Goldwasser, and S. Tessaro. 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)*, pages 356–376, 2013.

[18] E. Boyle, K. Chung, and R. Pass. Large-scale secure computation: Multi-party computation for (parallel) RAM programs. In *34th Annual International Cryptology Conference (CRYPTO), part II*, pages 742–762, 2015.

[19] E. Boyle, R. Cohen, D. Data, and P. Hubáček. Must the communication graph of MPC protocols be an expander? In *38th Annual International Cryptology Conference (CRYPTO), part III*, pages 243–272, 2018.

[20] E. Boyle, A. Jain, M. Prabhakaran, and C. Yu. The bottleneck complexity of secure multiparty computation. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 24:1–24:16, 2018.

[21] N. Braud-Santoni, R. Guerraoui, and F. Huc. Fast Byzantine agreement. In *Proceedings of the 32th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 57–64, 2013.

[22] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13 (1):143–202, 2000.

[23] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 136–145, 2001.

[24] R. Canetti. Universally composable signature, certification, and authentication. In *17th IEEE Computer Security Foundations Workshop, (CSFW)*, page 219, 2004.

[25] R. Canetti, D. Shahaf, and M. Vald. 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*, pages 265–296, 2016.

[26] T. H. Chan, R. Pass, and E. Shi. Consensus through herding. In *38th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), part I*, pages 720–749, 2019.

[27] T. H. Chan, R. Pass, and E. Shi. 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*, pages 246–265, 2020.

[28] N. Chandran, W. Chongchitmate, J. A. Garay, S. Goldwasser, R. Ostrovsky, and V. Zikas. 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*, pages 153–162, 2015.

[29] D. Chaum and E. van Heyst. Group signatures. In *10th Workshop on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 257–265, 1991.

[30] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 11–19, 1988.

[31] J. Chen and S. Micali. Algorand: A secure and efficient distributed ledger. *Theoretical Computer Science*, 777:155–183, 2019.

[32] A. Chiesa and E. Tromer. Proof-carrying data and hearsay arguments from signature cards. In *Innovations in Computer Science - ICS*, pages 310–331, 2010.

[33] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. 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)*, pages 383–395, 1985.

[34] R. Cohen. Asynchronous secure multiparty computation in constant time. In *Proceedings of the 19th International Conference on the Theory and Practice of Public-Key Cryptography (PKC), part II*, pages 183–207, 2016.

[35] R. Cohen, S. Coretti, J. A. Garay, and V. Zikas. Probabilistic termination and composability of cryptographic protocols. *Journal of Cryptology*, 32(3):690–741, 2019.

[36] R. Cohen, I. Haitner, N. Makriyannis, M. Orland, and A. Samorodnitsky. On the round complexity of randomized Byzantine agreement. In *Proceedings of the 33rd International Symposium on Distributed Computing (DISC)*, pages 12:1–12:17, 2019.

[37] R. Cohen, S. Coretti, J. A. Garay, and V. Zikas. Round-preserving parallel composition of probabilistic-termination cryptographic protocols. *Journal of Cryptology*, 34(2):12, 2021.

[38] S. Cohen, I. Keidar, and A. Spiegelman. Not a COINcidence: Sub-quadratic asynchronous Byzantine agreement WHP. In *Proceedings of the 34th International Symposium on Distributed Computing (DISC)*, pages 25:1–25:17, 2020.

[39] S. Coretti, J. A. Garay, M. Hirt, and V. Zikas. Constant-round asynchronous multi-party computation based on one-way functions. In *22nd International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT), part II*, volume 10032, pages 998–1021, 2016.

[40] I. Damgård and Y. Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *24th Annual International Cryptology Conference (CRYPTO)*, pages 378–394, 2005.

[41] I. Damgård and Y. Ishai. Scalable secure multiparty computation. In *25th Annual International Cryptology Conference (CRYPTO)*, pages 501–520, 2006.

[42] I. Damgård and J. B. Nielsen. Scalable and unconditionally secure multiparty computation. In *26th Annual International Cryptology Conference (CRYPTO)*, pages 572–590, 2007.

[43] I. Damgård, Y. Ishai, M. Krøigaard, J. B. Nielsen, and A. D. Smith. Scalable multiparty computation with nearly optimal work and resilience. In *27th Annual International Cryptology Conference (CRYPTO)*, pages 241–261, 2008.

[44] I. Damgård, Y. Ishai, and M. Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In *29th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 445–465, 2010.

[45] I. Damgård, S. Faust, and C. Hazay. Secure two-party computation with low communication. In *Proceedings of the 9th Theory of Cryptography Conference (TCC)*, pages 54–74, 2012.

[46] V. Dani, V. King, M. Movahedi, J. Saia, and M. Zamani. Secure multi-party computation in large networks. *Distributed Computing*, 30(3):193–229, 2017.

[47] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *8th Annual International Cryptology Conference (CRYPTO)*, pages 307–315, 1989.

[48] D. Dolev. The Byzantine generals strike again. *J. Algorithms*, 3(1):14–30, 1982.

[49] D. Dolev and R. Reischuk. Bounds on information exchange for Byzantine agreement. *Journal of the ACM*, 32(1):191–204, 1985.

[50] C. Dwork, D. Peleg, N. Pippenger, and E. Upfal. Fault tolerance in networks of bounded degree. *SIAM Journal on Computing*, 17(5):975–988, 1988.

[51] M. J. Fischer, N. A. Lynch, and M. Merritt. Easy impossibility proofs for distributed consensus problems. *Distributed Computing*, 1(1):26–39, 1986.

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

[53] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979. ISBN 0716710447.

[54] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. *Information and Computation*, 164(1):54–84, 2001.

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

[56] O. Goldreich. *Foundations of Cryptography – VOLUME 2: Basic Applications*. Cambridge University Press, 2004.

[57] O. Goldreich, S. Micali, and A. Wigderson. 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)*, pages 218–229, 1987.

[58] D. Gupta and A. Sahai. On constant-round concurrent zero-knowledge from a knowledge assumption. In *INDOCRYPT*, pages 71–88, 2014.

[59] Y. Harchol, I. Abraham, and B. Pinkas. Distributed SSH key management with proactive RSA threshold signatures. In *Proceedings of the 16th International Conference on Applied Cryptography and Network Security (ACNS)*, pages 22–43, 2018.

[60] S. Hohenberger and B. Waters. Synchronized aggregate signatures from the RSA assumption. In *37th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), part II*, pages 197–229, 2018.

[61] D. Holtby, B. M. Kapron, and V. King. Lower bound for scalable Byzantine agreement. *Distributed Computing*, 21(4):239–248, 2008.

[62] R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 230–235, 1989.

[63] R. Impagliazzo and M. Naor. Efficient cryptographic schemes provably as secure as subset sum. *Journal of Cryptology*, 9(4):199–216, 1996.

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

[65] R. M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, pages 85–103, 1972.

[66] J. Katz and C. Koo. On expected constant-round protocols for Byzantine agreement. In *25th Annual International Cryptology Conference (CRYPTO)*, pages 445–462, 2006.

[67] D. Kidron and Y. Lindell. Impossibility results for universal composability in public-key models and with fixed inputs. *Journal of Cryptology*, 24(3):517–544, 2011.

[68] V. King and J. Saia. 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)*, pages 464–478, 2009.

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

[70] V. King, J. Saia, V. Sanwalani, and E. Vee. Scalable leader election. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 990–999, 2006.

[71] V. King, S. Lonargan, J. Saia, and A. Trehan. Load balanced scalable Byzantine agreement through quorum building, with full information. In *Proceedings of the 12th International Conference on Distributed Computing and Networking (ICDCN)*, pages 203–214, 2011.

[72] L. Lamport. Constructing digital signatures from a one way function. Technical Report CSL-98, SRI International, 1979.

[73] L. Lamport, R. E. Shostak, and M. C. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[74] K. Lee, D. H. Lee, and M. Yung. Sequential aggregate signatures made shorter. In *Proceedings of the 11th International Conference on Applied Cryptography and Network Security (ACNS)*, pages 202–217, 2013.

[75] B. Libert, M. Joye, and M. Yung. Born and raised distributively: Fully distributed non-interactive adaptively-secure threshold signatures with short shares. *Theoretical Computer Science*, 645:1–24, 2016.

[76] Y. Lindell, A. Lysyanskaya, and T. Rabin. On the composition of authenticated Byzantine agreement. *Journal of the ACM*, 53(6):881–917, 2006.

[77] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters. Sequential aggregate signatures, multisignatures, and verifiably encrypted signatures without random oracles. *Journal of Cryptology*, 26(2): 340–373, 2013.

[78] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In *23rd International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 74–90, 2004.

[79] V. Lyubashevsky, A. Palacio, and G. Segev. Public-key cryptographic primitives provably as secure as subset sum. In *Proceedings of the 7th Theory of Cryptography Conference (TCC)*, pages 382–400, 2010.

[80] R. C. Merkle. A certified digital signature. In *8th Annual International Cryptology Conference (CRYPTO)*, pages 218–238, 1989.

[81] S. Micali. CS proofs (extended abstracts). In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 436–453, 1994.

[82] S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 120–130, 1999.

[83] S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignatures: extended abstract. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS)*, pages 245–254, 2001.

[84] J. B. Nielsen. A threshold pseudorandom function construction and its applications. In *21st Annual International Cryptology Conference (CRYPTO)*, pages 401–416, 2002.

[85] M. C. Pease, R. E. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, 1980.

[86] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 333–342, 2009.

[87] T. Rabin. A simplified approach to threshold and proactive RSA. In *17th Annual International Cryptology Conference (CRYPTO)*, pages 89–104, 1998.

[88] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 73–85, 1989.

[89] O. Regev. New lattice based cryptographic constructions. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 407–416, 2003.

[90] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 84–93, 2005.

[91] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *7th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, pages 552–565, 2001.

[92] V. Shoup. Practical threshold signatures. In *19th International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 207–220, 2000.

[93] P. Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *Proceedings of the 5th Theory of Cryptography Conference (TCC)*, pages 1–18, 2008.

[94] J. Wan, H. Xiao, S. Devadas, and E. Shi. Round-efficient Byzantine broadcast under strongly adaptive and majority corruptions. In *Proceedings of the 18th Theory of Cryptography Conference (TCC), part I*, pages 412–456, 2020.

[95] J. Wan, H. Xiao, E. Shi, and S. Devadas. Expected constant round Byzantine broadcast under dishonest majority. In *Proceedings of the 18th Theory of Cryptography Conference (TCC), part I*, pages 381–411, 2020.

[96] A. C. Yao. Protocols for secure computations (extended abstract). In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 160–164, 1982.

[97] M. Zamani, M. Movahedi, and J. Saia. Millions of millionaires: Multiparty computation in large networks. *IACR Cryptology ePrint Archive*, 2014:149, 2014.

A Preliminaries (Cont'd)

In this section, we provide additional definitions: for SNARKs, proof-carrying data and multi-signatures.

A.1 SNARKs

We follow the notation from [9]. The universal relation $\mathcal{R}_\mathcal{U}$ [5] is the set of instance-witness pairs $(y, w) = ((M, x, t), w)$, where $|y|, |w| \leq t$ and M is a random-access machine, such that M accepts (x, w) after running at most t steps. The universal language $\mathcal{L}_\mathcal{U}$ is the language corresponding to $\mathcal{R}_\mathcal{U}$.

A *non-interactive argument system* for $\mathcal{R}_\mathcal{U}$ is a triple of algorithms ($\mathsf{SNARK.Gen}$, $\mathsf{SNARK.Prover}$, $\mathsf{SNARK.Verify}$) with the following syntax:

- $\mathsf{SNARK.Gen}(1^\kappa, B) \to (\sigma, \tau)$: on input the security parameter κ and a time bound $B \in \mathbb{N}$, the generation algorithm outputs a common reference string (σ, τ) consisting of a prover reference string σ and a verification state τ.
- $\mathsf{SNARK.Prover}(\sigma, y, w) \to \pi$: given a prover reference string σ, an instance $y = (M, x, t)$ with $t \leq B$, and a witness w such that $(y, w) \in R$, the algorithm produces a proof π.
- $\mathsf{SNARK.Verify}(\tau, y, \pi) \to b$: given a verification state τ, an instance y, and a proof π, the verifier algorithm outputs a bit b.

Definition A.1 (SNARK). *A non-interactive argument system* ($\mathsf{SNARK.Gen}$*,* $\mathsf{SNARK.Prover}$*,* $\mathsf{SNARK.Verify}$) *for* $\mathcal{R}_\mathcal{U}$ *is a* SNARK *if the following conditions are satisfied:*

1. Completeness: *For every large enough security parameter* $\kappa \in \mathbb{N}$*, every time bound* $B \in \mathbb{N}$*, and every instance-witness pair* $(y, w) = ((M, x, t), w) \in \mathcal{R}_\mathcal{U}$ *with* $t \leq B$*,*

$$\Pr\left[\mathsf{SNARK.Verify}(\tau, y, \pi) = 1 \;\middle|\; \begin{array}{l} (\sigma, \tau) \leftarrow \mathsf{SNARK.Gen}(1^\kappa, B) \\ \pi \leftarrow \mathsf{SNARK.Prover}(\sigma, y, w) \end{array} \right] = 1.$$

2. Proof of Knowledge: *For every polynomial-size prover P^*, there exists a polynomial-size extractor $\mathcal{E}_{P^*}$ such that for every security parameter $\kappa \in \mathbb{N}$, every auxiliary input* $\mathsf{aux} \in \{0,1\}^{\mathsf{poly}(\kappa)}$, *and every time bound $B \in \mathbb{N}$,*

$$\Pr\left[\begin{array}{l|l} \mathsf{SNARK.Verify}(\tau, y, \pi) = 1 & (\sigma, \tau) \leftarrow \mathsf{SNARK.Gen}(1^\kappa, B) \\ (y, w) \notin \mathcal{R}_\mathcal{U} & (y, \pi) \leftarrow P^*(\mathsf{aux}, \sigma) \\ & w \leftarrow \mathcal{E}_{P^*}(\mathsf{aux}, \sigma) \end{array}\right] \leq \mathsf{negl}(\kappa).$$

3. Efficiency: *There exists a universal polynomial $p(\cdot)$ such that, for every large enough security parameter $\kappa \in \mathbb{N}$, every time bound $B \in \mathbb{N}$, and every instance $y = (M, x, t)$ with $t \leq B$,*

   - *The generator algorithm* $\mathsf{SNARK.Gen}(1^\kappa, B)$ *runs in time $p(k + B)$ for a fully succinct SNARK (and in time $p(k + \log B)$ for a preprocessing SNARK).*
   - *The prover algorithm* $\mathsf{SNARK.Prover}(\sigma, y, w)$ *runs in time $p(\kappa + |M| + t + \log B)$ for a fully succinct SNARK (and in time $p(k + |M| + |x| + B)$ for a preprocessing SNARK).*
   - *The verifier algorithm* $\mathsf{SNARK.Verify}(\tau, y, \pi)$ *runs in time $p(\kappa + |M| + |x| + \log B)$.*
   - *An honestly generated proof has size $p(\kappa + \log B)$.*

A.2 Proof-Carrying Data

A *proof-carrying data system* (PCD system) is a cryptographic primitive introduced by Chiesa and Tromer [32]. Informally speaking, given a predicate $\mathtt{C}$, consider a distributed system where nodes perform computations; each computation takes as input messages and generates a new output message. The security goal is to ensure that each output message is compliant with the predicate $\mathtt{C}$. Proof-carrying data ensures this goal by attaching short and easy-to-verify proofs of $\mathtt{C}$-compliance to each message.

Concretely, a generator $\mathsf{PCD.Gen}$ first sets up a reference string and a verification state. Anyone can then use the prover algorithm $\mathsf{PCD.Prover}$, which is given as input the reference string, prior messages z_{in} with proofs π_{in}, and an output message z_{out}, to generate a proof π_{out} attesting that z_{out} is $\mathtt{C}$-compliant. Anyone can use the verification algorithm $\mathsf{PCD.Verify}$, which is given as input the verification state, a message z, and a proof π, to verify that z is $\mathtt{C}$-compliant.

Crucially, the running time of proof generation and proof verification are “history independent”: the first only depends on the time to execute $\mathtt{C}$ on input a node’s messages, while the second only on the message length.

We now formally define the notions associated with a PCD system as defined in [9]. We refer the reader to [9, 8] for a detailed discussion.

Definition A.2. *A (distributed computation) transcript is a triplet* $\mathsf{trans} = (G, \mathsf{linp}, \mathsf{data})$, *where $G = (V, E)$ is a directed acyclic graph,* $\mathsf{linp} : V \to \{0,1\}^*$ *are local inputs (node labels), and* $\mathsf{data} : E \to \{0,1\}^*$ *are edge labels (messages sent on the edge). The output of* trans, *denoted* $\mathsf{out}(\mathsf{trans})$, *is equal to* $\mathsf{data}(\tilde{u}, \tilde{v})$ *where $(\tilde{u}, \tilde{v})$ is the lexicographically first edge such that $\tilde{v}$ is a sink.*

Syntactically a proof-carrying transcript is a transcript where messages are augmented by proof strings, i.e., a function $\mathsf{proof} : E \to \{0,1\}^*$ provides for each edge (u, v) an additional label prove (u, v), to be interpreted as a proof string for the message $\mathsf{data}(u, v)$

Definition A.3. *A proof-carrying (distributed computation) transcript PCT is a pair* $(\mathsf{trans}, \mathsf{proof})$ *where* trans *is a transcript and* $\mathsf{proof} : E \to \{0,1\}^*$ *is an edge label.*

Next, we define what it means for a distributed computation to be compliant, which as defined in [9] is the notion of "correctness with respect to a given local property." Compliance is captured via an efficiently computable compliance predicate $\mathtt{C}$, which must be locally satisfied at each vertex; here, "locally" means with respect to a node's local input, incoming data, and outgoing data. For convenience, for any vertex v, we let $\mathsf{child}(v)$ and $\mathsf{parent}(v)$ be the vector of v's children and parents respectively, listed in lexicographic order.

Definition A.4. *Given a polynomial-time predicate* $\mathtt{C}$*, we say that a distributed computation transcript* $\mathsf{trans} = (G, \mathsf{linp}, \mathsf{data})$ *is* $\mathtt{C}$*-compliant (denoted by* $\mathtt{C}(\mathsf{trans}) = 1$*) if for every* $v \in V$ *and* $w \in \mathsf{child}(v)$ *it holds that*

$$\mathtt{C}(\mathsf{data}(v,w); \mathsf{linp}(v), \mathsf{inputs}(v)) = 1,$$

where $\mathsf{inputs}(v) \coloneqq \mathsf{data}(u_1, v), \ldots, \mathsf{data}(u_c, v)$ *and* $(u_1, \ldots, u_c) \coloneqq \mathsf{parent}(v)$*. Furthermore, we say that a message* z *from node* v *to* w *is* $\mathtt{C}$*-compliant if* $\mathtt{C}(\mathsf{data}(v,w); \mathsf{linp}(v), \mathsf{inputs}(v)) = 1$ *and there is a transcript* trans *such that* v *is the sink and* $\mathtt{C}(\mathsf{trans}) = 1$*.*

Definition A.5. *Given a distributed computation transcript* $\mathsf{trans} = (G, \mathsf{linp}, \mathsf{data})$ *and any edge* $(v,w) \in E$*, we denote by* $t_{\mathsf{trans},\mathtt{C}}(v,w)$ *the time required to evaluate* $\mathtt{C}(\mathsf{data}(v,w); \mathsf{linp}(v), \mathsf{inputs}(v))$*. We say that* trans *is* B*-bounded if* $t_{\mathsf{trans},\mathtt{C}}(v,w) \leq B$ *for every edge* (v,w)*.*

Definition A.6. *The depth of a transcript* trans*, denoted* $d(\mathsf{trans})$*, is the largest number of nodes on a source-to-sink path in* trans *minus 2 (to exclude the source and the sink). The depth of a compliance predicate* $\mathtt{C}$*, denoted* $d(\mathtt{C})$*, is defined to be the maximum depth of any transcript* trans *compliant with* $\mathtt{C}$*. If* $d(\mathtt{C}) \coloneqq \infty$ *(i.e., paths in* $\mathtt{C}$*-compliant transcripts can be arbitrarily long) we say that* $\mathtt{C}$ *has unbounded depth.*

We note that for our application in Section 4, we can assume that for every $v \in V$, the label input is $\mathsf{linp}(v) = \perp$.

We now give a formal definition of a PCD system.

Definition A.7. *A proof-carrying data (PCD) system for a class of compliance predicates* C *is a triple of algorithms* $(\mathsf{PCD.Gen}, \mathsf{PCD.Prover}, \mathsf{PCD.Verify})$ *that work as follows:*

- $\mathsf{PCD.Gen}(1^\kappa, \mathtt{C}) \to (\sigma_{\mathsf{pcd}}, \tau_{\mathsf{pcd}})$*: on input the security parameter* κ *and compliance predicate* $\mathtt{C} \in C$*, the (probabilistic) generator* $\mathsf{PCD.Gen}$ *outputs a reference string* σ_{pcd} *and a corresponding verification state* τ_{pcd}*.*

- $\mathsf{PCD.Prover}(\tau_{\mathsf{pcd}}, z_{\mathit{in}}, \pi_{\mathit{in}}, \mathsf{linp}, z_{\mathit{out}}) \to \pi_{\mathit{out}}$*: given a reference string* τ_{pcd}*, inputs* z_{in} *with corresponding proofs* π_{in}*, a local input* linp*, and an output* z_{out}*, the (honest) prover algorithm* $\mathsf{PCD.Prover}$ *produces a proof* π_{out} *attesting to consistency of* z_{out} *with a* $\mathtt{C}$*-compliant transcript.*

- $\mathsf{PCD.Verify}(\tau_{\mathsf{pcd}}, z_{\mathit{out}}, \pi_{\mathit{out}}) \to b$*: given the verification state* τ_{pcd}*, an output* z_{out}*, and a proof string* π_{out}*, the verifier algorithm* $\mathsf{PCD.Verify}$ *accepts if it is convinced that* z_{out} *is consistent with some* $\mathtt{C}$*-compliant transcript.*

After the generator $\mathsf{PCD.Gen}$ is run to obtain σ_{pcd} and τ_{pcd}, the prover $\mathsf{PCD.Prover}$ is used (along with σ_{pcd}) at each node of a distributed computation transcript to dynamically compile it into a proof-carrying transcript by generating and adding a proof to each edge. Each of these proofs can be checked using the verifier $\mathsf{PCD.Verify}$ (along with τ_{pcd}). A PCD system $(\mathsf{PCD.Gen}, \mathsf{PCD.Prover}, \mathsf{PCD.Verify})$ must satisfy the following properties:

Completeness: An honest prover can convince a verifier that the output of any compliant transcript is indeed compliant. Namely, for every security parameter κ, compliance predicate $\mathtt{C}$, and distributed-computation generator G (described below),

$$\Pr\left[\begin{array}{l|l} \mathsf{trans} \text{ is } B\text{-bounded} & (\sigma_{\mathsf{pcd}}, \tau_{\mathsf{pcd}}) \leftarrow \mathsf{PCD.Gen}(1^{\kappa}, \mathtt{C}) \\ \mathtt{C}(\mathsf{trans}) = 1 & (z, \pi, \mathsf{trans}) \leftarrow \mathsf{Proof}_{\mathsf{Gen}}(\mathtt{C}, \sigma_{\mathsf{pcd}}, G, \mathsf{PCD.Prover}) \\ \mathsf{PCD.Verify}(\tau_{\mathsf{pcd}}, z, \pi) \neq 1 & \end{array}\right] \leq \mathsf{negl}(\kappa).$$

Above, $\mathsf{Proof}_{\mathsf{Gen}}$ is an interactive protocol between a distributed-computation generator $\mathsf{DC}_{\mathsf{Gen}}$ and the PCD prover $\mathsf{PCD.Prover}$, in which both are given the compliance predicate $\mathtt{C}$ and the reference string σ_{pcd}. Essentially, at every time step, $\mathsf{DC}_{\mathsf{Gen}}$ chooses to do one of the following actions: (1) add a new unlabeled vertex to the computation transcript so far (this corresponds to adding a new computing node to the computation), (2) label an unlabeled vertex (this corresponds to a choice of local data by a computing node), or (3) add a new labeled edge (this corresponds to a new message from one node to another). In case $\mathsf{DC}_{\mathsf{Gen}}$ chooses the third action, the PCD prover $\mathsf{PCD.Prover}$ produces a proof for the $\mathtt{C}$-compliance of the new message, and adds this new proof as an additional label to the new edge. When $\mathsf{DC}_{\mathsf{Gen}}$ halts, the interactive protocol outputs the distributed computation transcript trans, as well as trans's output and corresponding proof. Intuitively, the completeness property requires that if trans is compliant with $\mathtt{C}$, then the proof attached to the output (which is the result of dynamically invoking $\mathsf{PCD.Prover}$ for each message in trans, as trans was being constructed by $\mathsf{DC}_{\mathsf{Gen}}$) is accepted by the verifier.

Proof of knowledge (and soundness): Loosely speaking, if the verifier accepts a proof for a message, the prover "knows" a compliant transcript trans with output z. For every polynomial-size prover P^* there exists a polynomial-size extractor $\mathcal{E}_{P^*}$ such that for every polynomial-size compliance predicate $\mathtt{C} \in C$ and every auxiliary input $\mathsf{aux} \in \{0,1\}^{\mathsf{poly}(\kappa)}$,

$$\Pr\left[\begin{array}{l|l} \mathsf{PCD.Verify}(\tau_{\mathsf{pcd}}, z, \pi) = 1 & (\sigma_{\mathsf{pcd}}, \tau_{\mathsf{pcd}}) \leftarrow \mathsf{PCD.Gen}(1^{\kappa}, \mathtt{C}) \\ \mathsf{out}(\mathsf{trans}) \neq z \vee \mathtt{C}(\mathsf{trans}) \neq 1 & (z, \pi) \leftarrow P^*(\sigma_{\mathsf{pcd}}, \mathsf{aux}) \\ & \mathsf{trans} \leftarrow \mathcal{E}_{P^*}(\sigma_{\mathsf{pcd}}, \mathsf{aux}) \end{array}\right] \leq \mathsf{negl}(\kappa).$$

Succinctness: There exists a universal polynomial $p(\cdot)$ such that for every compliance predicate $\mathtt{C} \in C$, every time bound $B \in \mathbb{N}$, and every B-bounded distributed computation transcript trans,

- The computation time of $\mathsf{PCD.Prover}(\sigma_{\mathsf{pcd}}, z_{\mathsf{in}}, \pi_{\mathsf{in}}, \mathsf{linp}, z_{\mathsf{out}})$ is $p(\kappa + |\mathtt{C}| + B)$.
- The verification algorithm $\mathsf{PCD.Verify}(\tau_{\mathsf{pcd}}, z, \pi)$ runs in time $p(\kappa + |\mathtt{C}| + |z| + \log B)$
- An honestly generated proof has size $p(\kappa + \log B)$.

Theorem A.8 ([9]). *Let the size of a compliance predicate $\mathtt{C}$, denoted by $s(\mathtt{C})$, be the largest number of nodes in any transcript compliant with $\mathtt{C}$. Assuming the existence of SNARKs with linear extraction (i.e., $|\mathbb{E}_{\mathcal{P}^*}| \leq c|\mathcal{P}^*|$ for some constant c), there exist PCD systems for logarithmic-depth and polynomial-size compliance predicates.*

A.3 Merkle Hash Proof System

A Merkle hash proof system [80] corresponding to a hash function $H : \{0,1\}^\kappa \times \{0,1\}^\lambda \to \{0,1\}^{\lambda/2}$ is defined by a tuple of algorithms $(\mathsf{Merkle.Setup}, \mathsf{Merkle.Hash}, \mathsf{Merkle.Proof}, \mathsf{Merkle.Verify})$ as follows:

- $\mathsf{Merkle.Setup}(1^\kappa)$: On input the security parameter, the setup algorithm samples and outputs a random $\mathsf{seed} \leftarrow \{0,1\}^\kappa$ for the hash function.

- $\mathsf{Merkle.Hash}(\mathsf{seed}, x_1, \ldots, x_n)$: On input the seed and a vector $x_1, \ldots, x_n$, the Merkle hash algorithm computes a hash using a Merkle tree as follows:

  - For each $i \in [n]$, compute $y_i^0 = H(\mathsf{seed}, x_i)$.
  - For each $\ell \in [\log(n)]$ and $i \in [n/2^\ell]$,[25] compute $y_i^\ell = H(\mathsf{seed}, y_{2i-1}^{\ell-1} || y_{2i}^{\ell-1})$.

  Output $y = y_1^{\log(n)}$.

- $\mathsf{Merkle.Proof}(\mathsf{seed}, x_1, \ldots, x_n, x_i)$: On input the seed, a vector $x_1, \ldots, x_n$, and an element x_i, the Merkle proof algorithm computes and outputs a proof p as follows:

  - For each $k \in [n]$, compute $y_k^0 = H(\mathsf{seed}, x_k)$.
  - For each $\ell \in [\log(n)]$ and $k \in [n/2^\ell]$ compute $y_k^\ell = H(\mathsf{seed}, y_{2k-1}^{\ell-1} || y_{2k}^{\ell-1})$.
  - Initialize the proof $p = \{(i, \mathsf{sibling}(y_i^0))\}$ and for each level $\ell \in [\log(n)]$, set $p = p \cup \{(\lceil i/2^\ell \rceil, \mathsf{sibling}(y_{\lceil i/2^\ell \rceil}^\ell))\}$.

- $\mathsf{Merkle.Verify}(\mathsf{seed}, x_i, y, p)$: On input the seed, an input element x_i, Merkle hash y, and a Merkle proof p, the Merkle verification algorithm parses $p = ((i_0, x^0), \ldots, (i_{\log(n)}, x^{\log(n)}))$ and proceed as follows:

  - If i_0 is an even number, compute $y^1 = H(\mathsf{seed}, H(\mathsf{seed}, x_i) || x^0)$, else compute $y^1 = H(\mathsf{seed}, x^0 || H(\mathsf{seed}, x_i))$.
  - For each $\ell \in [\log(n)]$, if i_ℓ is an even number, compute $y^\ell = H(\mathsf{seed}, H(\mathsf{seed}, y^{\ell-1}) || x^\ell)$, else compute $y^\ell = H(\mathsf{seed}, x^\ell || H(\mathsf{seed}, y^{\ell-1}))$.

  If $y^{\log(n)} = y$, output 1; else, output 0.

The Merkle Hash Proof System has the following properties.

Theorem A.9 (Merkle hash proof system)**.** *Assuming existence of a length-halving, seeded, collision resistant hash function $H : \{0,1\}^\kappa \times \{0,1\}^\lambda \to \{0,1\}^{\lambda/2}$, the Merkle hash proof system* $(\mathsf{Merkle.Setup}, \mathsf{Merkle.Hash}, \mathsf{Merkle.Proof}, \mathsf{Merkle.Verify})$ *satisfies the following properties:*

- ***Completeness:*** *For any input string $x_1, \ldots, x_n \in \{0,1\}^{n\lambda}$ and $i \in [n]$, it holds that:*

$$\Pr\left[\mathsf{Merkle.Verify}(\mathsf{seed}, x_i, y, p) = 1 \;\middle|\; \begin{array}{l} \mathsf{seed} \leftarrow \mathsf{Merkle.Setup}(1^\kappa) \\ y = \mathsf{Merkle.Hash}(\mathsf{seed}, x_1, \ldots, x_n) \\ p = \mathsf{Merkle.Proof}(\mathsf{seed}, x_1, \ldots, x_n, x_i) \end{array} \right] = 1.$$

[25] For simplicity, we assume that n is a power of 2. The general case follows by including additional elements 0^λ, such that the length of the resulting input string becomes a power of 2.

- ***Soundness:*** *No PPT adversary $\mathcal{A}$, can win the following game with more than negligible probability (in κ):*
    1. *The challenger samples* $\mathsf{seed} \leftarrow \mathsf{Merkle.Setup}(1^\kappa)$ *and sends to* $\mathcal{A}$*.*
    2. $\mathcal{A}$ *responds with* $(i, \{x_j\}_{j\in[n]\setminus\{i\}})$*.*
    3. *The challenger samples* $x_i \leftarrow \{0,1\}^\lambda$*, computes* $\mathsf{Merkle.Hash}(\mathsf{seed}, x_1, \ldots, x_n) = y$ *and sends* (x_i, y) *to* $\mathcal{A}$*.*
    4. $\mathcal{A}$ *responds with a pair* (x', p)*, and wins if* $\mathsf{Merkle.Verify}(\mathsf{seed}, x', y, p) = 1$ *and* $x' \neq x_i$ *for every* $i \in [n]$*.*

A.4 Multi-signatures

In a multi-signature scheme, a single short object—the *multi-signature*—can take the place of n signatures by n signers, all on the same message.[26] The first formal treatment of multi-signatures was given by Micali, Ohta, and Reyzin [83]. We consider a variant of this model due to Boldyreva [12] that is also used by Lu et al. [77]. In this model, the adversary is given a single challenge verification key vk, and a signing oracle for that key. His goal is to output a forged multi-signature σ^* on a message m^* under keys $\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell$, where at least one of these keys is a challenge verification key (without loss of generality, vk_1). For the forgery to be nontrivial, the adversary must not have queried the signing oracle at m^*. The adversary is allowed to choose the remaining keys, but must prove knowledge of the private keys corresponding to them.

Definition A.10. *A* multi-signature *scheme is a tuple of algorithms*

- $\mathsf{MS.Setup}(1^\kappa) \rightarrow \mathsf{pp}$*: On input the security parameter, the setup algorithm outputs public parameters* pp*.*

- $\mathsf{MS.KeyGen}(\mathsf{pp}) \rightarrow (\mathsf{vk}, \mathsf{sk})$*: On input the public parameters* pp*, the key-generation algorithm outputs a pair of verification/signing keys* $(\mathsf{vk}, \mathsf{sk})$*.*

- $\mathsf{MS.Sign}(\mathsf{pp}, \mathsf{sk}, m) \rightarrow \sigma$*: On input* pp*, a signing key* sk*, and a message m, the signing algorithm outputs a signature σ.*

- $\mathsf{MS.Verify}(\mathsf{pp}, \mathsf{vk}, \sigma, m) \rightarrow b$*: On input* pp*, a verification key* vk*, a signature σ, and a message m, the verification algorithm outputs a bit $b \in \{0,1\}$.*

- $\mathsf{MS.Combine}(\mathsf{pp}, \{\mathsf{vk}_i, \sigma_i\}_{i=1}^{\ell}, m) \rightarrow \sigma$*: On input* pp*, a collection of signatures (or multi-signatures), and a message m, the combine algorithm outputs a combined multi-signature σ, with respect to the union of verification keys.*

- $\mathsf{MS.MVerify}(\mathsf{pp}, \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}, S, m, \sigma) \rightarrow b$*: On input* pp*, the set of all verification keys, a subset $\S \subseteq [n]$, a message m, and a multi-signature σ, the multi-signature verification algorithm outputs a bit $b \in \{0,1\}$.*

We require the following properties from a multi-signature scheme.

[26]Note that multi-signatures are a special case of *aggregate* signatures [14], which in contrast allow combining signatures from n different parties on n *different* messages.

Correctness: The correctness requirement of digital signatures must hold for ($\mathsf{MS.Setup}$, $\mathsf{MS.KeyGen}$, $\mathsf{MS.Sign}$, $\mathsf{MS.Verify}$). In addition, for any message m, any collection of honestly generated signatures $\{\sigma_i \leftarrow \mathsf{MS.Sign}(\mathsf{pp}, \mathsf{sk}_i, m)\}_{i \in S}$ on m (for some $S \subseteq [n]$), the combined multi-signature formed by $\bar{\sigma} \leftarrow \mathsf{MS.Combine}(\mathsf{pp}, \{\mathsf{vk}_i, \sigma_i\}_{i \in S}, m)$ will properly verify with overwhelming probability, i.e., $\Pr[1 \leftarrow \mathsf{MS.MVerify}(\mathsf{pp}, \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}, S, m, \bar{\sigma})] \geq 1 - \mathsf{negl}(k)$.

Unforgeability: For any PPT adversary $\mathcal{A}$, the probability that the challenger outputs 1 when interacting with $\mathcal{A}$ in the following game is negligible in the security parameter κ:

1. *Setup.* $\mathcal{A}$ selects a proper subset $\mathcal{I} \subseteq [n]$ (corresponding to corrupted parties). The challenger samples a pair of verification/signing keys $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{MS.KeyGen}(\mathsf{pp})$ for every $i \in [n] \backslash \mathcal{I}$, and gives $\mathcal{A}$ all verification keys $\{\mathsf{vk}_i\}_{i \in [n] \backslash \mathcal{I}}$. Next, $\mathcal{A}$ chooses keys $\{\mathsf{sk}_i, \mathsf{vk}_i\}_{i \in \mathcal{I}}$ for the corrupted parties and sends them to the challenger.

2. *Signing queries.* $\mathcal{A}$ can make polynomially many adaptive signature queries of the form (m, vk_i). For each query, the challenger responds with a signature $\sigma \leftarrow \mathsf{MS.Sign}(\mathsf{pp}, \mathsf{sk}_i, m)$ on the message m with respect to the signing key sk_i corresponding to vk_i.

3. *Output.* $\mathcal{A}$ outputs a triple $(\bar{\sigma}^*, m^*, \{\mathsf{vk}_i\}_{i \in S})$. The challenger outputs 1 if at least one of the provided verification keys vk_i corresponds to a challenge (honest party) key, the message m^* was not queried to the signature oracle with this verification key vk_i, and the provided forgery σ^* is a valid multi-signature, i.e., $1 \leftarrow \mathsf{MS.MVerify}(\mathsf{pp}, \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}, S, m^*, \sigma^*)$.

A.5 The Multi-Signatures Scheme of Lu et al. [77]

In this section we describe the LOSSW multi-signature scheme that is used in Section 6. We will let $\mathbb{G}$ and $\mathbb{G}_T$ are multiplicative groups of prime order p, and denote g a generator of $\mathbb{G}$. In addition, let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be an efficiently computable non-degenerate bilinear map. The multi-signature scheme of Lu et al. [77] is based on the Bilinear Computational Diffie-Hellman (BCDH) assumption. The message space is $\{0,1\}^k$ for some fixed k. The following is taken verbatim from [77]:

- $\mathsf{MS.Setup}(1^\kappa)$: Sample random elements $u', u_1, \ldots, u_k \in \mathbb{G}$ and output the public parameters $\mathsf{pp}_{\mathsf{ms}}$, consisting of descriptions of $\mathbb{G}, \mathbb{G}_T, p, e$, $u', u_1, \ldots, u_k$ and the generator g of $\mathbb{G}$.

- $\mathsf{MS.KeyGen}(\mathsf{pp}_{\mathsf{ms}})$: Sample a random signing key $\mathsf{sk} \in \mathbb{Z}_p$ and set the corresponding verification key vk as $e(g,g)^{\mathsf{sk}}$.

- $\mathsf{MS.Sign}(\mathsf{pp}_{\mathsf{ms}}, \mathsf{sk}, m)$: Parse the message m as $(m_1, \ldots, m_k) \in \{0,1\}^k$, sample $r \leftarrow \mathbb{Z}_p$, and compute $\sigma = (\mathsf{sig}_1, \mathsf{sig}_2)$ as follows:

$$\mathsf{sig}_1 = g^{\mathsf{sk}} \cdot \left(u' \cdot \prod_{i=1}^{k} u_i^{m_i}\right)^r \text{ and } \mathsf{sig}_2 = g^r.$$

- $\mathsf{MS.Verify}(\mathsf{pp}_{\mathsf{ms}}, \mathsf{vk}, m, \sigma_{\mathsf{ms}})$: Parse the message m as $(m_1, \ldots, m_k) \in \{0,1\}^k$ and $\sigma_{\mathsf{ms}} = (\mathsf{sig}_1, \mathsf{sig}_2)$, and outputs 1 if and only if

$$e(\mathsf{sig}_1, g) \cdot e\left(\mathsf{sig}_2, u' \cdot \prod_{i=1}^{k} u_i^{m_i}\right)^{-1} = \mathsf{vk}.$$

- $\mathsf{MS.Combine}(\mathsf{pp}_{\mathsf{ms}}, \{\mathsf{vk}_i, \sigma_i\}_{i \in S}, m)$: Parse each σ_i as $(\mathsf{sig}_1^{(i)}, \mathsf{sig}_2^{(i)})$ and compute the combined multi-signature $\sigma_{\mathsf{ms}} = (\mathsf{sig}_1, \mathsf{sig}_2)$ as follows:

$$\mathsf{sig}_1 = \prod_{i \in S} \mathsf{sig}_1^{(i)} \text{ and } \mathsf{sig}_2 = \prod_{i \in S} \mathsf{sig}_2^{(i)}.$$

- $\mathsf{MS.MVerify}(\mathsf{pp}_{\mathsf{ms}}, \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}, S, m, \sigma_{\mathsf{ms}})$: Output 1 if and only if

$$e\left(\mathsf{sig}_1, g\right) \cdot e\left(\mathsf{sig}_2, u' \cdot \prod_{i=1}^{k} u_i^{m_i}\right)^{-1} = \prod_{i \in S} \mathsf{vk}_i.$$

B Balanced Communication-Efficient BA (Cont'd)

In this section, we provide supplementary material for Section 4.

B.1 Balanced Byzantine Agreement from SRDS (Cont'd)

In this section, we give the proof of Lemma 4.5 and discuss applications of our Byzantine agreement protocol.

Proof. Let $\mathcal{A}$ be a PPT adversary for π_{ba}. We construct a simulator $\mathcal{S}$ as follows. The simulator $\mathcal{S}$ starts by simulating the setup for the protocol, while allowing adaptive corruptions by $\mathcal{A}$ (in a similar way to the robustness and unforgeability games). First, $\mathcal{S}$ runs the setup algorithm as $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa, 1^{n \cdot z})$, and for every $i \in [n]$ and $j \in [z]$ computes $(\mathsf{vk}_{i,j}, \mathsf{sk}_{i,j}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$. Next, $\mathcal{S}$ sends $(1^\kappa, 1^{n \cdot z}, \mathsf{pp}, \{\mathsf{vk}_{i,j}\}_{i \in [n], j \in [z]})$ to $\mathcal{A}$. As long as $|\mathcal{I}| \leq \beta \cdot n$ and $\mathcal{A}$ requests to corrupt a party P_i, the simulator sends $\{\mathsf{sk}_{i,j}\}_{j \in [z]}$ to $\mathcal{A}$ and receives back $\{\mathsf{vk}'_{i,j}\}_{j \in [z]}$; in the bare-PKI mode, $\mathcal{S}$ updates each $\mathsf{vk}_{i,j} = \mathsf{vk}'_{i,j}$. Let $\{\mathsf{vk}_{i,j}\}_{i \in [n], j \in [z]}$ be the PKI keys at the end of this process.

The simulator $\mathcal{S}$ proceeds to simulate the protocol execution towards $\mathcal{A}$. Initially, $\mathcal{S}$ receives from f_{ba} the input bits of all honest parties $\{x_i\}_{i \notin \mathcal{I}}$. To simulate $f_{\mathsf{ae\text{-}comm}}$ in Step 1, the simulator receives from $\mathcal{A}$ the communication-tree T defining the set of isolated parties $\mathcal{D}$. The simulator simulates sending the output to every corrupted party. Let $\mathcal{C}$ denote the supreme committee (the parties assigned to the root).

To simulate f_{ba} for the supreme committee in Step 2a, $\mathcal{S}$ sends to $\mathcal{A}$ the input bit x_i for every $i \in \mathcal{C} \setminus \mathcal{I}$ and receives inputs $\{x_i\}_{i \in \mathcal{I} \cap \mathcal{C}}$. If 2/3 of the honest committee members' bits are the same, denote this value by y; otherwise, let $\mathcal{A}$ determine y. Output the value y to every corrupted party in $\mathcal{C}$. To simulate f_{ct} in Step 2b, sample a random $s \in \{0,1\}^\kappa$ and send s to $\mathcal{A}$ for every P_i for $i \in \mathcal{I} \cap \mathcal{C}$.

To simulate the call to $f_{\mathsf{ae\text{-}comm}}$ in Step 3, receive inputs from $\mathcal{A}$ on behalf of corrupted supreme-committee members, and send (y, s) to $\mathcal{A}$ for every $i \in \mathcal{I}$. In addition, receive (y_i, s_i) for every $i \in \mathcal{D}$ from $\mathcal{A}$.

Next, for every honest party P_i for $i \notin \mathcal{I}$ do the following:

- For $i \notin \mathcal{D}$, compute $\sigma_{i,j} \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathsf{idmap}(i,j), \mathsf{sk}_{i,j}, (y, s))$ for each $j \in [z]$.
- For $i \in \mathcal{D}$, compute $\sigma_{i,j} \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathsf{idmap}(i,j), \mathsf{sk}_{i,j}, (y_i, s_i))$ for each $j \in [z]$.

To simulate Step 4, for every $i \in [n] \setminus \mathcal{I}$, let $L_i = \{v_{i_1}, \ldots, v_{i_z}\} \subseteq V$ be the subset of leaves assigned to P_i. For each $j \in [z]$, send $\sigma_{i,j}$ to all corrupted parties assigned to the leaf node v_{i_j} on behalf of P_i. In addition, for every P_i assigned to a leaf node v, receive a signature $\sigma_{j,k}$ from every corrupt P_j for which $v = v_{j_k} \in L_j$.

To simulate Step 5, for each level $\ell = 1, \ldots, \ell^*$ of the tree and each node v on level ℓ:

1. For each $i \in \mathsf{party}(v) \setminus \mathcal{I}$, prepare the set of signatures received in Step 5a as follows:
   - For $\ell = 1$: let $S_{\mathsf{sig}}^{\ell,i,1}$ be the set of following signatures. For every honest P_j with $v = v_{j_k} \in L_j$, the signature $\sigma_{j,k}$ simulated in the previous step. For every corrupt P_j with $v = v_{j_k} \in L_j$, the signature $\sigma_{j,k}^i$ received from the adversary (note that the adversary might send different signatures to different parties).
   - For $\ell > 1$: let $S_{\mathsf{sig}}^{\ell,i,1}$ be the set of following signatures. For each child node $u \in \mathsf{child}(v)$ and each $j \in \mathsf{party}(u) \setminus \mathcal{I}$, the signature σ_u (that was simulated for level $\ell - 1$). For each $j \in \mathsf{party}(u) \cap \mathcal{I}$, the signature σ_u^i received from the adversary $\mathcal{A}$ (note that the adversary might send different signatures to different parties).
2. Next, simulate $|\mathsf{party}(v)|$ broadcast protocols in Step 5b, where for every $i \in \mathsf{party}(v)$, party P_i broadcasts $S_{\mathsf{sig}}^{\ell,i,1}$. Let $S_{\mathsf{sig}}^{\ell,i,2}$ be the union of the sets of the broadcasted signatures.
3. To simulate Step 5c, for each party assigned to the node, i.e., for each $i \in \mathsf{party}(v) \setminus \mathcal{I}$, compute $S_{\mathsf{sig}}^{\ell,i,3} \leftarrow \mathsf{Aggregate}_1(\mathsf{pp}, \{\mathsf{vk}_{1,1}, \ldots, \mathsf{vk}_{n,z}\}, (y, s), S_{\mathsf{sig}}^{\ell,i,2})$. If $\ell = 1$, for each sig in $S_{\mathsf{sig}}^{i,\ell,3}$ check if $\mathsf{min}(\mathsf{sig}) = \mathsf{max}(\mathsf{sig})$ and if $\mathsf{min}(\mathsf{sig}) \in \mathsf{range}(v)$, whereas if $\ell > 1$ check if $\exists v' \in \mathsf{child}(v)$ such that the range $[\mathsf{min}(\mathsf{sig}), \mathsf{max}(\mathsf{sig})]$ falls within the range $\mathsf{range}(v')$. If this check fails for any sig, it updates $S_{\mathsf{sig}}^{i,\ell,3} = S_{\mathsf{sig}}^{i,\ell,3} \setminus \{\mathsf{sig}\}$. To simulate $f_{\mathsf{aggr\text{-}sig}}$, for every $i \in \mathsf{party}(v) \cap \mathcal{I}$, receive from $\mathcal{A}$ a message $((\tilde{y}_i, \tilde{s}_i), \tilde{S}_{\mathsf{sig}}^{\ell,i,3})$. If $|\mathsf{party}(v) \setminus \{\mathcal{I} \cup \mathcal{D}\}| \geq 2|\mathsf{party}(v)|/3$ (i.e., the node is good), compute
$$\sigma_v \leftarrow \mathsf{Aggregate}_2\Big(\mathsf{pp}, (y, s), S_{\mathsf{sig}}^{\ell,i,3}\Big).$$
Else (i.e., the node is bad), get σ_v from $\mathcal{A}$. Finally, send σ_v to $\mathcal{A}$ as the output of $f_{\mathsf{aggr\text{-}sig}}$.
4. If $\ell < \ell^*$, send for every σ_v from each honest party in $\mathsf{party}(v)$ to every corrupt party in $\mathsf{party}(u)$, where $u = \mathsf{parent}(v)$. In addition, receive from $\mathcal{A}$ a signature σ_v' from every corrupt party in $\mathsf{party}(v)$ to every honest party in $\mathsf{party}(u)$.

To simulate the call to $f_{\mathsf{ae\text{-}comm}}$ in Step 6, receive inputs from $\mathcal{A}$ on behalf of corrupted supreme-committee members, and send $(y, s, \sigma_{\mathsf{root}})$ to $\mathcal{A}$ for every $i \in \mathcal{I}$. In addition, receive (y_j', s_j', σ_j') for $j \in \mathcal{D}$ from $\mathcal{A}$. Finally, to simulate Step 7, for every $i \notin \mathcal{I} \cup \mathcal{D}$ evaluate $\mathcal{C}_i = F_s(i)$ and simulate party P_i sending $(y, s, \sigma_{\mathsf{root}})$ to every party P_j for $j \in \mathcal{I} \cap \mathcal{C}_i$. For every $i \in \mathcal{D}$ evaluate $\mathcal{C}_i = F_{s_j'}(i)$ and simulate party P_i sending (y_j', s_j', σ_j') to every party P_j for $j \in \mathcal{I} \cap \mathcal{C}_i$.

To conclude the simulation, the simulator sends the value y to the ideal functionality f_{ba} as the "tie-breaker" value and outputs whatever $\mathcal{A}$ outputs.

Note that $\mathcal{S}$ simulates a random honest execution towards the adversary, with only the syntactic difference that $\mathcal{S}$ simulates the ideal functionalities computing $f_{\mathsf{ae\text{-}comm}}$, f_{ba}, f_{ct} and $f_{\mathsf{aggr\text{-}sig}}$ (rather than using trusted parties). Thus, the view of the adversary is perfectly distributed in the real and ideal worlds. What remains to prove is that conditioned on the view of the adversary, the output of the honest parties is correct and identical in the real and ideal worlds. In other words, we need to show that this Byzantine agreement protocol satisfies both agreement and validity.

Claim B.1 (Agreement)**.** *For any adversarial strategy of $\mathcal{A}$, all honest parties output the same value, except for negligible probability.*

We show that our protocol satisfies agreement in three main steps; (1) We start by showing that with an overwhelming probability, every isolated party receives a message from at least one non-isolated honest party in the last round. (2) Next, we show that the aggregate signature σ_{root} obtained by the end of Step 5 is a valid SRDS on (y, s), where y and s are the outputs of f_{ba} and f_{ct} in Step 2b, respectively. We prove this by showing a reduction to the robustness property of the SRDS scheme. Thereby showing that every honest party receives a valid SRDS on the same message (y, s). (3) Finally we prove that every honest party only receives one valid SRDS (which is on (y, s)). We prove this by showing that an adversary cannot forge a valid SRDS on any other message by relying on the unforgeability of the SRDS scheme. Thus, each honest party outputs the same value y. Now we proceed to the formal proof.

Proof of Claim B.1. Let F_s be a truly random function. Then the set $\mathcal{C}_i$ defined by $F_s(i)$ is chosen randomly for each $i \in [n]$. Therefore, in expectation, each party P_j appears in $\mathsf{polylog}(n)$ sets. From Chernoff bound, except with some negligible probability (in n), each party receives messages from $\mathsf{polylog}(n) \pm \delta$ for $\delta = O(1)$ other parties. Similarly, except with negligible probability, each party receives messages from at least one non-isolated honest party. Therefore, each isolated party P_i for $i \in \mathcal{D}$ in the initial phase of the protocol, receives a message from at least one non-isolated honest party $P_j \in [n] \setminus \{\mathcal{I} \cup \mathcal{D}\}$. If this is true for a truly random function, the same must also hold for a pseudorandom F_s with overwhelming probability (in κ) over a random seed s. Recall that the message sent by P_j to P_i is $(y, s, \sigma_{\mathsf{root}})$ (where y is the output of f_{ba} in Step 2a). It remains to show the following:

1. Except with some negligible probability, σ_{root} is a valid SRDS on (y, s).
2. Except with some negligible probability, no adversary can compute a valid SRDS on any message other than (y, s).

1. Receiving valid signatures on (y, s)**.** Let us assume for the sake of contradiction that σ_{root} is *not* a valid SRDS on (y, s). We now construct an adversary $\mathcal{B}$ that can break *robustness* of the SRDS scheme. The adversary $\mathcal{B}$ interacts with the challenger of the SRDS scheme and the adversary $\mathcal{A}$ and proceeds as follows:

- $\mathcal{B}$ maps each corrupt virtual party to a party in the SRDS robustness game, i.e., elements in the set $[n \cdot z]$. In other words, the challenger of the SRDS scheme runs the setup algorithm as $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa, 1^{n \cdot z})$, and for every $i \in [n]$ and $j \in [z]$ computes $(\mathsf{vk}_{i,j}, \mathsf{sk}_{i,j}) \leftarrow \mathsf{KeyGen}(1^\kappa)$. Next, it sends $(1^\kappa, 1^{n \cdot z}, \mathsf{pp}, \{\mathsf{vk}_{i,j}\}_{i \in [n], j \in [z]})$ to $\mathcal{B}$, which it forwards to $\mathcal{A}$. For each $i \in \mathcal{I}$ that $\mathcal{A}$ requests to corrupt, $\mathcal{B}$ chooses to corrupt the corresponding parties $\{(i, j)\}_{j \in [z]}$ in the SRDS robustness game and receives $\{\mathsf{sk}_{i,j}\}_{j \in [z]}$ from the challenger, which it forwards to $\mathcal{A}$. Next, $\mathcal{B}$ receives verification keys $\{\mathsf{vk}'_{i,j}\}_{i \in \mathcal{I}, j \in [z]}$ of the corrupted parties from $\mathcal{A}$.
- For the bare-PKI mode, $\mathcal{B}$ updates $\mathsf{vk}_{i,j} = \mathsf{vk}'_{i,j}$ for each $i \in \mathcal{I}$ and $j \in [z]$.
- $\mathcal{B}$ then simulates step 1, as described in the simulator to receive the $(n, \mathcal{I})$-$\mathsf{almost\text{-}everywhere\text{-}communication\ tree\ with\ repeated\ parties}$ T from $\mathcal{A}$. It transforms this tree into an $(n \cdot$

$z, \{(i,j)\}_{i\in\mathcal{I}, j\in[z]}$)-almost-everywhere-communication tree by augmenting it with level 0 comprising of $n \cdot z$ nodes (representing the $n \cdot z$ virtual parties in the SRDS game), and adding an edge between each of these nodes and the leaf node that it (i.e., the party that they represent) is assigned to. It forwards this transformed tree to the challenger of the SRDS game.

- $\mathcal{B}$ then proceeds to simulate steps 2a, 2b, and 3 as described in the simulator and learns (y, s) and (y_i, s_i) for each $i \in \mathcal{D}$. $\mathcal{B}$ sets $m = (y, s)$, $m_{(i,j)} = (y_i, s_i)$ for each $i \in \mathcal{D}$, $j \in [z]$ and for each $(i, j) \in \mathcal{N} \setminus \{(i,j)\}_{i\in\mathcal{D}, j\in[z]}$, where $\mathcal{N}$ is the set of all honest parties in the SRDS game that are assigned to leaf nodes that do not have a good path in the transformed tree (described in the previous step), it sets $m_{(i,j)} = (y, s)$. It sends these messages to the challenger of the SRDS game.

- $\mathcal{B}$ receives signatures $\{\sigma_{i,j}\}_{i\in[n]\setminus\mathcal{I}, j\in[z]}$ of the honest parties from the challenger and forwards them to the adversary $\mathcal{A}$.

- For each level $\ell = 1, \ldots, \ell^*$ of the communication tree and each node v on level ℓ, it simulates Step 5 as described in the simulator, except in Step 5c, if the node is good, it sets σ_v to the partially aggregated signature sent by the challenger and if the node is bad, it forwards the partially aggregated signature σ_v received from $\mathcal{A}$ to the challenger of the SRDS game.

Note that if for some adversarial strategy $\mathcal{A}$, the signature σ_{root} is not a valid SRDS on (y, s), then by construction, $\mathcal{B}$ wins the robustness game of the SRDS scheme. From robustness of the SRDS scheme, we know that this only happens with at most negligible probability, therefore our assumption is incorrect and with overwhelming probability, σ_{root} is a valid SRDS on (y, s).

2. Not receiving valid signatures on other values. We now show that if the adversary $\mathcal{A}$ can forge an SRDS on any other message, then we can use this adversary to construct another adversary $\mathcal{B}$ that can break unforgeability of the SRDS scheme. The adversary $\mathcal{B}$ proceeds as follows:

- $\mathcal{B}$ maps each corrupt virtual party to a party in the SRDS game, i.e., elements in the set $[n \cdot z]$. In other words, the challenger of the SRDS scheme runs the setup algorithm as $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa, 1^{n\cdot z})$, and for every $i \in [n]$ and $j \in [z]$ computes $(\mathsf{vk}_{i,j}, \mathsf{sk}_{i,j}) \leftarrow \mathsf{KeyGen}(1^\kappa)$. Next, it sends $(1^\kappa, 1^{n\cdot z}, \mathsf{pp}, \{\mathsf{vk}_{i,j}\}_{i\in[n], j\in[z]})$ to $\mathcal{B}$, which it forwards to $\mathcal{A}$. For each $i \in \mathcal{I}$ that $\mathcal{A}$ requests to corrupt, $\mathcal{B}$ chooses to corrupt the corresponding parties $\{(i,j)\}_{j\in[z]}$ in the SRDS game and receives $\{\mathsf{sk}_{i,j}\}_{j\in[z]}$ from the challenger, which it forwards to $\mathcal{A}$. Next, $\mathcal{B}$ receives verification keys $\{\mathsf{vk}'_{i,j}\}_{i\in\mathcal{I}, j\in[z]}$ of the corrupted parties from $\mathcal{A}$. In the bare-PKI mode, $\mathcal{B}$ updates $\mathsf{vk}_{i,j} = \mathsf{vk}'_{i,j}$ for each $i \in \mathcal{I}, j \in [z]$.

- $\mathcal{B}$ then proceeds to simulate Steps 1, 2a, 2b, and 3 as described in the simulator. $\mathcal{B}$ chooses $m = (y, s)$ and $\mathcal{S} = \mathcal{D}$ and sends it to the challenger. For each $i \in \mathcal{D}$ and $j \in [z]$, it sets $m_{i,j} = (y_i, s_i)$ as received from the adversary.

- $\mathcal{B}$ receives signatures $\{\sigma_{i,j}\}_{i\in[n]\setminus\mathcal{I}, j\in[z]}$ of the honest parties from the challenger and forwards them to the adversary $\mathcal{A}$.

- $\mathcal{B}$ then simulates Steps 4, 5, 6, 7, and 8 as described in the simulator.

- Finally if $\mathcal{A}$ manages to send a valid SRDS on a message other than (y, s) to any of the honest parties, $\mathcal{B}$ forwards that to the challenger.

Clearly, $\mathcal{B}$ wins the forgery game only if $\mathcal{A}$ succeeds in forging a valid SRDS on a message other than (y, s). Since our SRDS scheme is unforgeable, this only happens with negligible probability. □

Claim B.2 (Validity)**.** *For any adversarial strategy of $\mathcal{A}$, if there exists a value x such that $x_i = x$ for each honest party $P_i \in [n] \setminus \mathcal{I}$, then the output of all honest parties is $y = x$.*

Proof. From Claim B.1, we know that with overwhelming probability, the final output y of all honest parties is the same as the output of f_{ba} in Step 2a. All that remains to prove now is that if there exists a value x, such that $x_i = x$ for each honest party $P_i \in [n] \setminus \mathcal{I}$, then the output of f_{ba} in Step 2a is x. Recall that f_{ba} in Step 2a is computed over the inputs of all parties in the supreme committee $\mathcal{C}$. From Definition 4.4, we know that at least 2/3 fraction of the parties in $\mathcal{C}$ are honest. Therefore, if there exists a value x such that x is the input of all honest parties, then the input of all honest parties in $\mathcal{C}$ is also x. Now, irrespective of the inputs of the remaining malicious parties in $\mathcal{C}$, from the validity of f_{ba}, we are guaranteed that the output of f_{ba} is $y = x$. □

This concludes the proof of Lemma 4.5. □

C Constructions of SRDS (Cont'd)

In this section we present the proofs on the SRDS constructions from Section 5.

C.1 SRDS from One-Way Functions (Cont'd)

We now present the proof of Theorem 5.1.

Theorem 5.1. *Let $\beta < 1/3$ be a constant. Assuming the existence of one-way functions, there exists a βn-secure SRDS scheme in the trusted PKI model.*

Proof of Theorem 5.1. In Lemma C.1, we prove succinctness, in Lemma C.2, we prove robustness, and in Lemma C.5, we prove unforgeability.

Lemma C.1. *The construction in Figure 7 is succinct.*

Proof. We start by proving the size of the signatures is succinct. Let $\mathcal{C} = \{i \mid \mathsf{sk}_i \neq \perp\}$ and let X be a random variable representing $|\mathcal{C}|$. By construction, $\mathbb{E}[X] = \ell$ and $\ell = \omega(\log(n))$. Therefore, by Chernoff bound for $\mu = \ell$ and $\delta = 1/2$,[27] it holds that

$$\Pr\left[|X - \ell| \geq \ell/2\right] \leq 2e^{-\ell/12} = \mathsf{negl}(n).$$

We therefore conclude that $\ell/2 \leq |\mathcal{C}| \leq 3\ell/2$ with overwhelming probability (in n). By definition of digital signatures, every signature in the support of $\mathsf{DS.Sign}$ is polynomial in κ. Therefore, every σ in the support of Sign (of the SRDS scheme) is also polynomial in κ. By construction, unless an adversary is able to successfully break the obliviousness of the signature scheme (which only happens with negligible probability in κ), an aggregate signature only consists of $|\mathcal{C}|$ "base" signatures from the parties in $\mathcal{C}$. Further, in the negligible event where the aggregate signature

[27]The exact Chernoff bound used is $\Pr\left[|X - \mu| \geq \delta\mu\right] \leq 2e^{-\mu\delta^2/3}$ for $0 < \delta < 1$, where $\mu = \mathbb{E}[X]$.

consists of more than $|\mathcal{C}|$ base signatures, the output is $\bot$. Therefore, the length of an aggregated signature is bounded by $\alpha(n,\kappa) \in \mathsf{poly}(\log n, \kappa)$.

Proving decomposability is immediate. Since the aggregation algorithm is deterministic, it can be entirely captured by the first algorithm $\mathsf{Aggregate}_1$, which outputs a set of $\mathsf{polylog}(n)$ signatures (since there are at most $|\mathcal{C}|$ signatures, with all but negligible probability). The second algorithm $\mathsf{Aggregate}_2$ simply outputs the same set of signatures. □

Lemma C.2. *The construction in Figure 7 is βn-robust.*

Proof. Let $\mathcal{A}$ be a PPT adversary. We will show that $\mathcal{A}$ can win the game $\mathsf{Expt}^{\mathsf{robust}}_{\mathsf{tr\text{-}pki},\Pi,\mathcal{A}}(\kappa, n, \beta n)$ (with the trusted PKI mode) with at most negligible probability. The game begins when the challenger computes $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa, 1^n)$ and $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ for every $i \in [n]$. Denote $\mathcal{C} = \{i \mid \mathsf{sk}_i \neq \bot\}$. Next, the adversary adaptively selects the set of corrupted parties; denote by $\mathcal{I}$ the set of corrupted parties.

In the *robustness challenge* phase, the adversary $\mathcal{A}$ chooses an $(n, \mathcal{I}, \mathsf{robust})$-$\mathsf{almost\text{-}everywhere\ communication\ tree}$ $T = (V, E)$ (see Definition 3.3). It also chooses a message $m \in \mathcal{M}$ and $\{m_i\}_{i \in \mathcal{N}}$, where $\mathcal{N}$ is the set of honest parties that are assigned to leaf nodes that do not have a good path to the root.

Recall that there are $n/\log^5 n$ leaf nodes in this tree out of which all but $3/\log n$ fraction have a good path to the root. In other words, the signatures of the parties assigned to "good" leaf nodes are guaranteed to be part of the final aggregate signature. Total number of parties assigned to the good leaf nodes are $\log^5 n \left(1 - \frac{3}{\log n}\right) \frac{n}{\log^5 n} = n \frac{\log n - 3}{\log n}$. Let us use S to denote this set of parties. We proceed to show that with overwhelming probability (in n), there are more than $\ell'/3$ honest parties in S, that have a valid signing key, where $\ell' = \ell/2$.

Claim C.3. $\Pr\left[|\mathcal{C} \cap (S \setminus \mathcal{I})| \leq \ell'/3\right] \leq \mathsf{negl}(n)$.

Proof. We know that $|S| = n\frac{\log n - 3}{\log n} > 2n/3$. In order to maximize its chance of winning the robustness game, an adversary who is allowed to arbitrarily choose the set S, will without loss of generality include all the corrupted parties in S. Denote by $\mathcal{H}_S = S \setminus \mathcal{I}$ the set of honest parties in S. Since $|\mathcal{I}| = (1/3 - \epsilon) \cdot n$ (where $\epsilon = 1/3 - \beta$), it holds that

$$|\mathcal{H}_S| > \frac{2}{3} \cdot n - \left(\frac{1}{3} - \epsilon\right) \cdot n = \left(\frac{1}{3} + \epsilon\right) \cdot n.$$

Thus, there are more than $(1/3 + \epsilon) \cdot n$ honest parties in the set S. Given the information with the adversary and the fact that the set of parties with valid signing keys are chosen at random, he will get the same success probability for any arbitrary choice of $\mathcal{C}$. Let X be a random variable representing the number of honest parties in S who have a valid signing key, i.e., $|\mathcal{C} \cap \mathcal{H}_S|$. If $\Pr\left[|\mathcal{C} \cap \mathcal{H}_S| \leq \ell'/3\right] \leq \mathsf{negl}(n)$ holds for $|\mathcal{C}| = \ell'$, it will also hold for any $|\mathcal{C}| > \ell'$. By Lemma C.1, we know that $|\mathcal{C}| \geq \ell'$ with an overwhelming probability. Therefore, we can assume that $|\mathcal{C}| = \ell'$; in this case it holds that

$$\mathbb{E}[|\mathcal{C} \cap \mathcal{H}_S|] = \frac{\ell'}{n} \cdot \left(\frac{1}{3} + \epsilon\right) \cdot n = \left(\frac{1}{3} + \epsilon\right) \cdot \ell'.$$

By Chernoff bound for $\mu = \ell' (1/3 + \epsilon)$ and $\delta = 3\epsilon/(1 + 3\epsilon)$,[28] it holds that

$$\begin{aligned}\Pr[X \le (1-\delta)\mu] &= \Pr\left[X \le \left(1 - \frac{3\epsilon}{1+3\epsilon}\right) \cdot \ell' \cdot \left(\frac{1}{3} + \epsilon\right)\right] \\ &= \Pr\left[X \le \left(\frac{1}{1+3\epsilon}\right) \cdot \ell' \cdot \left(\frac{1+3\epsilon}{3}\right)\right] \\ &= \Pr[X \le \ell'/3] \\ &\le e^{-\frac{9\epsilon^2}{2(1+3\epsilon)^2}\ell'(1/3+\epsilon)} \\ &= e^{-\frac{3\epsilon^2}{2(1+3\epsilon)}\ell'}.\end{aligned}$$

Since $\epsilon > 0$ is constant, we conclude that

$$\Pr[X \le \ell'/3] \le e^{-\omega(\log n)} = \mathsf{negl}(n).$$

Hence, for any arbitrary strategy deployed by the adversary, the probability that less than $\ell'/3$ honest parties with a valid signing key are chosen in the set S is negligible. □

The robustness phase proceeds with the challenger signing the message m on behalf of all the honest parties $\{\sigma_i\}_{i \in [n] \setminus (\mathcal{I} \cup \mathcal{N})}$ and signing the respective messages m_i on behalf of parties in $\mathcal{N}$ and handing their signatures to $\mathcal{A}$ who responds with signatures for corrupted parties $\{\sigma_i\}_{i \in \{\mathcal{I}\}}$ (potentially also for parties whose signing key is $\perp$). As described in Figure 2, using these "base" signatures $\{\sigma_i\}_{i \in [n]}$, the challenger then interacts with the adversary according to $T = (V, E)$ to compute the aggregate signature σ.

Claim C.4. $\Pr[\mathsf{Verify}(\mathsf{pp}, \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}, m, \sigma) = 0] \le \mathsf{negl}(\kappa, n)$.

Proof. An accepting signature on a message m consists of at least $\ell'/3$ valid signatures of the form $\sigma_i = (i, m, \mathsf{sig}_i)$, satisfying $\mathsf{DS.Verify}(\mathsf{vk}_i, m, \mathsf{sig}_i) = 1$. As proved earlier in Lemma C.1, since $\ell \in \omega(\log n)$ it holds with overwhelming probability that $\ell/2 \le |\mathcal{C}| \le 3\ell/2$; therefore, by the obliviousness of the signature scheme that the aggregate signature can consist of at most $|\mathcal{C}|$ base signatures.

The aggregate algorithm then checks if the "base" signatures contain a valid signature on m. We rely on the correctness of the underlying digital signature scheme to ensure that only valid signatures from the adversary (i.e., by committee members) get aggregated with an overwhelming probability (in κ).

Additionally, in the case where the adversary does not provide sufficiently many valid signatures, from Claim C.3 we know that the number of honest parties in S with a valid signing key is more than $\ell'/3$ with an overwhelming probability (in n). Therefore, the signatures of these honest parties are sufficient for generating an accepting signature. □

This concludes the proof of Lemma C.2. □

Lemma C.5. *The construction in Figure 7 is βn-unforgeable.*

[28] The exact Chernoff bounds used is $\Pr[X \le (1-\delta)\mu] \le e^{-\frac{\delta^2}{2}\mu}$ for $0 < \delta < 1$, where $\mu = \mathbb{E}[X]$.

Proof. Let $\mathcal{A}$ be a PPT adversary. We will show that $\mathcal{A}$ can win the game $\mathsf{Expt}^{\mathsf{forge}}_{\mathsf{tr\text{-}pki},\Pi,\mathcal{A}}(\kappa, n, \beta n)$ with at most negligible probability. The game begins when the challenger computes $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa, 1^n)$ and $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ for every $i \in [n]$. Next, the adversary adaptively selects the set of corrupted parties; denote by $\mathcal{I}$ the set of corrupted parties.

In the *forgery challenge* phase, the adversary $\mathcal{A}$ chooses a subset $S \subseteq [n] \setminus \mathcal{I}$ such that $|S \cup \mathcal{I}| < (1/3 - \epsilon')n$ for some constant $0 < \epsilon' < \epsilon$, where $\epsilon = 1/3 - \beta$, and messages m and $\{m_i\}_{i \in S}$ from $\mathcal{M}$. We now prove that with an overwhelming probability (in n), the fraction of parties who have a valid signing key in $S \cup \mathcal{I}$ is less than a third.

Claim C.6. *The number of parties with a valid signing key in a set $S \cup \mathcal{I}$ is less than $\ell'/3$ with an overwhelming probability in n, i.e.,*

$$\Pr\left[|\mathcal{C} \cap (S \cup \mathcal{I})| \geq \ell'/3\right] \leq \mathsf{negl}(n).$$

Proof. The parties with a valid signing key are chosen at random, and the information about whether a party has a valid signing key is not revealed to the adversary $\mathcal{A}$, unless it chooses to corrupt that party or it sees a signature from that party. The adversary chooses the honest set S only based on the knowledge of corrupted parties and their signing keys. Given this information with the adversary and the fact that the parties with valid signing keys are chosen at random, he will get the same success probability for any arbitrary choice of S.

Let X be a random variable representing the number of parties in $\mathcal{C} \cap (S \cup \mathcal{I})$. If for $|\mathcal{C}| = 3\ell/2$ it holds that $\Pr\left[|\mathcal{C} \cap (S \cup \mathcal{I})| \leq \ell'/3\right] \leq \mathsf{negl}(n)$, it will also hold for any $|\mathcal{C}| < 3\ell/2$. By Lemma C.1, we know that $|\mathcal{C}| < 3\ell/2$ with an overwhelming probability. Therefore, we an assume that $|\mathcal{C}| = 3\ell/2 = 3\ell'$; in this case it holds that $\mathbb{E}[X] = 3(1/3 - \epsilon'')\ell'$ for some $\epsilon'' > \epsilon'$. By Chernoff bound for $\mu = 3\ell'\left(1/3 - \epsilon''\right)$ and $\delta = \frac{9\epsilon'' - 2}{3 - 9\epsilon''}$ (note that $\delta > 0$ since $0 < \epsilon'' < 1/3$),[29] it holds that

$$\begin{aligned}
\Pr\left[X \geq (1+\delta)\mu\right] &= \Pr\left[X \geq \left(1 + \frac{9\epsilon'' - 2}{3 - 9\epsilon''}\right) \cdot \ell' \cdot \left(\frac{1}{3} - \epsilon''\right) \cdot 3\right] \\
&= \Pr\left[X \geq \left(\frac{1}{3 - 9\epsilon''}\right) \cdot \ell' \cdot \left(\frac{3 - 9\epsilon''}{3}\right)\right] \\
&= \Pr\left[X \geq \ell'/3\right] \\
&\leq e^{-\frac{\delta^2}{2+\delta}\mu} \\
&= e^{-\frac{(9\epsilon''-2)^2/(3-9\epsilon'')^2}{(4-9\epsilon'')/(3-9\epsilon'')} 3\ell'(1/3-\epsilon'')} \\
&= e^{-\frac{(9\epsilon''-2)^2}{3(4-9\epsilon'')}\ell'}.
\end{aligned}$$

Since $0 < \epsilon'' < 1/3$ is a constant, it holds that $4 - 9\epsilon'' > 0$, hence we conclude that

$$\Pr\left[X \geq \ell'/3\right] \leq e^{-\omega(\log n)} = \mathsf{negl}(n).$$

Hence, the probability that for any arbitrary strategy deployed by the adversary, the probability that more than $\ell'/3$ of the parties with a valid signing key are in $S \cup \mathcal{I}$ is negligible. $\square$

[29] The exact Chernoff bound used is $\Pr\left[X \geq (1+\delta)\mu\right] \leq e^{-\frac{\delta^2}{(2+\delta)}\mu}$ where $\mu = \mathbb{E}[X]$

The *forgery challenge* phase proceeds when for each $i \in S$, the challenger signs the message m_i on behalf of honest P_i, and signs the message m on behalf of all the remaining honest parties $i \notin S \cup \mathcal{I}$. Next, the challenger hands these signatures $\{\sigma_i\}_{i\in[n]\setminus\mathcal{I}}$ to $\mathcal{A}$ who responds with an aggregate signature $\sigma' \in \mathcal{X}$ and a message $m' \in \mathcal{M}$.

Claim C.7. $\Pr[(\mathsf{Verify}(\mathsf{pp}, \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}, m', \sigma') = 1) \wedge (m' \neq m)] \leq \mathsf{negl}(\kappa, n)$.

Proof. An accepting signature on any message $m' \neq m$ consists of at least $\ell'/3$ valid signatures of the form $\sigma_i = (i, m', \mathsf{sig}_i)$, satisfying $\mathsf{DS.Verify}(\mathsf{vk}_i, m', \mathsf{sig}_i) = 1$.

By Claim C.6, the number of parties with a valid signing key in $S \cup \mathcal{I}$ is less than $\ell'/3$ with an overwhelming probability (in n). Essentially, the adversary receives valid signatures on a message other than m only from less than $\ell'/3$ parties (in $\mathcal{C} \cap (S \cup \mathcal{I})$). Hence, the only way $\mathcal{A}$ can produce more than $\ell'/3$ valid signatures on any message other than m is by forging a valid signature for a corrupt party whose signing key is $\perp$ or by forging a signature for an honest party. Since the verification keys of the parties whose signing keys are $\perp$ correspond to oblivious keys, we rely on the obliviousness of these keys (see Definition 5.2) to ensure that this only happens with negligible probability (in κ). Similarly we can rely on the unforgeability of a digital signature scheme to ensure that an adversary will be able to forge a valid signature for an honest party with a valid signing key only with a negligible probability (in κ). Hence, except with negligible probability $\mathsf{negl}(\kappa, n)$, the adversary is unable to forge an accepting SRDS signature. $\square$

This concludes the proof of Lemma C.5 $\square$

This concludes the proof of Theorem 5.1. $\square$

C.2 SRDS from SNARKs (Cont'd)

We present the proof of Theorem 5.4.

Theorem 5.4. *Let $t < n/3$. Assuming the existence of CRH, digital signatures, and SNARKs with linear extraction, there exists a t-secure SRDS scheme in the CRS model with a bare PKI.*

Proof of Theorem 5.4. In Lemma C.8 we will show that the construction in Figure 8 is succinct, in Lemma C.9, we will show robustness and in Lemma C.10, we will show unforgeability.

Lemma C.8. *The construction in Figure 8 is succinct.*

Proof. We start by proving the size of the signatures is succinct. Each SRDS signature consists of a "truncated transcript" z' of size $(|m| + |c| + |\mathsf{max}| + |\mathsf{min}| + |\gamma|)$ along with a proof π. For "base" SRDS signatures, γ corresponds to a digital signature, and in all other cases $\gamma = \perp$. By definition, the size of each digital signature is $\mathsf{poly}(\kappa)$. Hence, the total size of each truncated transcript z' is $\mathsf{poly}(\kappa) + \log n + \log n + \log n + \log n = \mathsf{poly}(\kappa) + O(\log(n))$. Since $\pi = \perp$ for base signatures, the total size of each base SRDS signature (truncated transcript + digital signature) is $\mathsf{poly}(\kappa) + O(\log(n))$, and is thus succinct.

In each aggregate SRDS signature, this proof corresponds to the output of $\mathsf{PCD.Prover}$. In our construction, the size of PCD transcript z is $|z'| + |H_{\mathsf{vk}}| + |k| + |p| = \mathsf{poly}(\kappa) + O(\log(n))$. The Merkle verification algorithm runs in time $\mathsf{poly}(\kappa) + \mathsf{polylog}(n)$; therefore, by construction, the size of the compliance predicate is $\mathsf{poly}(\kappa) + \mathsf{polylog}(n)$ and the bound B on its running time is $|S_{\mathsf{sig}}| \cdot (\kappa + \mathsf{polylog}(n))$, where $|S_{\mathsf{sig}}| \leq q \leq n$. Therefore, by the *succinctness* property of PCD systems (see Appendix A.2), the size of each proof is $\mathsf{poly}(k+\log B) = \mathsf{poly}(\kappa)\cdot\mathsf{polylog}(n)$. Hence, the total size of each aggregate signature is $\mathsf{poly}(\kappa)\cdot\mathsf{polylog}(n)$.

The time required to verify validity of each "base" signature in this construction is $\mathsf{poly}(\log n, \kappa)$ (here $\mathsf{polylog}(n)$ appears because of the binary representation of indices). The time required to verify a PCD proof in our construction is $\mathsf{poly}(\kappa+|C|+|z|+\log B) = \mathsf{poly}(\kappa+\log n)$ (Definition A.7). Finally, the time required to generate an aggregate signature is equal to the time required to compute z_{out} and the time to run $\mathsf{PCD.Prover}$. The time required to generate z_{out} includes the time required to compute Merkle hash on all the verification keys, which is $\mathsf{poly}(\kappa, n)$, and the time required to verify in the incoming transcripts and proofs, which is $q\cdot\mathsf{poly}(\kappa+\log n)$. Therefore, the running time of $\mathsf{Aggregate}_1$ is $q\cdot\mathsf{poly}(\kappa, n)$. The time required to run $\mathsf{Aggregate}_2$ includes the time required for computing z_{out} given the above information, which is $|S_{\mathsf{sig}}|\cdot O(\log n)$ and the time required to run $\mathsf{PCD.Prover}$, which is $O(\log n)+\mathsf{poly}(\kappa+|C|+\log B) = \mathsf{poly}(\kappa+\log n)$ (see Definition A.7). Therefore, the total time required to run $\mathsf{Aggregate}_2$ is $|S_{\mathsf{sig}}|\cdot\mathsf{poly}(\kappa+\log n) = \mathsf{poly}(\log n, k)$ (since $\|S_{\mathsf{sig}}\|$ is bounded by $\alpha(n,\kappa)\in\mathsf{poly}(\log n,\kappa)$ as enforced by the check in $\mathsf{Aggregate}_1$). □

Lemma C.9. *The construction in Figure 8 is t-robust.*

Proof. Let $\mathcal{A}$ be a PPT adversary. We will show that $\mathcal{A}$ can win the game $\mathsf{Expt}^{\mathsf{robust}}_{\mathsf{b\text{-}pki},\Pi,\mathcal{A}}(\kappa,n,t)$ with at most negligible probability. The game begins when the challenger computes $\mathsf{pp} = (1^\kappa, \sigma_{\mathsf{pcd}}, \tau_{\mathsf{pcd}}, \mathsf{seed}) \leftarrow \mathsf{Setup}(1^\kappa)$ and $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ for every $i\in[n]$. Next, the adversary adaptively selects the set of corrupted parties $\mathcal{I}$ and determines their verification keys.

In the *robustness challenge* phase, the adversary $\mathcal{A}$ chooses an $(n,\mathcal{I},\mathsf{robust})$-almost-everywhere communication tree $T=(V,E)$ (See Definition 3.3). It also chooses a message $m\in\mathcal{M}$ and $\{m_i\}_{i\in\mathcal{N}}$, where $\mathcal{N}$ is the set of honest parties that are assigned to leaf nodes that do not have a good path to the root. Recall that there are $n/\log^5 n$ leaf nodes in this tree out of which all but $3/\log n$ fraction have a good path to the root. In other words, the signatures of the parties assigned to "good" leaf nodes are guaranteed to be part of the final aggregate signature. Total number of parties assigned to the good leaf nodes are $\log^5 n\left(1-\frac{3}{\log n}\right)\frac{n}{\log^5 n} = n\frac{\log n-3}{\log n}$. Let us use S to denote this set of parties. Then $|S| = n\frac{\log n-3}{\log n} > 2n/3$. Since $|\mathcal{I}|<n/3$, it holds that the number of honest parties $\mathcal{H}_S$ in the set S is at least $|\mathcal{H}_S|\geq 2n/3-|\mathcal{I}|>n/3$.

Next, the adversary gets signatures $\{\sigma_i\}_{i\in[n]\setminus\mathcal{I}}$ of all the honest parties on the respective messages (i.e., on message m_i for $i\in\mathcal{N}$ and on message m for $i\in[n]\setminus(\mathcal{I}\cup\mathcal{N})$) and it computes signatures of corrupted parties $\{\sigma_i\}_{i\in\mathcal{I}}$. As described in Figure 2, the challenger then interacts with the adversary according to $T=(V,E)$, to compute the aggregate signature σ.

Recall that the aggregation algorithm first checks the validity of incoming transcripts and proofs and only aggregates transcripts with a convincing proof. Starting from the "base" signatures, if the adversary does not provide valid signatures on m on behalf of the corrupted parties, they will not pass the validity check at level $\ell=2$ (this follows from the *correctness* of the digital signature scheme). The aggregation algorithm on the remaining "verified" base signatures mimics the interactive protocol $\mathsf{Proof}_{\mathsf{Gen}}$ (as described in the *completeness* definition of PCD in Appendix A.2). The tree T chosen by the adversary acts as the *distributed-computation generator* G (see Definition A.7). For each "good" node in T, the reconstruction algorithm aggregates the signatures (i.e., computes a C-compliance transcript and PCD proof) from its incoming edges and labels the outgoing edges from the node with this partially aggregated signature. For every "bad" node in T, the adversary can provide an arbitrary signature of its choice. From soundness of PCDs, it follows that the adversary cannot give a faulty proof/partially aggregate signature that verifies. The aggregation algorithm halts at the root node and outputs the corresponding truncated transcript and proof (i.e., the aggregated signature $(z'_{\mathsf{out}}, \pi_{\mathsf{out}})$). From this construction, we now have that the output transcript is compliant with $\mathtt{C}$, and even if the adversary does not provide valid partially aggregate signatures for bad nodes, since there were at least $n/3$ honest signatures from the honest parties that also had a good path to the root node, from the correctness of the digital signature scheme and completeness of the Merkle hash proof system, it follows that $c_{\mathsf{out}} \geq n/3$. Robustness now follows from the *completeness* and *succinctness* of the PCD system. □

Lemma C.10. *The construction in Figure 8 is t-unforgeable.*

Proof. Let $\mathcal{A}$ be a PPT adversary. We will show that $\mathcal{A}$ can win the game $\mathsf{Expt}^{\mathsf{forge}}_{\mathsf{b\text{-}pki},\Pi,\mathcal{A}}(\kappa, n, t)$ with at most negligible probability. The game begins when the challenger computes $\mathsf{pp} = (1^\kappa, \sigma_{\mathsf{pcd}}, \tau_{\mathsf{pcd}}, \mathsf{seed}) \leftarrow \mathsf{Setup}(1^\kappa)$ and $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ for every $i \in [n]$. Next, the adversary adaptively selects the set of corrupted parties and determines their verification keys; denote by $\mathcal{I}$ the set of corrupted parties.

In the *forgery challenge* phase, the adversary $\mathcal{A}$ chooses a subset $S \subseteq [n] \setminus \mathcal{I}$, such that $|S \cup \mathcal{I}| < n/3$, and messages m and $\{m_i\}_{i \in S}$ from $\mathcal{M}$. Subsequently, for each $i \in S$, the challenger signs the message m_i on behalf of honest P_i, and signs the message m on behalf of all the remaining honest parties $i \notin S \cup \mathcal{I}$. Next, the challenger hands these signatures $\{\sigma_i\}_{i \in [n] \setminus \mathcal{I}}$ to $\mathcal{A}$ who responds with an aggregate signature $\sigma' \in \mathcal{X}$ and a message $m' \in \mathcal{M}$.

Let us assume for the sake of contradiction that the adversary manages to generate an aggregate signature $\sigma' = (z', \pi)$, such that $\mathsf{Verify}(\mathsf{pp}, \{\mathsf{vk}_1, \ldots, \mathsf{vk}_n\}, m', \sigma') = 1$ and $m' \neq m$. From the proof of knowledge property of the PCD system, we know that given a verifying proof from a polynomial-size prover, there exists a polynomial-size extractor $\mathbb{E}_{\mathsf{PCD.Prover}}$ that can extract the witness. Recall that given a vector of input transcripts $\boldsymbol{z}_{\mathsf{in}}$ and an output transcript z_{out}, the compliance predicate in our construction checks if the maximas and minimas of the input and output transcripts are ordered properly, the value of counter c in the output transcript is equal to the sum of the counter values in the input transcripts and that the same Merkle hash of keys is used in all transcripts. Additionally, if any of the input transcripts correspond to base signatures, the compliance predicate also checks that the signature is valid with respect to the verification key specified in that transcript and also verifies the Merkle proof corresponding to this key and the Merkle hash. We now design an adversary $\mathcal{B}$ that uses this extractor to either break unforgeability of the digital signature scheme or break soundness of the Merkle hash proof system. The adversary $\mathcal{B}$ starts by computing $z = z' || (H_{\mathsf{vk}}, \perp, \perp)$, where $H_{\mathsf{vk}} = \mathsf{Merkle.Hash}(\mathsf{seed}, (1||\mathsf{vk}_1), \ldots, (n||\mathsf{vk}_n))$, initializing $S_{\mathsf{val}} = \emptyset$ and running the following recursive algorithm $\mathcal{B}_{\mathsf{ext}}(\sigma_{\mathsf{pcd}}, z)$:

1. Compute $\mathsf{trans} \leftarrow \mathbb{E}_{\mathsf{PCD.Prover}}(\sigma_{\mathsf{pcd}}, z)$.
2. If $\mathtt{C}(\mathsf{trans}) = 1$, for each valid input "base" transcript in trans of the form $(z_i, \perp)$ on m' with $z_i = (m', 1, i, i, \gamma_i, H_{\mathsf{vk}}, k_i, p_i)$ and $\gamma_i \neq \perp$, set $S_{\mathsf{val}} = S_{\mathsf{val}} \cup \{(z_i, \pi_i)\}$. For each partially aggregated signature on m' in trans of the form (z_i, π_i) with $z_i = (m', \cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot)$, check whether $\mathsf{PCD.Verify}(\tau_{\mathsf{pcd}}, z_i, \pi_i) = 1$ and if so, run $\mathcal{B}_{\mathsf{ext}}(\sigma_{\mathsf{pcd}}, z_i)$.

If $|S_{\mathsf{val}}| \geq n/3$, the adversary $\mathcal{B}$ succeeds in extracting at least $n/3$ transcripts of the form $(m', 1, i, i, \gamma_i, H_{\mathsf{vk}}, k_i, p_i)$, each with a distinct i (as enforced by the checks on the maximas and minimas) such that the following holds for each of these transcripts:

(a) $\mathsf{DS.Verify}(\mathsf{vk}_i, m', \gamma_i) = 1$.

(b) $\mathsf{Merkle.Verify}(\mathsf{seed}, (i||k_i), H_{\mathsf{vk}}, p_i) = 1$.

Since H_{vk} was computed honestly by $\mathcal{B}$, it holds for each extracted "base" transcript that either γ_i is a valid signature with respect to $k_i = \mathsf{vk}_i$, or if $k_i \neq \mathsf{vk}_i$, then the adversary $\mathcal{A}$ has managed to break the soundness of the Merkle proof hash proof system. However, from Theorem A.9, we know that this only happens with at most negligible probability (in κ). Now, since each i (and thereby each k_i) is distinct in the extracted "base" transcripts, adversary $\mathcal{B}$ has managed to extract at least $n/3$ valid signatures (γ_i) on m'. Since the adversary only had access to signatures on m' from less than $n/3$ parties, this would imply that it has successfully forged signatures of some honest parties in the set $[n] \setminus S$. From *unforgeability* of the digital signature scheme, we know that this can only happen with at most negligible probability (in κ). □

Lemmas C.8 to C.10 rely on PCD systems for logarithmic-depth and polynomial-size compliance predicates. By Theorem A.8, such PCD systems exist assuming the existence of SNARKs with linear extraction. This concludes the proof of Theorem 5.4. □

D Connection with Succinct Arguments (Cont'd)

In this section, we provide supplementary material for Section 6. In Appendix D.1, we prove Theorem 6.9, and in Appendix D.2 we formally define SNARG-compliant multi-signature schemes.

D.1 Proof of Theorem 6.9

Theorem 6.9. *There exists $s(n) \in \Theta(n)$ such that, for any field $\mathbb{F}$ with $\mathsf{char}(\mathbb{F}) \geq \max(\ell + 2, 63)$, any ring $R = \mathbb{F}^n$ of size $|R| = 2^{\Theta(n)}$ with Hadamard product, and any elementary symmetric polynomial ϕ_ℓ, the (s, R)-Subset-ϕ_ℓ problem is NP-complete.*

Proof. We divide the proof as follows: (1) First, we show that for any ring $R = \mathbb{F}^n$ with Hadamard product satisfying $|R| = 2^{\Theta(n)}$, then for any elementary symmetric polynomial ϕ_2, the R-Subset-ϕ_2 problem (see Definition 6.6) is NP-complete by showing a reduction to 3-SAT. (2) Second, we show the same for any ϕ_ℓ, where $\ell \geq 3$. (3) Finally, we show how these reductions can be modified to prove the existence of $s \in \Theta(n)$, for which (s, R)-Subset-ϕ_ℓ (see Definition 6.6) is also NP-complete.

For R-Subset-ϕ_2: Given a 3-CNF formula Φ over variables $x_1, \ldots, x_N$ with clauses $C_1, \ldots, C_m$, each containing exactly three distinct literals, the reduction algorithm constructs an instance $x = (a_1, \ldots, a_{2+2N+3m}, t)$ of the R-Subset-ϕ_2 problem such that Φ is satisfiable if and only if there exists a subset $S \subseteq [2 + 2N + 3m]$, such that $\phi_2(\{a_i\}_{i \in S}) = t$. The reduction algorithm constructs elements in $R = \mathbb{F}^{1+N+m}$ as follows:

1. A special element $a_1 = \alpha_0 \in R$, whose first entry is 1 and all other entries are 0.
2. A special element $a_2 = \alpha_1 \in R$ whose first $n + 1$ entries correspond to 1.
3. For each variable x_i (for $i \in [N]$), define two elements $a_{2+2i+1} = v_i \in R$ and $a_{2+2i+2} = v'_i \in R$ such that the $(1 + i)$th entry of these elements is set to 1.

4. Define three elements $a_{2+2N+3j+1} = c_j^1, a_{2+2N+3j+2} = c_j^2$, and $a_{2+2N+3j+3} = c_j^3$ corresponding to each clause C_j (for $j \in [m]$). The $(1+N+j)^{\text{th}}$ entry in c_j^1 corresponds to 9, the $(1+N+j)^{\text{th}}$ entry in c_j^2 corresponds to 4 and the $(1+N+j)^{\text{th}}$ entry in c_j^3 corresponds to 2. The remaining entries in each of these correspond to 0.

5. The target element t is also a vector of $1 + N + m$ elements in $\mathbb{F}$. The first $1 + N$ entries in t are set to 1, while the remaining entries are set to 9.

We now prove completeness and soundness of this reduction:

Completeness. Suppose Φ has a satisfying assignment X. We will construct a subset $S \subseteq [2 + 2N + 3m]$ such that $\phi_2(\{a_i\}_{i \in S}) = t$. For each variable x_i, if x_i is set to 1 in X, we include $a_{2+2i+1} = v_i$ in S, else we include $a_{2+2i+2} = v_i'$ in S. We also include the two special elements $a_1 = \alpha_0$ and $a_2 = \alpha_1$ in S. Note that, α_0 and α_1 are the only elements whose first entry is 1, the first entry of all other elements is 0. This ensures that we have exactly 2 elements with value 1 in the first column. Thus, the first entry of t is guaranteed to be 1. Also, apart from α_1, for each $1 \le i \le N$, there are only two other elements v_i and v_i' whose $(1+i)^{\text{th}}$ entry is set to 1. Including one of these for each $1 \le i \le n$ along with α_1 ensures that there are exactly two elements with value 1 in the $(1+i)^{\text{th}}$ column. Therefore, we are guaranteed to get 1 in each of the first $1+N$ entries of t.

Since X is a satisfying assignment, each clause must contain at least one literal with the value 1. For each clause C_j, if there is exactly one literal with value 1 in the satisfying assignment X, we include c_j^1. Note that S now has exactly one element whose $(1+N+j)^{\text{th}}$ entry is set to 1 and exactly one element with 9 in this column. All other elements in the subset have 0's in this position. This ensures that the $(1+N+j)^{\text{th}}$ entry of t adds up to 9. If there are exactly two literals with value 1, we include c_j^2. In this case, there are exactly two elements that have value 1 in the $(1+N+j)^{\text{th}}$ column and exactly one element that has a value of 4 in this position. All other elements in the subset have 0's in this position. This ensures that the $(1+N+j)^{\text{th}}$ entry of t adds up to

$$(1 \cdot 1) + (1 \cdot 4) + (1 \cdot 4) = 9.$$

Finally, if there are exactly three literals with value 1, we include c_j^3. In this case, there are exactly three elements that have value 1 in $(1+N+j)^{\text{th}}$ column and exactly one element that has a value of 2 in this position. All other elements in the subset have 0's in this position. This ensures that the $(1+N+j)^{\text{th}}$ entry in t adds up to

$$(1 \cdot 1) + (1 \cdot 1) + (1 \cdot 1) + (1 \cdot 2) + (1 \cdot 2) + (1 \cdot 2) = 9.$$

Thus, the last m entries in t all add up to 9.

Soundness. Suppose there exists a subset $S \subseteq [2 + 2N + 3m]$ whose pairwise sum of products is t. We show that this implies that there must be a satisfying assignment for Φ. Note that α_0 and α_1 are the only elements whose first entry is 1, while the first entry of all other elements is set to 0. Since the first entry in t is required to be 1, both α_0 and α_1 must be included in the set S.

For each $1 \le i \le N$, there are exactly three elements α_1, v_i and v_i' whose $(i+1)^{\text{th}}$ entry is 1. Since we have already included α_1 in S, if we include both v_i and v_i', then the $(i+1)^{\text{th}}$ entry in result of ϕ_2 applied over S will be $(1 \cdot 1) + (1 \cdot 1) = 2$. Since the characteristic of the field $\mathbb{F}$ is at least 63, we know that $2 \neq 1$. Therefore, we are assured that only one of v_i or v_i' can be included, but not both. Therefore, for each $1 \leq i \leq n$, the set S contains either v_i or v_i'. If $v_i \in S$, we set $x_i = 1$; else we set $x_i = 0$.

We want the last m entries in t to all add up to 9 each. We note that for each $1 \leq j \leq m$, there must be at least one element of the form v_i or v_i' in the subset S that has its $(1+n+j)^{\text{th}}$ entry set to 1. This is because none of the combinations of c_j^1, c_j^2, c_j^3 that have $9, 4, 2$ in this position, respectively, can add up to give 9 when all other elements have 0 in this position:

- If only one of either c_j^1 or c_j^2 or c_j^3 are included in S, then the $(1+n+j)^{\text{th}}$ entry in the result is trivially 0.
- If any two of c_j^1, c_j^2 and c_j^3 are included in S, then the $(1+n+j)^{\text{th}}$ entry in the result is $(9 \cdot 4) = 36$ or $(9 \cdot 2) = 18$ or $(4 \cdot 2) = 8$, depending on which c_j values are included. Since the characteristic of the field $\mathbb{F}$ is at least 63, we know that $36, 18, 8$ are all different than 9.
- If all three of c_j^1, c_j^2 and c_j^3 are included in S, then the $(1+n+j)^{\text{th}}$ entry in the result is $(9 \cdot 4) + (9 \cdot 2) + (4 \cdot 2) = 62$. As before, since the characteristic of the field $\mathbb{F}$ is at least 63, we know that $62 \neq 9$.

Therefore, there is at least one literal in each clause C_j whose value is 1 and Φ has a satisfying assignment.

Having proved NP-completeness of R-Subset-ϕ_2, we proceed to prove the general case of R-Subset-ϕ_ℓ for $\ell \geq 3$.

For R-Subset-ϕ_ℓ, where $\ell \geq 3$: The reduction algorithm for reducing a given 3-CNF formula Φ with N variables $x_1, \ldots, x_N$ and m clauses $C_1, \ldots, C_m$, each containing exactly three distinct literals to an instance of R-Subset-ϕ_ℓ and the proof of soundness for that reduction has already been discussed in the proof sketch of Theorem 6.9 in Section 6.3. Here we only prove the completeness for that reduction.

Completeness. Completeness follows similarly to the previous case. For a satisfying assignment X for Φ, for each $i \in [N]$, either v_i or v_i' is included in subset S. Since each monomial is a combination of ℓ numbers, we include all the special elements $\alpha_0, \alpha_1, \ldots, \alpha_{\ell-1}$ to get the value 1 in the first column ℓ times. This guarantees that the first $N+1$ entries in t are all 1. Since X is a satisfying assignment, each clause contains at least one literal with the value 1. For each clause C_j (for $j \in [m]$), if there is exactly one literal with value 1, we include all the $\ell - 1$ elements c_j. If there are exactly two literals with value 1, we include $\ell - 2$ elements c_j. And if there are exactly three literals with value 1, we include $\ell - 3$ elements c_j. As before, this ensures that the value 1 appears exactly ℓ times in the last m columns and ϕ_ℓ will evaluate to the target value 1 in these positions.

For (s, R)-Subset-ϕ_ℓ for some $s \in \Theta(n)$: Let Φ be a given 3-CNF formula with N variables $x_1, \ldots, x_N$ and m clauses $C_1, \ldots, C_m$. It is easy to see that this instance can be reduced to another 3-CNF instance Φ' with $n' = \mathsf{max}(m, N)$ variables and $n' = \mathsf{max}(m, N)$ clauses by adding "dummy" variables and clauses. We can then use the reduction algorithms discussed above to reduce Φ' to an instance of R-Subset-ϕ_ℓ with $n = \ell + 2n' + (\ell - 1)n'$ elements in R. Recall that this reduction is such that for a satisfying assignment X' for Φ', the corresponding witness S for the R-Subset-ϕ_ℓ instance contains the following:

- ℓ elements: It contains elements $\alpha_0, \ldots, \alpha_{\ell-1}$.
- n' elements: For each $i \in [n']$, it either contains v_i or v_i'.
- At least $(\ell-3)n'$ elements: Depending on how many literals have value 1, in clause C_j (for $j \in [n']$), S contains at least $\ell - 3$ elements c_j.

As a result, the subset S for the R-Subset-ϕ_ℓ instance contains at least $\ell + n' + (\ell-3)n'$ out of $n = \ell + 2n' + (\ell-1)n'$ elements, i.e., $s = |S| \in \Theta(n)$ for each $\ell \in [n]$. In other words, there exists $s \in \Theta(n)$, for which (s, R)-Subset-ϕ_ℓ is NP-complete.

This concludes the proof of Theorem 6.9. $\square$

D.2 SNARG-Compliant Multi-Signatures and Subset-ϕ_ℓ

In this section, we identify the properties of multi-signatures used in Lemma 6.5 to provide the connection with average-case SNARGs. We call multi-signature schemes that satisfy these properties as *SNARG-compliant* multi-signature schemes.

Definition D.1 (SNARG-compliant multi-signatures). *A multi-signature scheme* ($\mathsf{MS.KeyGen}$, $\mathsf{MS.Sign}$, $\mathsf{MS.Verify}$, $\mathsf{MS.Combine}$, $\mathsf{MS.MVerify}$) *is SNARG compliant if it satisfies the following properties:*

1. *The algorithm* $\mathsf{MS.MVerify}$ *is deterministic.*
2. *Verification keys are independently and uniformly sampled from a ring $R = \mathbb{F}^k$ (for some k) with Hadamard Product.*
3. *There exist polynomial-time algorithms* $\mathsf{MS.Verify}_{\mathsf{agg\text{-}key}}$ *and* f_{agg}*, such that given a multi-signature* $\sigma_{\mathsf{ms}} \in \mathcal{X}_{\mathsf{ms}}$ *on a message* $m \in \mathcal{M}$*, corresponding to a set of keys* $\{\mathsf{vk}_i\}_{i \in S}$ *for some subset* $S \subseteq [n]$*, the algorithm* $\mathsf{MS.MVerify}(\mathsf{pp}_{\mathsf{ms}}, \{\mathsf{vk}_i\}_{i\in[n]}, S, m, \sigma_{\mathsf{ms}})$ *can be decomposed as follows:*
   (a) $\mathsf{vk}_{\mathsf{agg}} = f_{\mathsf{agg}}(\{\mathsf{vk}_i\}_{i\in S})$.
   (b) $b = \mathsf{MS.Verify}_{\mathsf{agg\text{-}key}}(\mathsf{pp}_{\mathsf{ms}}, \mathsf{vk}_{\mathsf{agg}}, m, \sigma_{\mathsf{ms}})$.
4. *There exists a PPT algorithm* $\mathsf{MS.MVerifyInv}$ *that on input the public parameters* $\mathsf{pp}_{\mathsf{ms}}$*, a message m and a multi-signature* σ_{ms}*, outputs* $\mathsf{vk} \in R$.

   We require that for $\mathsf{vk}_{\mathsf{agg}} = f_{\mathsf{agg}}(\{\mathsf{vk}_i\}_{i\in S}) \in R$ *and* $\mathsf{MS.Verify}_{\mathsf{agg\text{-}key}}(\mathsf{pp}_{\mathsf{ms}}, \mathsf{vk}_{\mathsf{agg}}, m, \sigma_{\mathsf{ms}}) = 1$*, it holds that* $\mathsf{MS.MVerifyInv}$ *computes the corresponding unique and well-defined key* $\mathsf{vk}_{\mathsf{agg}}$*, i.e.,*
   $$\mathsf{MS.MVerifyInv}(\mathsf{pp}_{\mathsf{ms}}, m, \sigma_{\mathsf{ms}}) = f_{\mathsf{agg}}(\{\mathsf{vk}_i\}_{i\in S}).$$
5. *There exist degenerate keys* $\mathsf{sk}_{\mathsf{deg}}$ *and* $\mathsf{vk}_{\mathsf{deg}}$*, and a PPT algorithm* $\mathsf{MS.Sign}_{\mathsf{deg\text{-}key}}$ *such that* $\sigma_{\mathsf{ms}} \leftarrow \mathsf{MS.Sign}_{\mathsf{deg\text{-}key}}(\mathsf{pp}_{\mathsf{ms}}, \mathsf{sk}_{\mathsf{deg}}, m)$ *satisfies* $\mathsf{MS.Verify}_{\mathsf{agg\text{-}key}}(\mathsf{pp}_{\mathsf{ms}}, \mathsf{vk}_{\mathsf{deg}}, m, \sigma_{\mathsf{ms}}) = 1$.

We now show that an SRDS scheme based on a SNARG-compliant multi-signature scheme with key-aggregation function $f_{\mathsf{agg}} = \phi_\ell$, implies SNARGs for average-case Subset-ϕ_ℓ. This reduction can be viewed as a generalization of Lemma 6.5.

Lemma D.2. *Let $\mathbb{F}$ be a field, let $R = \mathbb{F}^k$ (for some k) be a ring with Hadamard product, let ϕ_ℓ (for some $\ell \in \mathbb{N}$, $\ell > 1$) be an elementary symmetric polynomial over R, let $0 < \alpha < 1$ be a constant, and let $s(n) = \alpha \cdot n$. Assume that $|\mathbb{F}| = n^{\omega(1)}$ and that $n/\log|R| < 1$.*

If there exists an SRDS scheme based on a SNARG-compliant multi-signature scheme with key-aggregate function $f_{\mathsf{agg}} = \phi_\ell$, then there exist SNARGs for average-case (s, R)-Subset-ϕ_ℓ.

Proof. We give a construction of average-case SNARGs for (s, R)-Subset-ϕ_ℓ using an SRDS scheme based on an SRDS-compliant multi-signature scheme as per Definitions 6.2 and D.1.

1. $\mathsf{S.Setup}(1^\kappa, 1^n)$: Run the setup of the SRDS scheme $\mathsf{Setup}(1^\kappa, 1^n)$ to output $\mathsf{crs} = (\mathsf{pp_{ms}}, \mathsf{pp}_2)$.

2. $\mathsf{S.Prove}(\mathsf{crs}, x, w)$: Given an average-case yes instance-witness pair $(x, w) \leftarrow \mathcal{D}_{\mathsf{yes}}(1^n)$ of the form $x = (a_1, \ldots, a_n, t)$ and $w = S$, proceed as follows:

   - Let $\alpha = \phi_{\ell-1}(\{a_i\}_{i\in S})$ and let $\mathsf{vk_{deg}}$ be the degenerate aggregate verification key. If α does not have an inverse in R, output $\perp$ and terminate. Else, compute
   $$a_{n+1} = (\mathsf{vk_{deg}} - \phi_\ell(\{a_i\}_{i\in S})) \cdot \alpha^{-1} = (\mathsf{vk_{deg}} - t) \cdot \alpha^{-1}.$$
   Parse $\mathsf{crs} = (\mathsf{pp_{ms}}, \mathsf{pp}_2)$ and interpret the set $\{a_1, \ldots, a_n, a_{n+1}\}$ as a set of $n+1$ verification keys $\{\mathsf{vk}_1, \ldots, \mathsf{vk}_{n+1}\}$. Note that $\phi_\ell(\{\mathsf{vk}_i\}_{i\in S'}) = \mathsf{vk_{deg}}$ for $S' = S \cup \{n+1\}$.
   - Choose an arbitrary $m \in \mathcal{M}$ and use $\mathsf{MS.Sign_{deg\text{-}key}}$ (as defined in Definition D.1) to compute
   $$\sigma_{\mathsf{ms}} \leftarrow \mathsf{MS.Sign_{deg\text{-}key}}(\mathsf{pp_{ms}}, \mathsf{sk_{deg}}, m).$$
   - Use the algorithm P (that exists from Definition 6.2) to compute
   $$\pi \leftarrow \mathsf{P}(\mathsf{crs}, \mathsf{vk}_1, \ldots, \mathsf{vk}_{n+1}, S', m, \sigma_{\mathsf{ms}}).$$
   - Finally, output $(m, \sigma_{\mathsf{ms}}, \pi)$.

3. $\mathsf{S.Verify}(\mathsf{crs}, x, \pi)$: Parse $\mathsf{crs} = (\mathsf{pp_{ms}}, \mathsf{pp}_2)$ and $x = (a_1, \ldots, a_n, t)$, and proceed as follows:

   - Compute a_{n+1} as in the prover algorithm. Interpret the set $\{a_1, \ldots, a_n, a_{n+1}\}$ as a set of $n+1$ verification keys $\{\mathsf{vk}_1, \ldots, \mathsf{vk}_{n+1}\}$.
   - Compute $\mathsf{vk} = \mathsf{MS.MVerifyInv}(\mathsf{pp_{ms}}, m, \sigma_{\mathsf{ms}})$ and check if vk equals the degenerate verification key $\mathsf{vk_{deg}}$ (that, by construction, satisfies $\mathsf{vk_{deg}} = \phi_\ell(\{\mathsf{vk}_i\}_{i\in S'})$). Set $b' = 1$ if $\mathsf{vk} = \mathsf{vk_{deg}}$ and $b' = 0$ otherwise.
   - Run the verification algorithm of the SRDS scheme
   $$b \leftarrow \mathsf{Verify}((\mathsf{pp_{ms}}, \mathsf{pp}_2), \mathsf{vk}_1, \ldots, \mathsf{vk}_n, m, (\sigma_{\mathsf{ms}}, \pi)).$$
   - Finally output $b \wedge b'$.

We now argue succinctness, completeness, and average-case soundness for this construction:

Succinctness. Succinctness follows from the succinctness of the SRDS scheme.

Completeness. Recall that each of the values $(a_1, \ldots, a_n)$ in an average case yes instance is sampled uniformly at random; hence, the output of an elementary symmetric polynomial on a randomly chosen subset S of these values is also uniformly distributed. Given any average-case yes instance-witness pair $(x, w) \leftarrow \mathcal{D}_{\mathsf{yes}}(1^n)$ of the form $x = (a_1, \ldots, a_n, t)$ and $w = S$, the probability that $\phi_{\ell-1}(\{a_i\}_{i\in S})$ has an inverse in R is $1 - k/|\mathbb{F}|$.[30] Since our proof system only works for such instances, the rest of this argument assumes that this is the case. Given $x = (a_1, \ldots, a_n, t)$ and $w = S$, it holds that $f_{\mathsf{agg}}(\{a_i\}_{i\in S}) = \phi_{\ell-1}(\{a_i\}_{i\in S}) = t$ or equivalently, it holds for $S' = S \cup \{n+1\}$ that

$$\phi_\ell(\{a_i\}_{i\in S'}) = \phi_\ell(\{a_i\}_{i\in S}) + a_{n+1} \cdot \phi_{\ell-1}(\{a_i\}_{i\in S}) = \mathsf{vk}_{\mathsf{deg}}.$$

Recall in an SRDS-compliant multi-signature scheme, it holds that

$$\mathsf{MS.MVerifyInv}(\mathsf{pp}_{\mathsf{ms}}, m, \sigma_{\mathsf{ms}}) = \phi_\ell(\{\mathsf{vk}_i\}_{i\in S'}) = \mathsf{vk}_{\mathsf{deg}}.$$

Hence, $\mathsf{MS.Verify}_{\mathsf{agg\text{-}key}}(\mathsf{pp}_{\mathsf{ms}}, \mathsf{vk}_{\mathsf{deg}}, m, \sigma_{\mathsf{ms}}) = 1$, i.e., σ_{ms} is a valid multi-signature on m with respect to $\mathsf{vk}_{\mathsf{deg}}$. Since the multi-signature satisfies $\mathsf{MS.MVerifyInv}(\mathsf{pp}_{\mathsf{ms}}, m, \sigma_{\mathsf{ms}}) = \mathsf{vk}_{\mathsf{deg}}$, completeness of SRDS based on an SRDS-compliant multi-signature scheme (see Definition 6.2) implies that the output of P, given this signature and S' will be a valid SRDS signature. Completeness now holds with an overwhelming probability since $\phi_{\ell-1}(\{a_i\}_{i\in S})$ has an inverse in R with an overwhelming probability of $1 - k/|\mathbb{F}|$.

Average-case soundness. Recall that each of the values $(a_1, \ldots, a_n, t)$ in $x \leftarrow \mathcal{D}_{\mathsf{no}}(1^n)$ is sampled uniformly at random. Let $\alpha = \phi_{\ell-1}(\{a_i\}_{i\in S})$ and assume that α^{-1} exists. Since t is a randomly sampled value, so is $a_{n+1} = (\mathsf{vk}_{\mathsf{deg}} - t) \cdot \alpha^{-1}$ for any $S \subseteq [n]$. We interpret the set of $n+1$ verification keys as $\mathsf{vk}_i = a_i$ for $i \in [n+1]$; thus, the verification keys $\{\mathsf{vk}_1, \ldots, \mathsf{vk}_{n+1}\}$ are uniformly distributed over R. Since $n/\log|R| < 1$ and the output of elementary symmetric polynomials is uniformly distributed, then with overwhelming probability (bounded by $2^{n+1}/|R|$), there does not exist a subset $S' \subseteq [n+1]$ of size $s+1$, such that $\phi_\ell(\{a_i\}_{i\in S'}) = \mathsf{vk}_{\mathsf{deg}}$.

Given $(m, \sigma_{\mathsf{ms}}, \pi)$, we check if: (1) σ_{ms} is a valid multi-signature on m with respect to $\mathsf{vk}_{\mathsf{deg}}$ and (2) if $(\sigma_{\mathsf{ms}}, \pi)$ is a valid SRDS on m. Recall that in a SNARG-compliant multi-signature scheme, given a multi-signature σ_{ms}, a message m, and public parameters $\mathsf{pp}_{\mathsf{ms}}$, there exists a unique aggregate verification key $\mathsf{vk}_{\mathsf{agg}}$ with respect to which σ_{ms} verifies, i.e.,

$$\mathsf{MS.MVerifyInv}(\mathsf{pp}_{\mathsf{ms}}, m, \sigma_{\mathsf{ms}}) = \mathsf{vk}_{\mathsf{agg}}.$$

Therefore, if check (1) goes through, then $\mathsf{vk}_{\mathsf{agg}} = \mathsf{vk}_{\mathsf{deg}}$ is the only aggregate verification key for which σ_{ms} is a valid multi-signature on m. As argued earlier, with a high probability there does not exist a subset $S' \subseteq [n+1]$ such that $\phi_\ell(\{\mathsf{vk}_i\}_{i\in S'}) = \mathsf{vk}_{\mathsf{deg}}$. Also, from the soundness of SRDS based on a multi-signature scheme (Definition 6.2), we know that if there does not exist a subset $S' \subseteq [n+1]$ of size $s+1$, such that σ_{ms} is a valid multi-signature on m with respect to $\{\mathsf{vk}_i\}_{i\in S'}$, then the probability of an adversary computing a valid SRDS $(\sigma_{\mathsf{ms}}, \pi)$ on a message m is negligible. Soundness now follows from the soundness of SRDS based on a multi-signature scheme. □

[30]We note that all elements of $R = \mathbb{F}^k$, except for the ones with a 0 in any of its vector coordinates, have an inverse in R.