Communication-Efficient Proactive Secret Sharing for Dynamic Groups with Dishonest Majorities

Karim Eldefrawy[1(✉)], Tancrède Lepoint[2], and Antonin Leroux[3]

[1] SRI International, Menlo Park, USA
karim.eldefrawy@sri.com
[2] Google LLC, New York, USA
tancrede@google.com
[3] École Polytechnique, Palaiseau, France
antonin.leroux@polytechnique.edu

Abstract. In Secret Sharing (SS), a dealer shares a secret s among n parties such that an adversary corrupting no more than t parties does not learn s, while any $t+1$ parties can efficiently recover s. Proactive Secret Sharing (PSS) retains confidentiality of s even when a *mobile adversary* corrupts all parties over the secret's lifetime, but no more than a threshold t in each epoch (called a refresh period). Withstanding such adversaries is becoming increasingly important with the emergence of settings where private keys are secret shared and used to sign cryptocurrency transactions, among other applications. Feasibility of (single-secret) PSS for *static groups* with *dishonest majorities* was recently demonstrated, but with a protocol that requires inefficient communication of $O(n^4)$.

In this work, using new techniques, we improve over prior work in two directions: *batching without incurring a linear loss in corruption threshold and communication efficiency*. While each of properties we improve upon appeared independently in the context of PSS and in other previous work, handling them simultaneously (and efficiently) in a single scheme faces non-trivial challenges. SomePSS protocols can handle batching of $\ell \sim n$ secrets, but all of them are for the honest majority setting. The techniques typically used to accomplish such batching decrease the tolerated corruption threshold bound by a linear factor in ℓ, effectively limiting the number of elements that can be batched with dishonest majority. We solve this problem by finding a way to reduce the decrease to $\sqrt{\ell}$ instead, allowing to reach the dishonest majority setting when $\ell \sim n$. Specifically, this work introduces new bivariate-polynomials-based sharing techniques allowing to batch up to $n-2$ secrets in our PSS. Next, we tackle the efficiency bottleneck and construct a PSS protocol with $O(n^3/\ell)$ communication complexity for ℓ secrets, i.e., an amortized communication complexity of $O(n^2)$ when the maximum batch size is used.

Keywords: Secret Sharing · Proactive adversary · Dishonest majorities

T. Lepoint and A. Leroux—Work performed while the author was at SRI International.

M. Conti et al. (Eds.): ACNS 2020, LNCS 12146, pp. 3–23, 2020.
https://doi.org/10.1007/978-3-030-57808-4_1

1 Introduction

Secret sharing (SS) is a fundamental cryptographic primitive used to construct secure distributed protocols and systems [1,9,10,17–19,22,26], and in particular secure multiparty computation (MPC) [2,4–6,11–13,25,27,30,32]. In standard SS [7,34], a secret s is encoded in a distributed form among n parties such that an adversary corrupting up to t parties cannot learn the secret s, while any $t+1$ parties can efficiently recover s. In some settings, SS should guarantee confidentiality of shared secrets and correctness of the computations performed on the shares (if such computation is required), even when the protocol is run for a long time [30]. Similarly, there are settings where secret shared private keys are used sporadically over long period of time, for example to (threshold) sign cryptocurrency transactions [8,16,24,28] or in other financial applications and settings [29]. Requiring security for long durations brings forward a new challenge, as it gives a *mobile adversary* the chance to *eventually* corrupt all parties. Ensuring security against such (mobile) adversaries has recently become of increasing importance. An SS protocol that withstands mobile adversaries is called a *Proactive Secret Sharing (PSS) protocol* [26,30].

In this work, we construct an *efficient* PSS protocol with the following key properties: (i) *batching without incurring linear loss in the corruption threshold*, (ii) *tolerating dishonest majorities*, and (iii) *efficient communication*. We achieve this with new techniques based on bivariate sharing. Below, we summarize gradually the progression from standard SS for passive adversaries to PSS for static groups with dishonest majorities. We explain why either the tolerated threshold or the performance of existing protocols fall short in simultaneously achieving the goals we strive towards. *Protocols required to handle dynamic groups are left out of this version due to space constraints, the details of handling dynamic groups can be found in the full version* [20].

Prior Work. A SS protocol [7,34] typically consists of two sub-protocols: `Share` and `Reconstruct`. `Share` can be used by a dealer to share a secret s among n parties such that an adversary corrupting no more than t parties does not learn s, while any $t+1$ parties can efficiently recover s via `Reconstruct`. Initially, SS schemes only considered (exclusively) *passive* or *active adversaries*. In the malicious setting, we say that a SS scheme is verifiable if some auxiliary information is exchanged that allows players to verify that the shares are consistent; such a SS scheme is called Verifiable Secret Sharing (VSS).

The Mixed Adversary Model and Gradual Secret Sharing. In [27], Hirt, Maurer, and Lucas introduced the concept of mixed adversaries in SS and MPC, to capture the trade-off between passive and active corruptions. In particular, they develop an MPC protocol using *gradual* VSS against mixed adversaries that corrupt k parties actively out of less than $n-k$ corrupted parties total. One of the main benefits of gradual SS is to ensure *fairness*, i.e., if corrupted parties can deny the output of a protocol to the set of honest parties, then they cannot

learn the secret themselves. The key idea is to additively share the secret s (i.e., $s = \sum_{i=1}^{d} s_i$) and then linearly share each of the s_i to the parties under polynomial of (gradually) increasing degrees $i = 1$ to $i = d$. In the `Reconstruct` protocol, the parties open the shares gradually, from $i = d$ to $i = 1$ and incorrect parties cannot deviate without being detected.

Proactive Secret Sharing (PSS). It may be desirable to guarantee the security of standard (and gradual) SS throughout the entire lifetime of the secret. The notion of proactive security was first suggested by Ostrovsky and Yung [30], and applied to SS later [26]. Proactive security protects against a mobile adversary that can change the subset of corrupted parties over time. Such an adversary could eventually gain control of all parties over a long enough period, but is limited to corrupting no more than t parties during the same time period. In this work, we use the definition of PSS from [16,27]: in addition to `Share` and `Reconstruct`, a PSS scheme contains a `Refresh` and a `Recover` sub-protocols. `Refresh` produces new shares of s from an initial set of shares. An adversary who controls a subset of the parties before the refresh and the remaining subset of parties after, will not be able to reconstruct the value of s. `Recover` is required when one of the participant is rebooted to a clean initial state. In this case, the `Recover` protocol is executed by all other parties to provide shares to the rebooted party. Such rebooting should ideally be performed sequentially for randomly chosen parties (irrespective of their state of corruption) at a predetermined rate – hence the "proactive" security concept. In addition, `Recover` could be executed after an active corruption is detected.

Dynamic Proactive Secret Sharing (DPSS). PSS initially considered only static groups. DPSS schemes are both proactively secure and allow the set of parties to change over time [3,15,33,36,37]. To the best of our knowledge, *there are currently no DPSS schemes for dynamic groups with dishonest majorities.* The authors in [3] extended the PSS introduced in [2] with ideas from [12–14] to produce a DPSS scheme for honest majorities only. We follow a similar approach to extend our PSS to dynamic settings (details in the full version [20]).

Limitations of Prior Work. Our goal in this work is to address limitations and gaps in, and open problems left by, prior PSS work, as shown in Table 1. First, the only PSS in the dishonest majority setting [16,21] assumes a static group of parties, i.e., unchanged during the secret lifetime. In this work, we present a PSS protocol with dishonest majorities. Security of our protocols is computational. Second, existing PSS protocols for the dishonest majority setting [16,21] do not explicitly handle batching of secrets [23], i.e., sharing, refreshing, and recovering shares of several secrets in parallel. While the authors in [16] mention a batched version of their PSS, the paper does not provide any detail on the effect of batching on the communication complexity nor on the security impact in the mixed adversary setting. In this work, we introduce a notion of batched PSS that retains fairness against mixed adversaries. Third, previous

Table 1. Overview of features and limitations of current PSS protocols. Communication complexity is amortized over the number of secrets handled by the schemes. Note that batching is briefly mentioned in [16], but no technical details are provided. A detailed comparison of complexity of the sub-protocols and tolerated corruption thresholds is provided in Table 2 and in the full version [20].

	PSS	Batching	Fairness	Dynamic groups	Dishonest majority	Communication (amortized)
[2]	✓	✓	✗	✗	✗	$O(1)$
[3]	✓	✓	✗	✓	✗	$O(1)$
[16,21]	✓	✗	✓	✗	✓	$O(n^4)$
This work	✓	✓	✓	✓	✓	$O(n^2)$

Table 2. Comparison of amortized communication complexity of sub-protocols in this work and existing PSS schemes in the dishonest majority setting for ℓ secrets. The communication complexities stated in the column "Dynamic" are the worst-case of three sub-protocols required to handle dynamic groups (see [20]). We note that the complexity of the `Recover` sub-protocol is per party, and this is the bottleneck since it has to be repeated n times, once when each party is (eventually) rebooted. This explains the $O(n^4)$ overall communication complexity of [16,21], this is not an issue in our work because our bottleneck is the `Reconstruct` sub-protocol not the `Recover` one.

	ℓ	PSS `Share`	PSS `Reconstruct`	PSS `Refresh`	PSS `Recover`	Dynamic `Redistribute`	Overall
[16,21]	1	$O(n^2)$	$O(n^2)$	$O(n^3)$	$O(n^3)$	–	$O(n^4)$
This work	$n-2$	$O(n)$	$O(n^2)$	$O(n)$	$O(n)$	$O(n^2)$	$\boldsymbol{O(n^2)}$
This work	1	$O(n)$	$O(n^3)$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$\boldsymbol{O(n^3)}$

PSS protocols [16,21] have a large communication complexity, $O(n^4)$, and an open problem is how to reduce the communication bottleneck in the PSS (due to the `Refresh` and `Recover` sub-protocols) to $O(n^2)$ or $O(n^3)$. Moreover, the additional *fairness* feature of [16] is costly in terms of communication and it is not clear how this additional cost can be handled. We solve these open questions by reducing the (amortized) communication complexity to $O(n^2)$ in the batched setting, and $O(n^3)$ in the single secret setting. In theses improvements, we obtain *fairness* with no additional cost in asymptotic communication complexity.

Our Contributions. In this work, we develop a new communication-efficient PSS protocol for groups dishonest majorities. To achieve this, we proceed in the following steps.

(1) Batched PSS (Without Linear Loss in the Corruption Threshold). The main feature of this work is a new PSS scheme with $O(n^2)$ amortized communication complexity, improving by a quadratic factor the complexity of the best known

construction for dishonest majority [16]. This improvement is mainly obtained through a bivariate polynomials based batching technique, which deviates from how secret sharing is performed in all previous PSS schemes for dishonest majorities. While bivariate polynomials have been used before for secret sharing, we devise a new approach to compute blinding bivariate polynomial (used in the recover protocol, Protocol 2) that will result in a significant improvement in the communication complexity. It is well-known that linear secret sharing with threshold t can be extended to share ℓ secrets $s_1, \ldots, s_\ell$ by sampling a random polynomial f of degree t such that $f(\beta_i) = s_i$ for public values $\beta_1, \ldots, \beta_\ell$ and distributing shares $f(\alpha_i)$ for $i = 1, \ldots, n$ to the n parties. However, in order to learn no information about $(s_1, \ldots, s_\ell)$ an adversary must learn at most $t - \ell + 1$ evaluations of f, which yields a secret sharing scheme with threshold $t - \ell + 1$. To remove this linear dependency, we revisit the idea of using secret sharing with bivariate polynomials (e.g., [35]). Our new approach to construct PSS from bivariate polynomials preserves secrecy with a corruption threshold of $t - \sqrt{\ell} + 1$ for any $\ell \leq n - 2$. This yields a batched PSS scheme with sublinear reduction in the corruption threshold.[1] Since gradual SS consists of a ladder of polynomial shares, the same linear dependency in the number of secrets being batched applies to the mixed adversarial model considered in [27], which then becomes secure against mixed adversaries that corrupt k parties actively out of less than $n - k - \ell$ corrupted parties total. Similarly, we introduce the notion of *batched gradual VSS*, a batched generalization of gradual VSS [27] which is secure against adversaries corrupting either $t - \lfloor\sqrt{\ell}\rfloor$ parties passively, or $((n - \lfloor\sqrt{\ell}\rfloor)/2) - 1$ parties actively, or k parties actively out of $n - k - \lfloor\sqrt{\ell}\rfloor$. Gradual SS aims to obtain fairness during the reconstruction, we note that our gradual batched PSS obtains this fairness property without any additional asymptotic costs.

(2) Efficient Communication. The techniques used above were also carefully designed to limit the communication complexity. As shown in Table 2, our (fully) batched PSS with dishonest majorities has an overall complexity[2] of $O(n^2)$ per secret.

(3) Accommodating Dynamic Groups. Additionally, we develop sub-protocols for nodes to join the (SS) group, and leave it, without increasing the overall (asymptotic) communication complexity. Protocols required to handle dynamic groups are left out of this version due to space constraints, the details of handling dynamic groups can be found in the full version [20].

Intuition Behind New Techniques for the Proactive Setting. We summarize here the main intuition behind the techniques that enable our performance and threshold improvements outlined in Sect. 1.

[1] In [20], we give a self-contained description of the special case of $\ell = 1$ that improves the communication complexity of the gradual VSS of [16] by $O(n)$ without any decrease in the corruption threshold.

[2] To simplify notation we often write complexity instead of amortized complexity.

(1) Addressing the Recover Bottleneck in PSS. As mentioned above, the real bottleneck of PSS is the `Recover` protocol. This protocol is costly in itself and is performed regularly on each of the participants (adding a $O(n)$ complexity factor to the overall communication complexity). The main challenge is to efficiently generate a set of blinding polynomials. We revisit the `Recover` protocol to overcome this limitation by optimizing the number of blinding polynomials generated. This improvement is made possible by the use of bivariate polynomials. The `Recover` protocol is also a necessary building block for our new `Refresh` protocol and the "gradual" `Reconstruct` protocol (see item (3) below), as it enables a subset of participants to generate random polynomials and share them with the rest of the parties.

(2) Batching with Bivariate Polynomials: Batching $O(n)$ secrets saves $O(n)$ in the overall communication complexity, but usually reduces the threshold by a linear factor proportional to the number of batched secrets. This severely limits the number of elements one can batch. We use bivariate polynomials to perform sharing (of a batch of secrets) instead of univariate polynomials. As mentioned above, the real bottleneck in this protocol is the generation of blinding polynomials in `Recover` that protects the secrets without changing their values. We develop a new technique to generate these polynomials in $O(n^2)$ with the number of blinding values being quadratic in $n - t_P$. To obtain information theoretic security for the batched secrets we need this term $(n - t_P)^2$ to be greater than ℓ. This leads to only a sub-linear $\sqrt{\ell}$ reduction in the threshold, as opposed to linear in ℓ. Note that our generation of blinding bivariate polynomial is optimal. Indeed, this blinding polynomial has degree d and the data size of a bivariate polynomial of this degree is $O(n^2)$ when we take $d = O(n)$ (in practice, we take $d = n - 2$ for maximum security). Hence, our technique cannot yield a protocol with better communication complexity than $O(n^2)$.

(3) Gradual Property only Needed in Reconstruction. We also observe that, in previous work, the "gradual" feature of the underlying SS scheme (to withstand dishonest majorities) is critically used during the `Reconstruct` operation only. We will therefore work only with regular shares. To recreate a gradual SS, we develop a new (gradual technique at the core of the) `Reconstruct` protocol that creates directly a ladder of blinding polynomials that sum to 0, adds the shares of the first element of the ladder, and then gradually reveals everything while preserving confidentiality of the shared secrets. At the bottom layer, what is revealed is the actual shared secret because all the blinding polynomials of the ladder add up to 0. This enables us to save an additional factor (after batching) of $O(n)$ in `Reconstruct`. This results in a final communication complexity of $O(n^2)$ for the `Reconstruct` which was the bottleneck as shown in Table 2. This also implies that we can obtain *fairness* during the reconstruction without increasing the total communication complexity.

Outline. The rest of this paper is organized as follows: Sect. 2 overviews some preliminaries required for the rest of the paper. Section 3 provides the definition of batched PSS, i.e., multi-secret PSS with dishonest majorities. Section 4 presents a concrete efficient instantiation of a batched (static) PSS using bivariate polynomials. Technical details, i.e., sub-protocols and their proofs, required to extend the above PSS scheme to deal with dynamic groups are provided in the full version [20].

2 Preliminaries

Throughout this paper, we consider a set of n parties $\mathcal{P} = \{P_1, ..., P_n\}$, connected by pairwise synchronous secure (authenticated) channels and an authenticated broadcast channel. $\mathcal{P}$ want to share and proactively maintain a confidential secret s over a finite field $\mathbb{F} = \mathbb{Z}_q$ for a prime q.

For integers a, b, we denote $[a, b] = \{k : a \leq k \leq b\}$ and $[b] = [1, b]$.[3] We denote by $\mathbb{P}_k$ the set of polynomials of degree k exactly over $\mathbb{F}$. When a variable v is drawn randomly from a set S, we denote $v \leftarrow S$.

2.1 Mixed Adversaries

We first recall the model of mixed adversaries from [27]; we consider a central adversary $\mathcal{A}$ with polynomially bounded computation power who corrupts some parties passively (i.e., $\mathcal{A}$ learns the view of a P_i) and actively (i.e., $\mathcal{A}$ makes a P_i behave arbitrarily) during a stage σ. We denote by $\mathcal{P}_P$ (resp. $\mathcal{P}_A \subseteq \mathcal{P}_P$) the set of passively (resp. actively) corrupted parties and denote by t_P (resp. t_A) its cardinality. A multi-threshold is a set of pairs of thresholds (t_1, t_2). We say that $(t_P, t_A) \leq T$ for a multi-threshold T if there exists $(t_1, t_2) \in T$ such that $t_P \leq t_1$ and $t_A \leq t_2$. For two multi-thresholds T_a, T_b we say that $T_a \leq T_b$ if for all $(t_{a1}, t_{a2}) \in T_a$, it holds that $(t_{a1}, t_{a2}) \leq T_b$.

2.2 Security Properties

Throughout the paper, we study four security properties: *correctness, secrecy, robustness, and fairness*. We denote the corresponding multi-thresholds T_c, T_s, T_r, and T_f. Each property is considered guaranteed if (t_P, t_A) is smaller than the corresponding multi-threshold. These properties are standard analytic tools for protocols security. For a protocol Π:

- *Correctness*: Given the inputs from $P_1, .., P_n$, each party engaged in Π either obtains the correct output or obtains a special output $\perp$.
- *Secrecy*: The adversary cannot learn more information about other parties' inputs and outputs than can be learned from its own inputs and outputs.
- *Robustness*: The adversary cannot deny their output to the honest parties.
- *Fairness*: Either every party obtains its output or nobody does.

[3] In particular, if $a > b$, we have $[a, b] = \emptyset$.

We have $T_r \leq T_c$ and $T_f \leq T_s \leq T_c$ since we cannot define secrecy, fairness or robustness without correctness and secrecy is required by fairness. Note that all the protocols in this work are not robust when there are more than a few (generally 1 or 2) active corruptions. Thus, we do not study robustness of the developed protocols as they do not provide it in most cases. As such, unless explicitly specified, the robustness threshold is $T_r = \{(n, 1)\}$.

2.3 Definitions for Verifiable, Proactive, and Dynamic PSS

Verifiable Secret Sharing (VSS). A VSS scheme enables an (untrusted) dealer to securely share a secret s among the parties in $\mathcal{P}$, such that a set of honest parties can reconstruct s if they reveal their shares to each other.

Definition 1 (Verifiable Secret Sharing [27]). *A (T_s, T_r)-secure Verifiable Secret Sharing (VSS) scheme is a pair of protocols* `Share` *and* `Reconstruct`*, where* `Share` *takes inputs s from the dealer and* `Reconstruct` *outputs s to each party, if the following conditions are fulfilled:*

- Secrecy*: if $(t_P, t_A) \leq T_s$, then in* `Share` *the adversary learns no information about s;*
- Correctness*: After* `Share`*, the dealer is bound to the values s', where $s' = s$ if the dealer is honest. In* `Reconstruct`*, either each honest party outputs s' or all honest parties abort.*
- Robustness*: the adversary cannot abort* `Share`*, and cannot abort* `Reconstruct` *if $(t_P, t_A) \leq T_r$.*

Proactive Secret Sharing (PSS). A PSS scheme is a VSS scheme secure against a mobile adversary, i.e., realizes proactive security. We recall the definition of PSS from [16]. In particular, a PSS scheme is a VSS scheme extended with two additional sub-protocols: `Refresh` and `Recover`. An execution of PSS will be divided into phases. A refresh phase (resp. recovery phase) is the period of time between two consecutive executions of `Refresh` (resp. `Recover`). Furthermore, the period of time between `Share` and the first `Refresh` (resp. `Recover`) is a refresh phase (resp. recovery phase), and similarly for the period of time between the last `Refresh` (resp. `Recover`) and `Reconstruct`.

Definition 2 (Proactive Secret Sharing [16]). *A Proactive Secret Sharing (PSS) scheme consists of four protocols* `Share`*,* `Reconstruct`*,* `Refresh`*, and* `Recover`*.* `Share` *takes inputs s from the dealer and* `Reconstruct` *outputs s' to each party.* `Refresh` *is executed between two consecutive phases σ and $\sigma + 1$ and generates new shares for phase $\sigma + 1$ that encode the same secrets as the shares for phase σ.* `Recover` *allows parties that lost their shares to obtain new shares encoding s with the help of the other honest parties. A (T_s, T_r, T_c)-secure PSS scheme fulfills the following conditions:*

- Termination*: all honest parties complete each execution of* `Share`*,* `Refresh`*,* `Recover`*, and* `Reconstruct`*.*

- Secrecy*: if* $(t_P, t_A) \leq T_s$*, then in* `Share` *the adversary learns no information about* s*. If* $(t_P, t_A) \leq T_s$ *in both phases* σ *and* $\sigma + 1$*, and if* `Refresh` *and* `Recover` *are run between phases* σ *and* $\sigma + 1$*, then the adversary learns no information about* s*.*
- Correctness*: After* `Share`*, the dealer is bound to the values* s'*, where* $s' = s$ *if the dealer is honest. If* $(t_P, t_A) \leq T_c$*, upon completing* `Refresh` *or* `Recover`*, either the shares held by the parties encodes* s *or all honest parties abort. In* `Reconstruct` *either each honest party outputs* s' *or all honest parties abort.*
- Robustness*: the adversary cannot abort* `Share`*, and cannot abort* `Refresh`*,* `Recover`*, and* `Reconstruct` *if* $(t_P, t_A) \leq T_r$*.*

Dynamic Proactive Secret Sharing (DPSS). A DPSS scheme is a PSS scheme extended by a `Redistribute` protocol that enables (secure distributed) transfer of the secret s from one group of participants to another. Our DPSS definition is inspired by a previous one in [3]. The only difference is that we do not combine `Refresh`, `Recover` and `Redistribute` into one phase. We define a redistribute phase analogously to the refresh and recover phases. The refresh phases are denoted by σ, the redistribute phases by ω, $n^{(\omega)}$ is the number of participants at phase ω. The multi-thresholds T_r, T_c, T_s are considered as functions of n (the number of participants). We denote $T_r^{(\omega)}, T_c^{(\omega)}, T_s^{(\omega)}$ the thresholds at phase ω computed from $n^{(\omega)}$.

Definition 3 (Dynamic Proactive Secret Sharing). *A Dynamic Proactive Secret Sharing (DPSS) scheme consists of a PSS constituted of four protocols* `Share`*,* `Reconstruct`*,* `Refresh`*,* `Recover` *according to Definition 2 completed by a* `Redistribute` *protocol.* `Redistribute` *is executed between consecutive redistribute phases* ω *and* $\omega + 1$ *and allows a set of* $n^{(\omega)}$ *participants at phase* ω *to transfer its shares to the set of* $n^{(\omega+1)}$ *participants of phase* $\omega + 1$*. In the following, when we denote* $(t_P, t_A) \leq T_s^{(\omega)}$*, it is implicit that this is true during redistribute phase* ω*. A* (T_s, T_r, T_c)*-secure DPSS scheme fulfills the following conditions:*

- *For any phase* ω*,* `Share`*,* `Reconstruct`*,* `Refresh` *and* `Recover` *is a* $T_s^{(\omega)}, T_r^{(\omega)}, T_c^{(\omega)}$*)-secure PSS under Definition 2.*
- Termination*: all honest parties complete each execution* `Redistribute`*.*
- Secrecy*: if* $(t_P, t_A) \leq T_s^{(\omega)}$ *and* $(t_P, t_A) \leq T_s^{(\omega+1)}$*, the adversary learns no information about* s *during the execution of* `Redistribute` *between phases* ω *and* $\omega + 1$*.*
- Correctness*: After* `Share`*, the dealer is bound to the values* s'*, where* $s' = s$ *if the dealer is honest. If* $(t_P, t_A) \leq T_c^{(\omega)}$*, upon completing* `Redistribute`*, either the shares held by the parties encodes* s *or all honest parties abort.*
- Robustness*: the adversary cannot abort* `Redistribute` *if* $(t_P, t_A) \leq T_r^{(\omega)}$*.*

2.4 Homomorphic Commitments and VSS

To obtain security against malicious adversaries, we use a homomorphic commitment scheme, e.g., Pedersen commitments [32]. We assume that all values

(secrets, polynomial coefficients, commitments) are in $\mathbb{Z}_q$ for a prime q and that a cyclic group $\mathbb{G}$ of order q with two random generators g, h is distributed to the parties. Commitment to a secret s is $C(s,r) = g^s \cdot h^r$ for a random value r. Due to the use of Pedersen commitment scheme, our protocols are computationally secure under the Discrete Logarithm Problem (DLP) hardness assumption.

2.5 Bivariate Polynomials

We rely on bivariate polynomials as a building block in our design of a batched SS scheme for groups with dishonest majorities. We use polynomials of degree d in both x and y variables. Such a polynomial g is uniquely defined by $(d+1)^2$ points $g(x,y)$ with $(x,y) \in X \times Y$ and $|X| = |Y| = d+1$. Indeed, for any (x_0, y_0), the value $g(x_0, y_0)$ can be found by the interpolation of $g(x, y_0)$ for all $x \in X$. The values $g(x, y_0)$ can be interpolated with $g(x,y)$ for all $y \in Y$. In the following, when we say that g is a bivariate polynomial of degree d, it means that g is of degree d in both its variables.

3 Batched PSS for a Static Group with a Dishonest Majority

In this section, we introduce the definition of (Dynamic) Batched Proactive Secret Sharing (BPSS). In order to understand why the batched setting requires new definitions, we first explain the issue arising when using batching in PSS against mixed adversaries in Sect. 3.1. Then, we introduce the definition of a ℓ-Batch ℓ'-Gradual Secret Sharing in Sect. 3.2.

3.1 The Issue with the Number of Shared Secrets

Recall the naive version of Shamir's (t,n)-secret sharing [34] for $t < n$: a secret $s \in \mathbb{F}$ is stored in the constant coefficient $f(0) \coloneqq s$ of a polynomial $f \in \mathbb{P}_t$. Each party P_r for $r \in [n]$ will receive $f(\alpha_r)$ where the α_j's are (public) distinct nonzero elements and reconstruction is performed by interpolating of the value in 0 using $t+1$ evaluations of f.

The extension the above secret sharing scheme to handle batching is a well-known construction [23]: to share ℓ secrets $s_1, \ldots, s_\ell$, sample a polynomial $f \in \mathbb{P}_{t+\ell-1}$ such that $f(i) = s_i$ and set $\alpha_r \notin [\ell]$. However, now one must ensure that $t + \ell - 1 < n$ so that $(s_1, \ldots, s_\ell)$ remains information-theoretically hidden given up to t evaluations of f in the α_r's; i.e., there is a linear dependency between the number of shared batched secrets and the bound on the tolerated corruption threshold (with respect to n).

Now, let us recall the core idea from [27] to design a fair secret sharing scheme against mixed adversaries. We consider Shamir's secret sharing extended with homomorphic commitments in order to provide verifiability [31]. Now, during the reconstruction step, all correct parties broadcast their shares, and secrecy is given up against all subsets at one. Therefore, the reconstruction protocol does not

achieve fairness (that is, every party obtains its output or nobody does). In order to achieve fairness and handle mixed adversaries, Hirt et al. [27] propose to first split the secret into additive summands, i.e., $s = s^{(1)} + \cdots + s^{(d)}$, with $d = n - 1$ and then use Shamir's (i, n)-secret sharing on $s^{(i)} = f_i(0)$ for all $i \in [d]$. Next, P_r for $r \in [n]$ receives as share the tuple $(f_1(\alpha_r), \ldots, f_d(\alpha_r))$. Reconstruction then recovers each of the $s^{(i)}$ for i from $d = n - 1$ to 1 sequentially. If there is a violation of fairness at any step, i.e., an $s^{(i)}$ cannot be reconstructed, the protocol `aborts`. A mixed adversary cannot abort before the degree $i_0 = t_P$ (for $i < t_P$ the adversary already knows all the values $f_i(0)$). In this case, to preserve fairness the honest parties need to be able to recover all the remaining values $f_i(0)$. Thus we have $i_0 + 1 \leq n - t_A$. By putting the two constraints together we obtain the bound $(t_P, t_A) \leq (n - k - 1, k)$. Additionally, since $t_A \leq t_P$, we get $k \leq \lceil \frac{n}{2} \rceil - 1$.

Now, assume we want to design a batched secret sharing scheme against mixed adversaries. Combining the above arguments prevents a mixed adversary from aborting before the degree $i_0 = t_P + \ell - 1$ and therefore we obtain the bound $(t_P, t_A) \leq (n - k - \ell, k)$. In particular, this implies that as soon as one batches $\ell \geq n/2$ secrets, achieving security with a dishonest majority is not attainable.

To overcome this issue, we introduce a notion of ℓ-Batch ℓ'-Gradual Secret Sharing against mixed adversaries with bound $(t_P, t_A) \leq (n - k - \ell', k)$ in Sect. 3.2; and then similarly to [16], it is easy to extend the latter primitive to define a Batched PSS against mixed adversaries. In Sect. 4, we will instantiate such a primitive for $\ell \leq n - 2$ and $\ell' = \lfloor \sqrt{\ell} \rfloor$ by revisiting the idea of secret sharing using bivariate polynomials (e.g., [35]).

3.2 Batched Gradual Secret Sharing Against Mixed Adversaries

Definition 4 (Gradual VSS [27]). *A (T_s, T_r, T_c)-secure VSS scheme is gradual if the following conditions are fulfilled: If* `Reconstruct` *aborts, each party outputs a non-empty set $B \subset \mathcal{P}_A$ and the adversary cannot obtain information about the secret s if $(t_P, t_A) \leq T_s$ and $t_P \leq n - |B| - 1$.*

Note that this definition is equivalent to fairness when the adversary is bounded by a multi-threshold $T_f = \{(n - k - 1, k) : \ k \in [0, \lceil \frac{n}{2} \rceil - 1] \text{ and } (n - k - 1, k) \leq T_s\}$.

Batched Gradual VSS. We naturally extend Definitions 1 and 4 to batch ℓ secrets. A Batch VSS scheme enables a dealer to share ℓ secrets $s_1, \ldots, s_\ell$ among the parties in $\mathcal{P}$, such that the parties can reconstruct the secrets.

Definition 5 (ℓ-Batch VSS). *A (T_s, T_r)-secure ℓ-Batch VSS scheme is a pair of protocols* `Share` *and* `Reconstruct`*, where* `Share` *takes inputs $s_1, \ldots, s_\ell$ from the dealer and* `Reconstruct` *outputs $s'_1, \ldots, s'_\ell$ to each party, if the following conditions are fulfilled:*

- Secrecy*: if $(t_P, t_A) \leq T_s$, then in* `Share` *the adversary learns no information about $s_1, \ldots, s_\ell$;*

- Correctness*: After* **Share**, *the dealer is bound to the values* $s'_1, \ldots, s'_\ell$, *where* $s'_i = s_i$ *if the dealer is honest. In* **Reconstruct**, *either each honest party outputs* $s'_1, \ldots, s'_\ell$ *or all honest parties abort.*
- Robustness*: the adversary cannot abort* **Share**, *and cannot abort* **Reconstruct** *if* $(t_P, t_A) \leq T_r$.

Definition 6 (ℓ-Batch ℓ'-Gradual VSS). *A* (T_s, T_r, T_c)*-secure* ℓ*-Batch VSS is* ℓ'*-gradual if the following conditions are fulfilled: If* **Reconstruct** *aborts, each party outputs a non-empty set* $B \subset \mathcal{P}_A$ *and the adversary cannot obtain information about the secret* s *if* $(t_P, t_A) \leq T_s$ *and* $t_P \leq n - |B| - \ell'$.

This definition is equivalent to fairness when the adversary is bounded by a multi-threshold $T_f = \{(n-k-\ell', k) : k \in [0, \lceil \frac{n-\ell'}{2} \rceil - 1] \text{ and } (n-k-\ell', k) \leq T_s\}$.

4 Efficient Batched PSS Using Bivariate Polynomials

We defer the definitions of the ideal functionalities for `Share`, `Reconstruct`, `Refresh`, and `Recover`, and their formal simulator-based security proofs, to the full version [20]. In this section, we introduce the protocols and prove in preliminary lemmas the core elements of their security proofs.

In the protocols below, we highlight the critical steps using boxes, as the full protocols include (standard) use of commitments and openings to resist against malicious/mixed adversaries.

4.1 The `Share` Protocol

We assume that $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_\ell \in \mathbb{F}$ are distinct public values. The number ℓ is assumed to be smaller than d, the degree of the bivariate polynomial produced by the sharing. With $d = n-2$ in practice, we have the bound $\ell \leq n-2$ that we mentioned above.

Protocol 1. `Share`

INPUT: Secrets $s_1, \ldots, s_\ell$ held by a dealer $P_\mathcal{D}$.
OUTPUT: Each party P_r holds shares $\{g(\alpha_r, \alpha_{r'})\}_{r' \in [d+1]}$ of the secrets $s_1, \ldots, s_\ell$ (and the corresponding commitments).

1. For $j \in [\ell]$, the dealer samples $f_j \leftarrow \mathbb{P}_d$ such that $f_j(\beta_j) = s_j$.
2. For $r \in [d+1]$, the dealer samples $g(\alpha_r, \cdot) \leftarrow \mathbb{P}_d$ such that $\forall j \in [\ell]$, $g(\alpha_r, \beta_j) = f_j(\alpha_r)$.
   (Note that this implicitly defines a bivariate polynomial g *of degree* d*.)*
3. The dealer interpolates $g(x, y)$ and computes $\{g(\alpha_r, \alpha_{r'})\}_{r' \in [d+1]}$ for all $r \in [n]$.
4. The dealer broadcasts (homomorphic) commitments of the $g(\alpha_r, \alpha_{r'})$ for all $r, r' \in [d+1]$.

5. Each party P_r locally computes commitments for $\{g(\alpha_r, \alpha_{r'})\}_{r' \in [d+1]}$ (using the homomorphic property for $r > d+1$), and the dealer sends the corresponding opening informations to party P_r. Each party broadcasts a complaining bit indicating if an opening received from the dealer is incorrect.
6. For each element $g(\alpha_r, \alpha_{r'})$ for which a complaint was broadcast, the dealer broadcasts its opening. If the opening is correct, P_r accepts the value, otherwise the dealer is disqualified.

Lemma 1. *Let $d \leq n-1$. `Share` is correct and preserves the secrecy of a batch of secrets $s_1, \ldots, s_\ell$ if $(t_P, t_A) \leq \{(d, d)\}$.*

Proof. Correctness follows from the use of homomorphic commitments which allow the parties to verify that the dealer distributed shares for a bivariate polynomial g of degree d in both variables.

For secrecy, we show that the adversary cannot find the values $s_1, \ldots, s_\ell$ when $t_P \leq d$. Without loss of generality, we assume that the adversary controls passively $\{P_1, \ldots, P_{t_P}\}$ and that the dealer is honest. Hence, the adversary knows the values $\{g(\alpha_r, \alpha_{r'})\}_{r' \in [d+1]}$ for $r \in [t_P]$. It can interpolate $g(\alpha_r, \beta_j) = f_j(\alpha_r)$ for all $r \in [t_P]$ and $j \in [\ell]$. For every j, since $t_P \leq d$, $f_j(\beta_j) = s_j$ is information-theoretically hidden. □

Remark 1 (Communication). In Step 4, the dealer broadcasts $(d+1)^2$ commitments, and in Step 5, $(d+1) \cdot n$ messages are sent. With $d = O(n)$, we obtain an amortized communication complexity of $O(n^2)/\ell$.

Remark 2 (Corruption Threshold). Lemma 1 claims security for up to d corruption when we mentioned several time already that our protocol is secure up to $d + 1 - \sqrt{\ell}$. This is because the `Share` protocol in itself tolerates more corruptions. The threshold $d + 1 - \sqrt{\ell}$ is a consequence of the `Recover` protocol, as is explained below.

4.2 The `Recover` Protocol

The `Recover` protocol enables a set of $d+1$ parties $\{P_1, \ldots, P_{d+1}\}$ to send to a recovering party P_{r_C} its shares $(g\{\alpha_{r_C}, \alpha_{r'})\}_{r' \in [d+1]}$. In [16], to perform the recovery of one value $f(\alpha_{r_C})$, each participant P_r generates one blinding polynomial f_r verifying $f(\alpha_{r_C}) = 0$ and share it among the other participants so that P_{r_C} can receive $f(\alpha_r) + \sum_{u=1}^{n} f_u(\alpha_r)$ for $r \in [n]$ and interpolate $f(\alpha_{r_C})$. This is inefficient as each value $f(\alpha_r)$ requires $O(n)$ communication to be blinded. In our secret sharing, each participant P_r have a polynomial $g(\alpha_r, \cdot)$. Just like in [16], our `Recover` protocol requires each P_r to generate one polynomial f_r verifying $f_r(\alpha_{r_C}) = 0$ and share it to the other. The number of blinding polynomials remains the same, but the size of the sharing has been multiplied by a factor $O(n)$, it yields an optimal $O(1)$ communication complexity per value. Yet, it will be enough to blind the batch of ℓ secrets when the corruption threshold is decreased to $d + 1 - \sqrt{\ell}$. Indeed, P_{r_C} is going tor receive the values

$g(\alpha_r, \alpha_{r'}) + f_{r'}(\alpha_r)$ from each of the P_r for $r' \in [d+1]$. When $P_{r'}$ and P_{r_C} are corrupted, the adversary will be able to learn the values $g(\alpha_r, \alpha_{r'})$ for $r \in [d+1]$ that were unknown to the adversary prior to `Recover`. However, when both P_r and $P_{r'}$ are honest, the value $g(\alpha_r, \alpha_{r'})$ is blinded by $f_{r'}(\alpha_r)$. Therefor, the security of the ℓ secrets is going to be protected by the $(d+1-t_P)^2$ values corresponding to pairs $(P_r, P_{r'})$ of honest participants in $\mathcal{P}^2$. That yields the bound $t_P \leq d+1-\sqrt{\ell}$. The formal security analysis of `Recover` is provided in the full version [20].

Overall, our `Recover` protocol consists of the following steps:

(a) First, the set of parties jointly generate random univariate polynomials $f_1, \ldots, f_{d+1}$ of degree d that evaluates to 0 in α_{r_C}.
(b) Then, every party uses its shares of $f_{r'}$'s to randomize its shares $g(\alpha_r, \alpha_{r'})$ so that P_{r_c} can interpolate $g(\alpha_{r_C}, \alpha_{r'})$ for $r' \in [d+1]$.

Protocol 2. `Recover`

INPUT: A set $\mathcal{P} = \{P_1, \ldots, P_{d+1}\}$ with respective shares $\{g(\alpha_r, \alpha_{r'})\}_{r' \in [d+1]}$ and a recovering party P_{r_C}.
OUTPUT: Each party P_r for $r \in [d+1] \cup \{r_C\}$ obtains $\{g'(\alpha_r, \alpha_{r'})\}_{r' \in [d+1]}$, where $g'(\beta_j, \beta_j) = g(\beta_j, \beta_j)$ for all $j \in [\ell]$.

1. For $r \in [d+1]$, P_r broadcasts the commitments to $\{g(\alpha_r, \alpha_{r'})\}_{r' \in [d+1]}$. Each broadcast commitment consistency is locally verified; if consistency fails, P_r broadcasts a complaining bit and the protocol aborts.
2. For $r \in [d+1]$, P_r samples $f_r \leftarrow \mathbb{P}_d$ such that $f_r(\alpha_{r_C}) = 0$, then broadcasts commitments of $f_r(\alpha_{r'})$ for all $r' \in [d+1]$, and then sends an opening to the commitment of $f_r(\alpha_{r'})$ to each $P_{r'}$.
3. Each party verifies that $f_{r'}(\alpha_{r_C})$ opens to 0 for every $r' \in [d+1]$. When the opening fails, $P_{r'}$ is disqualified and added to the set of corrupted parties B, and the protocol aborts and each party outputs B.
4. For $r \in [d+1]$, P_r locally computes $f_{r'}(\alpha_r), r' \in [d+1]$ and broadcasts a complaining bit indicating if the opening is correct. For each share $f_{r'}(\alpha_r)$, for which an irregularity was reported, $P_{r'}$ broadcasts the opening. If the opening is correct, P_r accepts the value, otherwise $P_{r'}$ is disqualified and added to the set of corrupted parties B. The protocols aborts and each party outputs B.
5. For $r \in [d+1]$, P_r sends to P_{r_C} openings to the values $g(\alpha_r, \alpha_{r'}) + f_{r'}(\alpha_r)$ for all $r' \in [d+1]$. P_{r_C} is able to compute locally a commitment to the values $g(\alpha_r, \alpha_{r'}) + f_{r'}(\alpha_r)$ and for each r' broadcasts a bit indicating if the opening was correct.
6. For each share $g(\alpha_r, \alpha_{r'}) + f_{r'}(\alpha_r)$, for which an irregularity was reported, P_r broadcasts the opening. If the opening is correct, P_{r_C} accepts the

value, otherwise P_r is disqualified and added to the set of corrupted parties B. The protocols aborts and each party outputs B.

7. P_{r_C} locally interpolates $g(\alpha_{r_C}, \alpha_{r'})$ for all $r' \in [d+1]$.

Remark 3 (Communication). In Step 1, $(d+1)^2$ commitments are broadcast. In Step 2, $(d+2)(d+1)$ openings are sent. In Step 5, $(d+1)^2$ openings are sent. With $d = O(n)$, we obtain an amortized communication complexity of $O(n^2)/\ell$.

4.3 The `Reconstruct` Protocol

Recall that gradual verifiable secret sharing was introduced in [27] to capture the notion of a mixed adversary by gradually reducing the number of corrupted parties against which secrecy is guaranteed during reconstruction, and at the same time increasing the number of corrupted parties against which robustness is guaranteed. In particular, in [27] a secret s is split into summands $s = s_1 + \cdots + s_d$ and each s_i is secret shared using a polynomial of degree i. During reconstruction, the protocol aborts at step $n-k$ only if strictly less than $n-k+1$ parties opened their commitments correctly and therefore the number of active parties is lower bounded by k. Now, if the total number of corruptions is less than $n-k$, then the adversary learns nothing, which retains secrecy against adversaries controlling k parties actively out of $n-k$ compromised parties.

Now, let's assume we instead have a sharing of $0 = e_1 + \cdots + e_d$ (as polynomials), where $e_1, \ldots, e_{d-1}$ are bivariate polynomials of degrees $1, \ldots, d-1$ respectively. Then the above protocol can be reproduced with $s_i = e_i(\beta)$ for $i < d$ and $s_d = s + e_d(\beta)$; this is the core idea in the protocol below. The core novelty of the protocol is in how to construct this ladder. We will show that by using (i) some fixed public values $\lambda_1, \ldots, \lambda_d$ such that $\sum_{i=1}^{d} \lambda_i = 0$ and (ii) the `Recover` above to share freshly generated polynomials, gradually constructing such a ladder is possible. The key idea is the following: at each step from $i = d$ to $i = 2$, the current bivariate polynomial of degree i is blinded by a random bivariate polynomial of degree $i-1$ *generated by a subset of size i of the parties and recovered with a $i+1$-th party using* `Recover`. All the blinding polynomials e_i will be constructed so that $e_1 + \cdots + e_d = \left(\sum_{k=1}^{d} \lambda_k\right) \cdot Q$, at the end of the protocol for Q a random bivariate polynomial, so that $\sum_{k=1}^{d} \lambda_k = 0$ can eventually be factored out. Note that it does not harm the security to take public λ_i values. Indeed, the security requires that each of the s_i appears uniformly random (up to s_1 that depends on s and the previous s_i). The way that each g_i is constructed from the Q_i polynomials that are random polynomials ensures this property.

The `Reconstruct` protocol is described in Protocol 3, and its correctness and security proofs can be found in the full version [20].

Protocol 3. `Reconstruct`

INPUT: A set $\mathcal{P} = \{P_1, \ldots, P_n\}$ with respective shares $\{g(\alpha_r, \alpha_{r'})\}_{r' \in [d+1]}$. A (public) set of nonzero values $(\lambda_k)_{1 \le k \le d}$ such that $\lambda_1 + \cdots + \lambda_d = 0$ and

$\lambda_1 + \ldots + \lambda_i \neq 0$ for all $i < d$.
OUTPUT: Values $s_j = g(\beta_j, \beta_j)$ for $j \in [\ell]$ to all parties in $\mathcal{P}$.

1. *Initialization*: Set $B = \emptyset$,$i = d$ and the number of remaining parties as $N = n$. Each party in $\mathcal{P}$ sets locally $s_j = 0$ for all $j \in [\ell]$.
2. *First step* $(i = d)$:
   (a) Without loss of generality, assume $\mathcal{P} = \{P_1, \ldots, P_N\}$.
   For $r \in [d]$, P_r samples $Q_{d-1}(\alpha_r, \cdot) \leftarrow \mathbb{P}_{d-1}$ and broadcast commitments to $\{Q_{d-1}(\alpha_r, \alpha_{r'})\}_{r' \in [d]}$.
   Note that this implicitly defines Q_{d-1} a random bivariate polynomial of degree $d-1$.
   (b) Using `Recover`, $P_1, \ldots, P_d$ reveal $\{Q_{d-1}(\alpha_{d+1}, \alpha_{r'})\}_{r' \in [d+1]}$ to P_{d+1}. If `Recover` aborts with output B', sets $B = B \cup B'$, $N = N - |B'|$ and $\mathcal{P} = \mathcal{P} \setminus B'$. If $N > d$, go to step (a), otherwise the protocol aborts and outputs B.
   (c) Denote $g_d = g + \lambda_d Q_{d-1}$. For $r \in [d+1]$, P_r locally updates their shares to $\{g_d(\alpha_r, \alpha_{r'})\}_{r' \in [d+1]}$ using the $Q_{d-1}(\alpha_r, \alpha_{r'})$'s, and broadcasts commitments thereof.
3. *Gradual Reconstruction*: While $i \geq 2$:
   (a) Wlog, assume $\mathcal{P} = \{P_1, \ldots, P_N\}$. For $r \in [i+1]$, P_r broadcasts openings to $\{g_i(\alpha_r, \alpha_{r'})\}_{r' \in [i+1]}$, and all parties locally verify the openings. Let B' denote the parties with incorrect openings. Each party sets $B = B \cup B'$, $N = N - |B'|$ and $\mathcal{P} = \mathcal{P} \setminus B'$. If $N > i$, go the step (b), otherwise the protocol aborts and outputs B.
   (b) For $r \in [i+1, N]$, P_r interpolates its shares $\{g_i(\alpha_r, \alpha_{r'})\}_{r' \in [i+1]}$. Then, computes the values $\{Q_{i-1}(\alpha_r, \alpha_{r'})\}_{r' \in [i]}$.
   Note that we have the invariant $g_i + \cdots + g_d = g + (\lambda_d + \cdots + \lambda_i) Q_{i-1}$.
   (c) All parties interpolate g_i and update $s_j \leftarrow s_j + g_i(\beta_j, \beta_j)$.
   Set $i \leftarrow i - 1$.
   (d) If $i = 1$, sets $Q_0 = 0$ and go to Step (f).
   Else, for $r \in [i]$, P_r samples $Q_{i-1}(\alpha_r, \cdot) \leftarrow \mathbb{P}_{i-1}$ and broadcast commitments to $\{Q_{i-1}(\alpha_r, \alpha_{r'})\}_{r' \in [i+1]}$.
   Note that this implicitly defines Q_{i-1} a random bivariate polynomial of degree $i-1$.
   (e) Using `Recover`, $P_1, \ldots, P_i$ enable P_{i+1} to obtain evaluations of $\{Q_{i-1}(\alpha_r, \alpha_{r'})\}_{r' \in [i+1]}$. If `Recover` aborts with output B', sets $B = B \cup B'$, $N = N - |B'|$ and $\mathcal{P} = \mathcal{P} \setminus B'$. If $N > i$, go to step (d), otherwise the protocol aborts and outputs B.

(f) Denote $\boxed{g_i = \lambda_i Q_i + \left(\sum_{k=1}^{i-1} \lambda_k\right) \cdot (Q_i - Q_{i-1})}$.
For $r \in [i+1]$, P_r $\boxed{\text{locally updates}}$ its shares to $\boxed{\{g_i(\alpha_r, \alpha_{r'})\}_{r' \in [i+1]}}$ and broadcast commitments to these values.

4. *Last Step* ($i = 1$):
Wlog, assume $\mathcal{P} = \{P_1, \ldots, P_N\}$. Each party $\boxed{P_r \in \mathcal{P} \text{ broadcasts}}$ openings to $\boxed{g_1(\alpha_r, \alpha_1)}$ and $\boxed{g_1(\alpha_r, \alpha_2)}$. If there are at least 2 correct set of openings, all parties $\boxed{\text{compute } g_1(\beta_j, \beta_j)}$ for all $j \in [\ell]$ and set $\boxed{s_j \leftarrow s_j + g_1(\beta_j, \beta_j)}$; otherwise the protocol aborts.

Remark 4. We reiterate that we have the invariant $\sum_{k=i}^{d} g_k = g + \left(\sum_{k=i}^{d} \lambda_k\right) \cdot Q_{i-1}$, for all $i \geq 2$, that comes from the fact that $\sum_{k=1}^{d} \lambda_k = 0$. In particular since $Q_0 = 0$, it holds that $\sum_{k=1}^{d} g_k = g$. Hence, Step 3 (b) and Step 4 yield $s_j = \sum_{i=1}^{d} g_i(\beta_j, \beta_j) = g(\beta_j, \beta_j)$.

Remark 5 (Communication). Note that the `Recover` in Steps 2(b) and 3(e) are ran a maximum of $d + t_A = O(n)$ times total, which yields a communication complexity of $O(n^3/\ell)$. Ignoring the `Recover`, Step 2 requires $O(n^2)$ communication (broadcast of commitments for the new polynomials and new shares). Then, each iteration of the loop is performed in $O(i^2) = O(n^2)$ with $(i+1)^2$ openings in 3(a), $(i-1)^2$ commitments in 3(d) and i^2 commitments in 3(f). Overall, the communication complexity of `Reconstruct` is $O(n^3/\ell)$ for ℓ secrets.

Theorem 1. *The pair of protocols (`Share`, `Reconstruct`) constitutes a (T_s, T_c)-secure (under the DLP assumption) ℓ-Batch $\sqrt{\ell}$-Gradual VSS, as in Definition 6, for $T_s = \{(n-1-\lfloor\sqrt{\ell}\rfloor, n-1-\lfloor\sqrt{\ell}\rfloor)\}$ and $T_c = \{(n, n-1)\}$.*

The proof of Theorem 1 is in the full version of this paper [20].

4.4 The `Refresh` Protocol

Similarly to **Reconstruct**, the `Refresh` protocol uses a blinding polynomial Q to guarantee privacy of the secrets. This blinding polynomial Q needs to verify $Q(\beta_j, \beta_j) = 0$ for $j \in [\ell]$. The easiest way to achieve this property is to take $Q(x, y) = (x - y)R(x, y)$ where R is a random bivariate polynomial of degree $d - 1$. However, this polynomial Q is equal to zero on the entire diagonal (x, x). To obtain the level of secrecy required for `Refresh` we also need to refresh the shares $g(x, x)$ for any $x \notin \{\beta_1, \ldots, \beta_\ell\}$.To solve this issue, inspired by the univariate blinding factor in `Recover`, we blind the other diagonal values by a univariate polynomial that evaluates to 0 in the β_j. More precisely, at the end of the protocol, we constructed g' as $g'(x, y) = g(x, y) + Q(x, y) = g(x, y) + (x - y) \cdot R(x, y) + h(x) \cdot \prod_{j \in \ell}(y - \beta_j)$ where h is a random univariate polynomial in $\mathbb{P}_d$ and R is a random bivariate polynomial.

Concretely, the `Recover` protocol is used to build and share the blinding polynomial in the following manner:

(a) First, a set of d participants generates R a random bivariate polynomial of degree $d-1$ and uses `Recover` to share it with the remaining participants.
(b) Then, every party generates a random univariate polynomial h_r and share it among each other, so that every participant P_r can compute its value $h(\alpha_r) = \sum_{u=1}^{n} h_u(\alpha_r)$
(c) Finally, all parties compute $g'(\alpha_r, \alpha_{r'}) = g(\alpha_r, \alpha_{r'}) + (\alpha_r - \alpha_{r'}) \cdot R(\alpha_r, \alpha_{r'}) + h(\alpha_r) \cdot \prod_{j \in \ell}(\alpha_{r'} - \beta_j)$ from their blinded shares.

`Refresh` is described in Protocol 4 and its correctness and security proofs can be found in the full version [20].

Protocol 4. `Refresh`

INPUT: A set $\mathcal{P} = \{P_1, \ldots, P_n\}$ with respective shares $\{g(\alpha_r, \alpha_{r'})\}_{r' \in [d+1]}$.
OUTPUT: Each party $P_r \in \mathcal{P}$ obtains $\{g'(\alpha_r, \alpha_{r'})\}_{r' \in [d+1]}$, where $g'(\beta_j, \beta_j) = g(\beta_j, \beta_j)$ for all $j \in [\ell]$.

1. For $r \in [d]$, P_r samples $R(\alpha_r, \cdot) \leftarrow \mathbb{P}_{d-1}$ and broadcasts homomorphic commitments to the values $\{R(\alpha_r, \alpha_{r'})\}_{r' \in [d]}$.
   Note that this implicitely defines a bivariate polynomial $R(x, y)$ of degree $d-1$.
2. For $i \in \{d+1, ..., n\}$, $\{P_i\} \cup \{P_1, ..., P_d\}$ perform `Recover` to provide P_i with the shares $R(\alpha_i, \alpha_{r'})$ for $r' \in [d]$.
   Note that the first step of `Recover` is unnecessary since each P_i already knows the homomorphic commitments to R.
3. For $r \in [n]$, P_r samples $h_r \leftarrow \mathbb{P}_d$, and broadcasts commitments to the coefficients of $h_r(\alpha_{r'})$ for all $r' \in [d+1]$. P_r sends to $P_{r'}$ an opening of the commitment to $h_r(\alpha_{r'})$ for all $r' \in [d+1]$.
4. For $r \in [n]$, P_r locally verifies the commitments and for each r' broadcasts a bit indicating if the opening was correct. For every irregularity on $h_{r'}(\alpha_r)$, $P_{r'}$ broadcasts the opening. If the opening is correct, P_r accepts the value, otherwise $P_{r'}$ is disqualified and added to the set of corrupted parties B. The protocols aborts and each party outputs B.
5. For $r \in [n]$, P_r computes $h(\alpha_r) = \sum_{r'=1}^{n} h_{r'}(\alpha_r)$.
6. For $r \in [n]$, for all $r' \in [d+1]$, P_r computes
   $g'(\alpha_r, \alpha_{r'}) = g(\alpha_r, \alpha_{r'}) + (\alpha_r - \alpha_{r'}) \cdot R(\alpha_r, \alpha_{r'}) + h(\alpha_r) \cdot \prod_{j \in [\ell]}(\alpha_{r'} - \beta_j)$.

Remark 6 (Communication). The bottleneck of the communication is during Step 2 when $(n-d)$ `Recover` are performed. In the case of maximum security (when $n-d-1 = O(1)$), the communication complexity is $O(n^2)/\ell$ for ℓ secrets.

Theorem 2. *The four protocols* `Share`, `Reconstruct`, `Refresh`, `Recover` *constitute a* T_s, T_c*-secure (under the DLP assumption)* ℓ*-Batch PSS with multi-threshold* $T_c = \{(n, n-1)\}$ *and* $T_s = \{(n-1-\lfloor\sqrt{\ell}\rfloor, n-1-\lfloor\sqrt{\ell}\rfloor)\}$ *and* $\ell = n-2$.

The proof of Theorem 2 is in the full version of this paper [20].

References

1. Backes, M., Cachin, C., Strobl, R.: Proactive secure message transmission in asynchronous networks. In: Proceedings of the Twenty-Second ACM Symposium on Principles of Distributed Computing, PODC 2003, Boston, Massachusetts, USA, 13–16 July 2003, pp. 223–232 (2003). https://doi.org/10.1145/872035.872069. http://doi.acm.org/10.1145/872035.872069
2. Baron, J., Eldefrawy, K., Lampkins, J., Ostrovsky, R.: How to withstand mobile virus attacks, revisited. In: PODC, pp. 293–302. ACM (2014)
3. Baron, J., Defrawy, K.E., Lampkins, J., Ostrovsky, R.: Communication-optimal proactive secret sharing for dynamic groups. In: Malkin, T., Kolesnikov, V., Lewko, A.B., Polychronakis, M. (eds.) ACNS 2015. LNCS, vol. 9092, pp. 23–41. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-28166-7_2
4. Beerliová-Trubíniová, Z., Hirt, M.: Perfectly-secure MPC with linear communication complexity. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 213–230. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78524-8_13. http://dl.acm.org/citation.cfm?id=1802614.1802632
5. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: STOC, pp. 1–10. ACM (1988)
6. Ben-Sasson, E., Fehr, S., Ostrovsky, R.: Near-linear unconditionally-secure multiparty computation with a dishonest minority. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 663–680. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_39
7. Blakley, G.R.: Safeguarding cryptographic keys. In: Proceedings of AFIPS National Computer Conference, vol. 48, pp. 313–317 (1979)
8. Boneh, D., Gennaro, R., Goldfeder, S.: Using level-1 homomorphic encryption to improve threshold DSA signatures for bitcoin wallet security. In: Lange, T., Dunkelman, O. (eds.) LATINCRYPT 2017. LNCS, vol. 11368, pp. 352–377. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25283-0_19
9. Canetti, R., Herzberg, A.: Maintaining security in the presence of transient faults. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 425–438. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48658-5_38
10. Castro, M., Liskov, B.: Practical Byzantine fault tolerance and proactive recovery. ACM Trans. Comput. Syst. **20**(4), 398–461 (2002)
11. Chaum, D., Crépeau, C., Damgard, I.: Multiparty unconditionally secure protocols. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, STOC 1988, pp. 11–19. ACM, New York (1988). https://doi.org/10.1145/62212.62214. http://doi.acm.org/10.1145/62212.62214
12. Damgård, I., Ishai, Y., Krøigaard, M.: Perfectly secure multiparty computation and the computational overhead of cryptography. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 445–465. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_23

13. Damgård, I., Ishai, Y., Krøigaard, M., Nielsen, J.B., Smith, A.: Scalable multiparty computation with nearly optimal work and resilience. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 241–261. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_14
14. Damgård, I., Nielsen, J.B.: Scalable and unconditionally secure multiparty computation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 572–590. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_32
15. Desmedt, Y., Jajodia, S.: Redistributing secret shares to new access structures and its applications (1997). Technical Report ISSE TR-97-01, George Mason University
16. Dolev, S., Eldefrawy, K., Lampkins, J., Ostrovsky, R., Yung, M.: Proactive secret sharing with a dishonest majority. In: Zikas, V., De Prisco, R. (eds.) SCN 2016. LNCS, vol. 9841, pp. 529–548. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44618-9_28
17. Dolev, S., Garay, J., Gilboa, N., Kolesnikov, V.: Swarming secrets. In: Proceedings of the 47th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2009, pp. 1438–1445. IEEE Press, Piscataway (2009). http://dl.acm.org/citation.cfm?id=1793974.1794220
18. Dolev, S., Garay, J.A., Gilboa, N., Kolesnikov, V.: Secret sharing Krohn-Rhodes: private and perennial distributed computation. In: ICS (2011)
19. Dolev, S., Garay, J.A., Gilboa, N., Yelena Yuditsky, V.K.: Towards efficient private distributed computation on unbounded input streams. J. Math. Cryptol. **9**(2), 79–94 (2015). https://doi.org/10.1515/jmc-2013-0039
20. Eldefrawy, K., Lepoint, T., Leroux, A.: Communication-efficient proactive secret sharing for dynamic groups with dishonest majorities. Cryptology ePrint Archive, Report 2019/1383 (2019). https://eprint.iacr.org/2019/1383
21. Eldefrawy, K., Ostrovsky, R., Park, S., Yung, M.: Proactive secure multiparty computation with a dishonest majority. In: Catalano, D., De Prisco, R. (eds.) SCN 2018. LNCS, vol. 11035, pp. 200–215. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98113-0_11
22. Frankel, Y., Gemmell, P., MacKenzie, P.D., Yung, M.: Optimal resilience proactive public-key cryptosystems. In: 38th Annual Symposium on Foundations of Computer Science, FOCS 1997, Miami Beach, Florida, USA, 19–22 October 1997, pp. 384–393. IEEE Computer Society (1997). https://doi.org/10.1109/SFCS.1997.646127
23. Franklin, M.K., Yung, M.: Communication complexity of secure computation (extended abstract). In: STOC, pp. 699–710 (1992)
24. Gennaro, R., Goldfeder, S.: Fast multiparty threshold ECDSA with fast trustless setup. In: ACM Conference on Computer and Communications Security, pp. 1179–1194. ACM (2018)
25. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: Aho, A.V. (ed.) STOC, pp. 218–229. ACM (1987)
26. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive secret sharing or: how to cope with perpetual leakage. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 339–352. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-44750-4_27
27. Hirt, M., Maurer, U., Lucas, C.: A dynamic tradeoff between active and passive corruptions in secure multi-party computation. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 203–219. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_12

28. Lindell, Y., Nof, A.: Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, pp. 1837–1854. ACM, New York (2018). https://doi.org/10.1145/3243734.3243788. http://doi.acm.org/10.1145/3243734.3243788
29. Lindell, Y., Nof, A.: Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In: ACM Conference on Computer and Communications Security, pp. 1837–1854. ACM (2018)
30. Ostrovsky, R., Yung, M.: How to withstand mobile virus attacks (extended abstract). In: PODC, pp. 51–59. ACM (1991)
31. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_9
32. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In: STOC, pp. 73–85. ACM (1989)
33. Schultz, D.: Mobile proactive secret sharing. Ph.D. thesis, Massachusetts Institute of Technology (2007)
34. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
35. Tassa, T., Dyn, N.: Multipartite secret sharing by bivariate interpolation. J. Cryptol. **22**(2), 227–258 (2009)
36. Wong, T.M., Wang, C., Wing, J.M.: Verifiable secret redistribution for archive system. In: IEEE Security in Storage Workshop, pp. 94–106. IEEE Computer Society (2002)
37. Zhou, L., Schneider, F.B., van Renesse, R.: APSS: proactive secret sharing in asynchronous systems. ACM Trans. Inf. Syst. Secur. **8**(3), 259–286 (2005)