Threshold Cryptography as a Service (in the Multiserver and YOSO Models)

Fabrice Benhamouda, Shai Halevi, Hugo Krawczyk, Alex Miao, and Tal Rabin

ABSTRACT

We consider large deployments of threshold cryptographic services that can run on traditional multi-server settings and, at a much larger scale, in blockchain environments. We present a set of techniques that improve performance enabling executions with large number of servers and high rate of threshold operations. More fundamentally, our techniques enable threshold cryptographic applications to run in the more challenging decentralized permissionless systems, such as contemporary blockchains. In particular, we design and implement a novel threshold solution in the recently introduced YOSO (You Only Speak Once) model. The model builds on ever changing, unpredictable committees that perform ephemeral roles in a way that evades targeting by attackers and enables virtually unlimited scalability in very large networks. When deployed on a blockchain setting our solution allows for the maintenance of system wide keys that can be generated, used and proactivized as needed. The specific techniques build on optimized protocols for multi-secret multi-dealer verifiable secret sharing and their adaptation to the YOSO model. We show the practical feasibility of our solutions by reporting on an end-to-end implementation of a proactive re-sharing protocol in the YOSO model, and showing benchmarks for committees of sizes up to 500 nodes.

1 INTRODUCTION

In threshold cryptography, cryptographic functions are performed by running a protocol among a set of parties (typically servers), in a way that preserves secrecy and integrity as long as "most parties" are honest. A typical example is a system of n servers that produces signatures in a secure way as long as no more than a threshold, t, of the servers is corrupted. Threshold cryptography has been studied intensively for more that 30 years. While deployment in the real world has been limited so far, modern trends such as the outsourcing of computation to the cloud and decentralized applications provide new motivation and opportunities to deploy threshold cryptography.

Efficient threshold cryptography systems for small sets of servers are known, but most systems in the literature scale rather poorly to large deployments. In this work we develop techniques that make very large deployment feasible. We envision a large set of servers that provides cryptographic services to applications by operating cryptographic functions with threshold security. Examples include threshold encryption services to protect valuable near-term and long-term information, signature services to support sensitive applications such as (software) code signing or certification authorities, and many more. We refer to such deployments as *threshold cryptography as a service.* Our primary motivation for looking at large deployment comes from their application to decentralized permissionless systems, such as contemporary blockchains. These systems are open, allowing anyone to join and run a node, creating the potential for the existence of large number of nodes. Thus, the protocols than are executed in these systems must perform well even when large number of nodes wish to participate.

One technique for dealing with such large systems rely on smaller committees for doing the work, that are sub-sampled from the entire population. But these committees must still be large enough so as to ensure that they are "mostly honest" with high probability. Hence, even in this case we may need to handle committees of many hundreds (or even thousands) of parties.

YOSO protocols. Recent years saw a new style of cryptographic protocols, where long-term parties are replaced by stateless ephemeral ones. This protocols are motivated by the general shift towards "serverless computing", as well as by specific blockchain-based architectures that emphasize player replaceability. That style of protocol was recently formalized by Gentry et al. [18] and called YOSO (You Only Speak Once). In this model, long-lived nodes are replaced by ephemeral stateless entities called *roles.*[1] These roles are addressable (i.e. other roles can send them messages), and each role has access to all the messages that were sent to it (or broadcast) in the past. Other than this communication transcript, however, each role begins its life without any state. A role performs some computation, sends (or broadcast) a single batch of messages, and then immediately self destructs.

While this serverless model is very appealing from a system perspective, realizing secure protocol of this type is very challenging. Some early protocols were described, e.g., in [5, 6, 9], and Gentry et al. provided in [18] a general feasibility result, showing that every function can in principle be computing in this way. But obtaining YOSO protocols that are efficient enough to be useful is still a major challenge. And aiming at large committees make this challenge a whole lot harder. As we explain later, adapting standard cryptographic protocols to the YOSO model typically entails at least an $n\times$ increase in complexity, where n is the number of parties.

1.1 Our contributions and techniques

In this work we describe optimized threshold protocols that are efficient enough to support large committees, even in the YOSO model. Our solutions are all built from a basic primitive of *multi-dealer/multi-secret verifiable secret sharing* (that we call *multiVSS* for short). Namely, we show efficient protocols that allow n' dealers, each with m secrets, to share all their secrets among a set of n shareholders.

As a warm-up, we show how to amortize the classical Pedersen VSS protocol [25] to the setting of many secrets that are shared by many dealers, in the traditional (non-YOSO) setting. Very roughly, we consider a 3-dimensional matrix with axis corresponding to dealers, secrets, and shareholders, and perform consistency checks across the multiple dimensions to save on work. This basic protocol is described in Section 3.

[1] An example of a role could be "party #3 in committee #2".

The main technical meat of this work is devoted to adapting that basic protocol to the YOSO model, and using it to implement a protocol for "proactive" refreshing of secrets (as needed in the architecture of Benhamouda et al [5]).

To see some of the issues that are involved, recall that the Pedersen VSS protocol is based on the paradigm of accusations and resolution. Very roughly, it features a dealer than sends shares of its secret to the shareholders, and publishes some associated verification information. The shareholders compare their shares to the verification information, and publish an accusation against the dealer if they don't match. The dealer then respond to accusations by publishing the secret shares that it sent to the accusing shareholders. Finally, every participants can decide whether to accept the published responses, or to disqualify the dealer.

Consider now what happens when we try to port such a protocol to the YOSO model. We need to adapt it so that the dealer only sends a single batch of messages, and shareholders are completely passive and do not send any messages at all. This may seem impossible: How can they shareholders accuse if they cannot speak at all, and how can the dealer respond if it cannot speak a second time? To solve this conundrum, we let the dealer send its messages via two intermediary committees: A *verification committee* that will runs checks and broadcast accusations, and a *response committee* that responds to these accusations.

The response committee (called "future broadcast" in [18]) is somewhat easier: Whenever a dealer D_i sends a message to a verifier V_j, it also uses a simple secret-sharing protocol to share that message among the members of the response committees. If V_j accuses D_i, then the response committee simply publishes the shares of that message for everyone to see. Since it must be the case that either V_j or D_i are bad, then there is no security loss in revealing that message.

The verification committee is harder: A dealer D_i that wants to relay a message M via the verifiers to some shareholder P_ℓ, cannot directly send it to any of the verifiers, since any one of them can be bad. Instead, *it has to share that message among all the verifiers*, and have them reconstruct that message to P_ℓ. (Moreover, that sharing has to be "verifiable enough" so that the verifiers can accuse D_i if the sharing is bad.)

These ideas can be used to "compile" protocols to the YOSO model, but doing so naively is very expensive. Indeed, it seems to require *two extra levels of secret sharing*: every message $D \longrightarrow P$ in the original protocol must be shared among the verifiers, and each of these shares must again be shared among the responders. Hence, trying to compile an n-party protocol in this may increase the complexity by a factor of n^2.

Reducing this overhead is the main technical contribution of this work. On one hand, we show in Section 4 a collection of techniques that can reduce the expansion factor from $n^2\times$ to $n\times$. Then we let the scaling features of our multiVSS protocol really shine: we show that this $n\times$ factor can be expressed as having the dealers share n secrets each (rather than just one), and then use our MultiVSS protocol to do so much more efficiently than running n copies of the original protocol.

The combination of all these techniques allow our protocol to remain feasible even with many hundreds of parties and even in the YOSO model. For example In fact, this makes our protocol a promising avenue, perhaps the first one, for an actual feasible implementation of the architecture from [5].

To demonstrate the practicality of our YOSO resharing protocol, we provide an end to end implementation of it (see the additional materials) and report performance results. It is written in Go and portable C without assembler optimization, using a modified version of libsodium [14] with some new optimizations. With $n = 513$ parties, the protocol takes less than 2 minutes of single-threaded computation per party (when run over an in-memory communication layer). Since the workload per party is embarrassingly parallel, a multi-core version would scale almost linearly in number of cores and would reduce even more the computation time per party. In addition, each party broadcasts less than 100MB of data.

For the non-YOSO MultiVSS protocol, we provide in the appendix some performance results that were obtained from running micro-benchmarks on the various steps. These results show that our amortization techniques provide significant speedup over a naive implementation of the basic Pedersen protocol.

1.2 Related Work

Optimizing VSS and MPC via batch verification has been examined extensively, especially in information-theoretic settings [1–4, 12, 13, 22]. A limitation of this approach is that it requires a larger honest majority (at least $n \geq 3t + 1$). A more serious challenge is to augment these information-theoretic protocols with *public keys* that correspond to the secrets or their shares, as needed in many applications, particularly those that motivate our work. Moreover, porting these protocols to the YOSO setting seems very hard. While information-theoretic YOSO protocols are theoretically feasible [18], they typically feature complexity of $O(n^6)$ or more, which is too expensive to be usable in practice.

Some recent works on efficient computational (non-YOSO) VSS include, for example, [7, 23]. The former describe a very efficient VSS, but *where the shareholders can only recover $s \cdot G$ in the group*, rather than the secret s itself. This may be enough for some applications, but is too restrictive for others (such as threshold signatures or decryption). The latter can be used for more settings, but it is less efficient.

A very relevant recent work is the VSS protocol of Gentry et al. [19]. That work is motivated similarly to ours, namely, they aim at feasible threshold protocols for the YOSO model. But it features a different solution approach: Rather than adapt an interactive protocol to the YOSO model, they describe a completely non-interactive VSS protocol. Namely, the dealer(s) just broadcast encrypted shares to the shareholders, and provide a direct NIZK proofs that these shares are valid. They use a combination of lattice-based batched encryption scheme and DL-based proof systems to obtain rather efficient NIZKs. While potentially feasible even for very large committees (and achieving non-interactive VSS, which we do not), their protocol is a lot slower than ours. For example, running their protocol with $n = 512$ parties, verifying *each NIZK proof* reportedly takes 17.4 seconds. This means that each shareholder must work $17.4 \cdot 512 \approx 8900$ seconds (about 2.5 hours) to verify all the proofs. In contrast, our protocol with 513 parties would requires about two minutes to complete the entire verifiable sharing.

1.3 Organization

We go briefly over some background material in Section 2, and then describe our "warm-up" non-YOSO amortized VSS protocol in Section 3. Our YOSO protocols are described in Section 4, and we show some initial implementation results from running them in Section 5. All the proofs are deferred to the appendix.

2 NOTATIONS AND BACKGROUND

Notation. For an integer n, we denote $[n] = \{1, 2, \dots n\}$ and $[[n]] = \{0, 1, 2, \dots, n\}$. We consider a hard-discrete-logarithm group $\mathbb{G}$ of prime order q, written additively, with a designated generator $G \in \mathbb{G}$. The number of shareholders is usually denoted n, and the bound on the number of faulty parties is t. Secrets are usually denoted by lowercase $s \in \mathbb{Z}_q$, and we use uppercase $S \in \mathbb{G}$ to denote the corresponding group elements, $S = sG$. We often use σ to denote shares and $Z = \sigma G$. We extend the group additive notations to vectors as follows:

- For a vector $\vec{z} = (z_1, \dots, z_n) \in \mathbb{Z}_q^n$ and an element $X \in \mathbb{G}$, we denote $\vec{z}X = (z_1X, \dots, z_nX) \in \mathbb{G}^n$. Similarly for $z \in \mathbb{Z}_q$ and $\vec{X} = (X_1, \dots, X_n) \in \mathbb{G}^n$, we denote $z\vec{X} = (zX_1, \dots, zX_n) \in \mathbb{G}^n$.
- For same-dimension vectors $\vec{z} = (z_1, \dots, z_n) \in \mathbb{Z}_q^n$ and $\vec{X} = (X_1, \dots, X_n) \in \mathbb{G}^n$ their pointwise product is $\vec{z} \odot \vec{X} = (z_1X_1, \dots, z_nX_n) \in \mathbb{G}^n$, and their inner product is $\vec{z} \bullet \vec{X} = \sum_{i=1}^n z_iX_i \in \mathbb{G}$.

These notations extend in the natural way also to matrix-vector products.

2.1 Pedersen Commitments

Pedersen commitments [24] rely on a setup phase in which two random generator $G_0, G_1 \in \mathbb{G}$ are set with the discrete logarithm between them remaining unknown. To commit to a secret value $z \in \mathbb{Z}_q$, the committer chooses a uniform $r \in \mathbb{Z}_q$ and outputs the group element $C = rG_0 + zG_1$. To open, the committer reveals z and r.

This is extended to multi-valued commitments to m-vectors of secret elements as follows: The setup is extended to emit $m + 1$ random generators with unknown pairwise discrete logarithm, $G_0, G_1, \dots, G_m \in \mathbb{G}$. To commit to $(z_1, \dots, z_m) \in \mathbb{Z}_q^m$, the committer chooses a random scalar $z_0 \in \mathbb{Z}_q$ and outputs the commitment element $C = \sum_{\ell=0}^m z_\ell G_\ell$. To open, the committer reveals all the z_ℓ's, $\ell = 0, 1, \dots, m$. The following are well-known properties of this scheme.

- Perfectly hiding: Given an element $C \in \mathbb{G}$ and $z_1, \dots, z_n \in \mathbb{Z}_q$, there is exactly one value $z_0 \in \mathbb{Z}_q$ for which $C = \sum_{\ell=0}^m z_\ell G_\ell$.
- Computationally binding: If the discrete logarithm problem is hard in $\mathbb{G}$ then it is infeasible to find two different vectors $(z_0, z_1, \dots, z_m), (z_0', z_1', \dots, z_m') \in \mathbb{Z}_q^m$ such that $\sum_{\ell=0}^m z_\ell G_\ell = \sum_{\ell=0}^m z_\ell' G_\ell$.
- Linearity: Given vectors $\vec{z}, \vec{z}' \in \mathbb{Z}_q^m$ and their Pedersen commitments C, C' with randomness z_0, z_0', the element $C + C' \in \mathbb{G}$ is a valid commitment to $\vec{z} + \vec{z}' \in \mathbb{Z}_q^m$ with randomness $z_0 + z_0' \in \mathbb{Z}_q$.

2.2 Vector Commitments

Vector commitment [8] is a commitment scheme to a vector that allows the committer to open individual entries of the vector without opening them all. A vector commitment solution is committing to each entry individually. But more efficient solutions are known, using Merkle trees or accumulators. Our YOSO protocol from Section 4 uses vector-commitment, but that component has only a negligible effect on the overall complexity of the overall protocol. Hence in our implementation we used the trivial solution above.

2.3 Linearity Testing

Our protocols need parties to verify that vectors of croup elements $\vec{C} = (C_1, \dots, C_n)$ belong to some known linear space "in the exponent". (In particular, we will need to check that $C_j = f(j)G$ for some low-degree polynomial f.) This condition can be tested by checking that $[h]\vec{C} = \vec{0}$ where $[h]$ is the parity-check matrix of that linear space over $\mathbb{Z}_q$. But checking this directly is expensive, as it requires as many scalar-point multiplications as there are entries in h. Instead, the verifier can choose a random vector $\vec{e}$ over $\mathbb{Z}_q$, compute $\vec{u} = \vec{e}[h]$, then check the inner product $\vec{v} \bullet \vec{C} = 0$ using a single n-multi-scalar-point multiplications.

2.4 NIZK-POK for Discrete Logarithm

Schnorr-type proofs of knowledge for discrete logarithm are extremely well-studied and useful primitives for cryptographic protocols, see for example [10, 11] and references within.

For our purposes, we will be proving knowledge of (possibly multi-valued) commitments that satisfy some linear relations. Abstractly, the statements to be proven include public generators $\vec{G} \in \mathbb{G}^n$ and some other elements $\vec{Z} \in \mathbb{G}^m$ (for some m, n), with the prover claiming knowledge of a representation of $\vec{Z}$ in the bases $\vec{G}$ that may also satisfy some public linear relations over $\mathbb{Z}_q$.

Fig. 1 describes two simple cases that will be used in our protocols (in the multi-value setting). Given $G_1, \dots, G_m, Z_1, \dots, Z_m \in \mathbb{G}$, these protocols prove knowledge of $z_1, \dots, z_m \in \mathbb{Z}_q$ such that $Z_\ell = z_\ell G_\ell, \ell \in [m]$. In the second protocol, the parties also have $H_1, \dots, H_m, Y$ and the prover claims that the same z_ℓ's also satisfy $\sum_\ell z_\ell H_\ell = Y$. Note that for large enough values of m, these proofs can be made more efficient by using a few large m-multi-scalar-point multiplications rather than similar number of individual scalar-point multiplications. (This will be more efficient in practice even if we need to replace $2m$ scalar-point multiplications by three m-multi-scalar-point multiplications, or $3m$ scalar-point multiplications by four m-multi-scalar-point multiplications.)

2.5 Pedersen Verifiable Secret Sharing

Recall that a (t, n) Verifiable Secret Sharing (VSS) is a multi-party protocol with two phases: Dealing and Reconstruction. The dealing phase involves a dealer D_i and n shareholders $P_1, \dots, P_n$, and the reconstruction phase involves $t + 1$ (or more) of the shareholders. [2] At the end of the dealing phase, each shareholder either decides to *disqualify* the dealer, or keeps some secret state called a *share of the*

[2] We use an index i for the dealer, to conform to the notation we have later when there are multiple dealers.

Parameters: generators $G_1, \dots, G_m$ (, $H_1, \dots, H_m$)
Public input: elements $Z_1, \dots, Z_m$ (, Y)
Prover input: scalars $z_1, \dots, z_m$

(1) Prover chooses at random $r_1, \dots, r_m \in \mathbb{Z}_q$, sends to verifier $R_\ell = r_\ell G_\ell$ (and $S_\ell = r_\ell H_\ell$).
(2) Verifier sends to prover a random $c \in \mathbb{Z}_q$.
(3) Prover responds with $z'_\ell = r_\ell + c \cdot z_\ell, \ell \in [m]$.
(4) Verifier picks uniformly at random $e_1, \dots, e_\ell \in \mathbb{Z}_q$. Then it checks that

$$\sum_\ell ce_\ell Z_\ell + \sum_\ell e_\ell R_\ell \big(+ cY + \sum_\ell S_\ell \big) = \sum_\ell z'_\ell e_\ell G_\ell \big(+ \sum_\ell z'_\ell H_\ell \big).$$

(Actually, $c, e_1, \dots, e_\ell$ can be taken in $[2^\lambda]$ with λ the security parameter rather than uniform in $\mathbb{Z}_q$.)

Figure 1: Σ-protocols to prove knowledge of discrete-logarithm in a multi-valued setting, $Z_\ell = z_\ell G_\ell$. With the parenthesized parts, it also proves that $\sum z_\ell H_\ell = Y$ for the same z_ℓ's. These proofs can be made non-interactive via the Fiat-Shamir transformation.

global secret. The reconstruction phase then have the participating shareholders use their shares to recover the global secret.

A (t, n)-VSS ensures that all the honest parties agree on whether or not the dealer is disqualified. If the honest shareholders do not disqualify the dealer, then here is a unique secret value $s \in \mathbb{Z}_q$ that will be reconstructed during the reconstruction phase, regardless of the subset of honest shareholders that tries to reconstruct it (or the adversary's actions during reconstruction). Moreover, if the dealer was honest then it will not be disqualified, and the global shared secret is indeed the secret input of the dealer. Finally, when the dealer is honest, then at the end of the Dealing phase, no set of t or less parties learns anything about the secret s. These guarantees only hold assuming a large enough honest majority among the shareholders. Typically we have $n \geq 2t + 1$ and at most t dishonest shareholders.

Pedersen VSS [25] uses (t, n)-Shamir sharing, with verification of shares using the linear homomorphism of the commitments. The protocol uses a broadcast channel and private channels from the dealer to all the shareholders. It is described in Fig. 2.

2.6 Multi-Dealer Pedersen VSS with Public Verification Values

Most applications of VSS have multiple parties playing dealers, and often the target shared secrets are linear combinations of the original secrets that were shared by these dealers. The linear combination is public, and can depend on the set of qualified dealers. (at least to the extent that the secrets of disqualified dealers are not included in it.)

This multi-dealer variant provides the VSS security guarantees for each of the target secrets, while ensuring that all the honest shareholders agree on the set of qualified dealers.

Such applications usually require also the creation of public values that are associated with the target shared secrets. For example, the application may use the target shared secrets as encryption/signature keys, and require parties to publicly output the corresponding public keys. The Pedersen-VSS protocol is therefore augmented by another protocol to compute (and verify) these public values. In this work, we consider the case where the application makes public all the elements $Z = zG \in \mathbb{G}$ for all the target secrets $z \in \mathbb{Z}_q$.

Sometimes, this second protocol uses (and even leaks information about) internal state of the Pedersen VSS protocol, beyond just the sharing of the target secrets. While this leakage may be harmless for specific applications, it makes it hard to reason about the augmented protocol on its own, separately from the applications that may use it. In this work we therefore settle for analyzing this "augmented VSS" in the context of our applications.

3 AMORTIZED VSS

An important contribution of our work is the ability to amortize the sharing of many secrets with only a minor increase in complexity. We start by describing the approach for amortizing multiple Pedersen VSS protocols for one dealer, then show how we apply this approach in the multi-dealer case (which is what most applications require).

3.1 Single-Dealer, Multi-Secret, Pedersen VSS

Consider a single dealer D_i that holds m secrets $s_{i\ell}, \ell \in [m]$, and wants to verifiably share all of them *to the same set of n shareholders*, $P_1, \dots, P_n$. We show how to accomplish this with better efficiency than running the VSS protocol m times. The idea in a nutshell it to start with multiple Pedersen VSS protocol from Fig. 2, and combine all the shares of one shareholder using a multi-Pedersen commitment as in Section 2.1. This in essence is the batching techniques for secret sharing [4], computed in the exponent.

In more detail, the dealer generates an independent (t, n) Shamir-sharing for each of the secrets $s_{i,\ell}$, and in addition for another random secret s_{i0}. Then consider the share matrix $\mathbb{A}_i \in \mathbb{Z}_q^{(m+1)\times n}$, where $\mathbb{A}_i[\ell, j] = f_{i\ell}(j)$ is the share of party j in the sharing of $s_{i\ell}$:

$$\mathbb{A}_i = \begin{pmatrix} \sigma_{i10} & & \sigma_{in0} \\ \sigma_{i11} & & \sigma_{in1} \\ & \ddots & \\ \sigma_{i1m} & & \sigma_{inm} \end{pmatrix} \quad \begin{matrix} \sigma_{ij0} = \sum_{k=0}^{t} a_{ik0} j^k \ (a_{i00} = s_{i0}) \\ \sigma_{ij1} = \sum_{k=0}^{t} a_{ik1} j^k \ (a_{i01} = s_{i1}) \\ \vdots \\ \sigma_{ijm} = \sum_{k=0}^{t} a_{ikm} j^k \ (a_{i0m} = s_{im}) \end{matrix}$$

The dealer generates and broadcasts a Pedersen multi-values commitment to each column of the matrix, using σ_{ij0} as randomness. Namely, $C_{ij} = \sum_{\ell=0}^{m} \sigma_{ij\ell} G_\ell \in \mathbb{G}$, where the G's are public random generators. The dealer also sends over a private channel to each shareholder P_j its shares for all the secrets, $\sigma_{ij\ell} = f_{i\ell}(j)$, $\ell = 0, 1, \dots, m$.

Each shareholder P_j compares its own shares to the public commitments C_{ij}, by checking that indeed $C_{ij} = \sum_{\ell=0}^{m} \sigma_{ij\ell} G_\ell$. In addition, everyone verifier that the check values $C_{i1}, \dots, C_{in}$ lie on a degree-t polynomial, using the linearity test from Section 2.3.

Parameters: Integers t, n, $n \geq 2t+1$. Group $\mathbb{G}$ of order q and two generators $G_0, G_1 \in \mathbb{G}$.
Inputs: Dealer D_i has private input $s_i \in \mathbb{Z}_q$.

(1) **Share distribution.** D_i chooses two random polynomials $f_i(\cdot), r_i(\cdot)$ over $\mathbb{Z}_q$ of degree t s.t. $f_i(0) = s_i$:

$$f_i(z) = a_{i0} + a_{i1}z + \cdots + a_{it}z^t,\ \ r_i(z) = b_{i0} + b_{i1}z + \cdots + b_{it}z^t,$$

with $a_{i0} = s_i$ and the other a_{ik}, b_{ik} chosen at random in $\mathbb{Z}_q$.
- D_i broadcasts $C_{ik} = a_{ik}G_1 + b_{ik}G_0$ for $k = 0, \ldots, t$.
- D_i computes the shares $\sigma_{ij} = f_i(j), \sigma'_{ij} = r_i(j)$ (over $\mathbb{Z}_q$) for $j = 1, \ldots, n$ and sends $\sigma_{ij}, \sigma'_{ij}$ to P_j.

(2) **Accusations.** Each shareholder P_j verifies the shares from the dealer by checking that

$$\sigma_{ij}G_1 + \sigma'_{ij}G_0 = \textstyle\sum_{k=0}^{t} j^k C_{ik} \text{ (over } \mathbb{G}). \tag{1}$$

If the check fails for some $i \in [n']$, then P_j broadcasts an accusation against D_i.

(3) **Response.** For every P_j that accused D_i, the dealer broadcasts the values $\sigma_{ij}, \sigma'_{ij}$ that satisfy Eq. (1).

(4) **Disqualification.** Each shareholder P_j marks the dealer as *disqualified* if either:
- it received more than t accusations in Step 2, or
- it failed to answer an accusation with values that satisfy Eq. (1).

(Note that disqualification is a deterministic function of the parameters and broadcast channel.)

If the dealer is not disqualified, then each P_j that sent an accusation replaces its shares by the values $\sigma_{ij}, \sigma'_{ij}$ that were broadcast in Step 3. (Note that these values satisfy Eq. (1).)

Reconstruction. If the dealer was not disqualified, then shareholders can reconstruct the secret s_i by pooling their shares $\sigma_{ij}, \sigma'_{ij}$, which are checked using Eq. (1). Any $t+1$ shares σ_{ij} that pass verification can be used to interpolate s_i.

Figure 2: The Pedersen VSS Protocol.

If both D_i and P_j are honest then this verification will pass. To see that, let $e_\ell = DL_G(G_\ell)$ (for some arbitrary generator G), then

$$C_j = \sum_{\ell=0}^{m} \sigma_{ij\ell} \cdot G_\ell = \sum_{\ell=0}^{m} f_{i\ell}(j) \cdot e_\ell G = F_i(j)G,$$

where $F_i(\cdot) = \sum_\ell e_\ell \cdot f_{i\ell}(\cdot)$ is a degree-t polynomial.

If verification fails, P_j broadcasts a complaint against D_i. This is resolved as usual by D_i broadcasting all the shares $\sigma_{ij\ell}$ that it sent to P_j, and everyone checking that $C_{ij} = \sum_{\ell=0}^{m} \sigma_{ij\ell}G_\ell$.

The dealer D_i is disqualified either if it was accused by more than t shareholders, or if it failed to respond to an accusation by broadcasting valid shares. Since disqualifying the dealer is a deterministic function of the content of the broadcast channel, then all honest parties will agree on whether or not D_i was disqualified.

To reconstruct all the $s_{i\ell}$'s, the participating shareholders pool their shares $\sigma_{ij\ell}$, which are checked against the check values C_{ij}. Any $t+1$ shares σ_{ij} that pass verification can be used to interpolate $s_{i\ell}$. (Note that share verification relies on the fact that *all the secrets* $s_{i0}, \ldots, s_{im}$ *are reconstructed together* and hence all the $\sigma ij*$'s are available.)

The soundness of this protocol is analyzed in Appendix A.1.

3.2 Multi-VSS

We now describe explicitly our multi-secret, multi-dealer VSS protocol (which we call *multi-VSS* for short). Namely, a protocol with multiple dealers (n' of them), each sharing multiple secrets (m of them), in parallel, to the same set of n shareholders. We denote the dealers by $D_1, ..., D_{n'}$, the shareholders by $P_1, \ldots, P_n$, and the ℓ'th secret of the i'th dealer is denoted $s_{i\ell}$.

The goal of the protocol is to establish sharing of "target secrets" $z_1, \ldots, z_m$, which are linear combinations of the $s_{i\ell}$'s, using public coefficients λ_i. Namely, $z_\ell = \sum_i \lambda_i s_{i\ell}$ for all ℓ. Moreover, for each target secret z_ℓ the protocol also outputs publicly the elements $Z_\ell = z_\ell G \in \mathbb{G}$.

We use the same linear combination for all the z_ℓ's, and that combination depends on the set Q of dealers that were not disqualified during their sharing. (In particular, the disqualified dealers do not contribute to the target sharing.) Namely, our protocol is parametrized by a public, efficiently computable, mapping $L : 2^{[n']} \to \mathbb{Z}_q^{n'}$ (that comes from the high-level application), such that for any $Q \subseteq [n']$, the i'th entry of $\vec{\lambda} = L(Q) \in \mathbb{Z}_q^{n'}$ is zero for any disqualified dealer, $i \notin Q$. After sharing the original secrets $s_{i\ell}$ and determining the set Q of qualified dealers, the protocol sets $\vec{\lambda} = L(Q)$ and computes a sharing of the target secrets $z_\ell = \sum_{i=1}^{n'} \lambda_i s_{i\ell} = \sum_{i \in Q} \lambda_i s_{i\ell}$. (For example, if we just want to sum up the shares of the qualified dealers then we have $\lambda_j = 1$ for $j \in Q$ and $\lambda_j = 0$ otherwise.)

The protocol uses a broadcast channel and a private communication channel between the dealers and the shareholders: For each dealer, we first run the amortized multi-secret VSS from Section 3.1. Then each shareholder computes its share of the target secrets z_ℓ as $\sigma_{j\ell} = \sum_i \lambda_i \sigma_{ij\ell}$. Finally, the shareholders engage in a protocol to generate the public values $Z_\ell = z_\ell G_\ell$ as follows: Each shareholder broadcasts the elements $Z_{j\ell} = \sigma_{j\ell}G_\ell$ for all ℓ, along with a NIZK-POK for the corresponding $\sigma_{j\ell}$. Then everyone verifies these NIZK proofs, and also uses the commitments to the columns of the matrices $\mathbb{A}_i$ (cf. Section 3.1) to check that these $Z_{j\ell}$'s lie on degree-t polynomials. If all these checks pass, then everyone can compute the public values Z_ℓ by interpolating "in the exponent" the $Z_{j\ell}$'s from $t+1$ shareholders. The protocol is described explicitly in Fig. 3.

Parameters:

- Integers m, n', n, t with $n \geq 2t+1$.
- Group $\mathbb{G}$ of prime order q and generators $G_0, G_1, \ldots, G_m \in \mathbb{G}$.
- Mapping $L : 2^{[n']} \to \mathbb{Z}_q^{n'}$ such that for all $Q \subseteq [n']$, $L(Q)_j = 0$ for all $j \notin Q$.

Inputs: Each dealer D_i has input secrets $s_{i1}, \ldots, s_{im} \in \mathbb{Z}_q$.

Multi secret VSS:

(1) **Dealing.** Each dealer D_i ($i \in [n']$):
  (a) Chooses a random value $s_{i0} \in \mathbb{Z}_q$.
  (b) For all $\ell \in [[m]]$, sets $a_{i0\ell} = s_{i\ell}$, chooses at random $a_{ik\ell} \in \mathbb{Z}_q$ for all $k \in [t]$.
  This defines the polynomials $f_{i\ell}(x) = \sum_{k=0}^{t} a_{ik\ell} x^k$ over $\mathbb{Z}_q$ for all $\ell \in [[m]]$.
  (c) For all $j \in [n], \ell \in [[m]]$, computes the shares $\sigma_{ij\ell} = f_{i\ell}(j)$ and sends to P_j over private channels.
  (d) For all $j \in [n]$, computes $C_{ij} = \sum_{\ell=0}^{m} \sigma_{ij\ell} G_\ell \in \mathbb{G}$ and broadcasts $C_{i1}, \ldots, C_{in}$.

(2) **Accusations.** Each shareholder P_j ($j \in [n]$):
  (a) For every dealer D_i, P_j verifies that $C_{i1}, \ldots, C_{in}$ lie on a degree-t polynomial using the linearity test from Section 2.3, and also checks that $C_{ij} = \sum_{\ell=0}^{m} \sigma_{ij\ell} G_\ell$.
  (b) P_j broadcasts an *accusation* message against D_i if the verification fails.

(3) **Response.** Each D_i broadcasts the shares $\sigma_{ij\ell}, \ell \in [[m]]$ for every P_j that accused it.

(4) **Disqualification.** A dealer D_i is disqualified if either
  - It was accused by more than t shareholders, or
  - It failed to broadcast shares $\sigma_{ij\ell}$ that pass the verification Step 2a in response to some accusation.

  Let $Q \subset [n']$ be the set of qualified dealers, that were not disqualified.
  For every $i \in Q$, each P_j that accused D_i replaces its shares $\sigma_{ij\ell}$ with the ones that D_i broadcasted.

Shareholder public values. Let $\vec{\lambda} = L(Q) \in \mathbb{Z}_q^{n'}$. Each shareholder P_j does the following:

(5) Computes $\sigma_{j\ell} = \sum_{i \in Q} \lambda_i \sigma_{ij\ell}$ for all $\ell \in [[m]]$.
(6) Computes and broadcasts $Z_{j\ell} = \sigma_{j\ell} G_\ell$ for all $\ell \in [[m]]$.
(7) For each $Z_{i\ell}$, broadcast a NIZK-POK of the corresponding $\sigma_{j\ell}$'s, using the protocol from Fig. 1.

Shareholder disqualifications. Every party $P_{j'}$ does the following for each shareholder P_j:

(8) Verifies the NIZK-POK of $Z_{j\ell}$, for all $\ell \in [[m]]$, and checks that $\sum_{\ell=0}^{m} Z_{j\ell} = \sum_{i \in Q} \lambda_i C_{ij}$
(9) If verification fails for then P_j is disqualified.

Let $Q' \subseteq [n]$ be the set of qualified shareholders, that were not disqualified. (If $|Q'| < t+1$ then abort.)

Reconstruction and global public key. The shares $\sigma_{j\ell}$ of any subset of $t+1$ shareholders from Q' (which can be verified against the public values $Z_{j\ell}$) can be used to interpolate the global secrets z_ℓ. Moreover, the $Z_{j\ell}$'s themselves can be used to compute the global public keys via $Z_\ell = z_\ell G_\ell = \sum_j \gamma_j Z_{j\ell}$, where the γ_j's are the Lagrange interpolation coefficients.

Figure 3: Multi-dealer, multi-secret VSS

3.3 Functionality and Simulation

We codify the security properties that we need from our MultiVSS protocol by an "ideal functionality" $\mathcal{F}_{MultiVSS}$. See Fig. 4 for a schematics of the functionality and the corresponding simulator.

Parameters. The functionality interacts with n' dealers D_i and n shareholders P_j, some of which are controlled by the ideal-world adversary. Let $\mathcal{B}$ be the set of adversarially-controlled dealers and $\mathcal{B}'$ are the adversarially-controlled shareholders, and assume that $\|\mathcal{B}'\| \leq t$.

The functionality is parametrized by the same group $\mathbb{G}$ of order q with generators $G_0, \ldots, G_m$ and the same mapping $L : 2^{[n']} \to \mathbb{Z}_q^{n'}$ from the high-level application.

Inputs. The functionality receives m "original secrets" $s_{i1}, \ldots, s_{im} \in \mathbb{Z}_q$ from each honest dealer D_i, and optionally also some of the dishonest dealers.

Qualified dealers. The functionality also receives from the adversary a set $P \subseteq \mathcal{B}$ of dealers under its control that should be included in the qualified set. All these dealers $i \in P$ must supply original secrets $s_{i\ell}$. The set of qualified dealers will include P and the honest dealers, $Q = P \cup \overline{\mathcal{B}}$.

Target secrets. The functionality computes the linear combination coefficients $\vec{\lambda} = L(Q)$, and computes the target secrets $z_\ell = \sum_{i \in Q} \lambda_i s_{i\ell}$ for all $\ell \in [m]$.

Shares and verification values. The simulator also sends to the functionality the alleged shares of the bad shareholders, $\{\sigma_{ij\ell} : i \in [n'], j \in B', \ell \in [m]\}$. The functionality prepares sharing of all the target secrets, which is consistent with those shares.

Specifically, for each qualified dealer $i \in Q$ it computes the unique polynomials $f_{i\ell}(\cdot)$ consistent with the bad shareholders' view and the secrets $s_{i\ell}$, then sets $g_\ell = \sum_{i \in Q} \lambda_i f_{i\ell}$. It then sets $\sigma_{j\ell} = g_\ell(j)$ for all $j \in [n]$ and $\ell \in [m]$, and sends to the adversary the public values $Z_{j\ell} = \sigma_{j\ell} G_\ell$ for the honest shareholders $j \notin \mathcal{B}'$ and all $\ell \in [m]$.

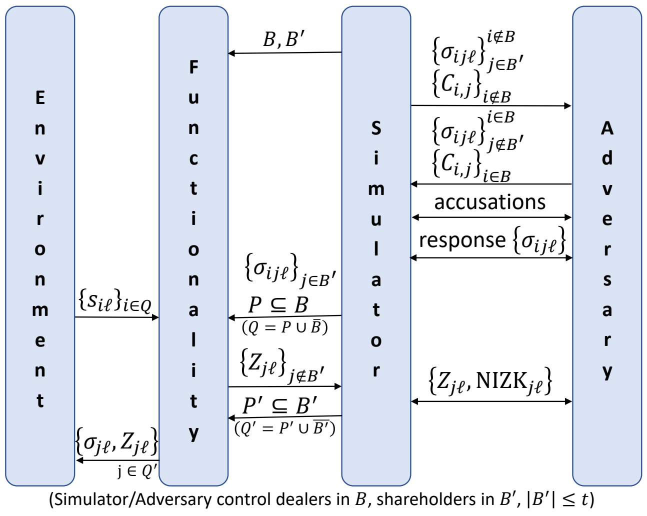

Environment
Functionality
Simulator
Adversary
B, B'
$\{\sigma_{ij\ell}\}^{i\notin B}_{j\in B'}$
$\{C_{i,j}\}_{i\notin B}$
$\{\sigma_{ij\ell}\}^{i\in B}_{j\notin B'}$
$\{C_{i,j}\}_{i\in B}$
accusations
response $\{\sigma_{ij\ell}\}$
$\{\sigma_{ij\ell}\}_{j\in B'}$
$\{s_{i\ell}\}_{i\in Q}$
$P \subseteq B$
$(Q = P \cup \bar{B})$
$\{Z_{j\ell}\}_{j\notin B'}$
$\{Z_{j\ell}, \mathrm{NIZK}_{j\ell}\}$
$P' \subseteq B'$
$(Q' = P' \cup \overline{B'})$
$\{\sigma_{j\ell}, Z_{j\ell}\}$
$j \in Q'$
(Simulator/Adversary control dealers in B, shareholders in B', $|B'| \le t$)

Figure 4: Schematics of the MultiVSS functionality $\mathcal{F}_{MultiVSS}$ and the simulator

Qualified shareholders and output. The adversary responds with a set $P' \subseteq \mathcal{B}'$ of shareholders under its control that should be included in the qualified set.

The functionality sets $Q' = P' \cup \overline{\mathcal{B}'}$. It returns to each qualified shareholder $j \in Q'$ the shares $\sigma_{j\ell} = g_\ell(j)$ for all $\ell \in [m]$. It also sends to everyone the verification values $Z_{j\ell} = \sigma_{j\ell} G_\ell$, for all $j \in Q'$ and $\ell \in [m]$.

Is $\mathcal{F}_{MultiVSS}$ meaningful? The functionality above is not very "ideal", in that the adversary $\mathcal{S}$ has a lot of influence over the output shares $\{\sigma_{j\ell}\}$. In what sense, then, does it provide meaningful secrecy guarantees?

Some answers can be found in Lemmas A.4 and A.5 below. The former says that the view in the sharing phase is independent of the original secrets shared by the honest dealers. The latter says that the entire protocol provides information-theoretic "semantic security" for the original secrets: Any two sets of original secrets that result in the same set of target secrets, will induce the same probability distribution over the adversary's view of the entire protocol. (This means that we could rewrite the functionality above to sample a new set of original secrets that results in the same target secrets, and use that other set to compute the shares $\sigma_{j\ell}$.)

Of course, whether or not this guarantee is enough may depends on the application . In our main motivation application where the target secrets are random secret keys, it turns out to be enough.

Analysis of the protocol, including the proof that it realize the functionality above, is provided in the appendix.

3.4 Complexity

We now briefly analyze the number of scalar-point multiplications and the bandwidth requirements in this protocol, which are the main complexity measures for this type of protocols.

Dealing. Each dealer $i \in [n']$ needs to compute the n "column commitments" C_{ij}. Each requires a single m-multi-scalar-point multiplication to compute the inner product between one column of the matrix $\mathbb{A}_i$ and the set of generators $(G_0, \ldots, G_m)$. The total number of scalar-point multiplications is thus $n(m+1)$ per dealer.

The dealer broadcasts the n group elements C_{ij} and sends $m+1$ scalars in $\mathbb{Z}_q$ to each shareholder, the total bandwidth over secret channels is $n(m+1)$ for each dealer.

Accusations. Each shareholder $j \in [n]$ must compute two inner-products to check each dealer: one for the linearity test and the other is an inner product between its shares $(\sigma_{ij0}, \ldots, \sigma_{ijm})$ and the generators $(G_0, \ldots, G_m)$. The total number of scalar-point multiplications per shareholder in this step is therefore $n'(n+m+1)$.

The bandwidth is upto n' accusation broadcasts per shareholder.

Response. There are no scalar-point multiplications computed in this step, but each dealer needs to broadcast $m+1$ scalars for each accusation. We can have upto t accusations (above that the dealer is automatically disqualified), so this step can consume broadcast bandwidth of upto $t(m+1)$ scalars.

Dealer disqualification. For every accusation $P_j \to D_i$, every other shareholder $P_{j'}$ must replicate the work of P_j (without the linearity test, which was already done). Namely perform $m+1$ scalar-point multiplications.

As there can be upto t accusations per dealer, this may require upto $tn'(m+1)$ more scalar-point multiplications for each shareholder. In the worst case, this is by far the most time-consuming step in this protocol.

Verification values. Each shareholder P_j performs $m+1$ scalar-point multiplications to compute the $Z_{j\ell}$'s, and $m+1$ more to prepare the NIZK proofs for them. Hence a total of $2m+2$ scalar-point multiplications.

In terms of bandwidth, each shareholder broadcasts $2m+2$ group elements ($Z_{j\ell}$'s and $R_{j\ell}$'s). [3]

Shareholder disqualification. Every shareholder $P_{j'}$ must verify the values of every other shareholder P_j. This requires verifying the NIZK ($2m+2$ scalar-point multiplications per P_j), and $\sum_{i\in Q} \lambda_i C_{ij}$ ($|Q| \le n'$ scalar-point multiplications per P_j). Hence the total number of scalar-point multiplications in this step is upto $n(n'+2m+2)$.

Summing up the above, we get the following total complexity:

Multiplications. Each dealer computes a total of $n(m+1)$ scalar-point multiplications.

If there are no accusations, then each shareholder computes

$$n'(n+m+1) + 2m + 2 + n(n' + 2m + 2)$$

scalar-point multiplications. In the worst case with many accusations, each shareholder may need to do upto $tn'(m+1)$ more scalar-point multiplications.

Broadcast bandwidth. Each dealer broadcasts n group elements if there are no accusations, and upto $t(m+1)$ more scalars if there are accusations.

[3]We use the Fiat-Shamir transformation, so the NIZK challenges c and the e_ℓ's are computed as a hash of the prover's message.

Each shareholder broadcasts upto n' accusations, $2m+2$ group elements and $m+1$ scalars.

The total broadcast bandwidth over the entire protocol (i.e., the number of "broadcast units" that each participants must listen to) is $n(n'+3m+3)$ if there are no accusations, and upto $n't(m+1)$ more if there are many accusations.

Point-to-point bandwidth. Each dealer sends $n(m+1)$ scalars over these channels, and each shareholder listens to $n'(m+1)$ scalars.

Comparison to naive Pedersen VSS. Using $m \cdot n'$ runs of single-dealer/single-secret Pedersen VSS would require that each shareholder computes roughly $mn't$ scalar-point multiplications, which is roughly a factor of t more than in our protocol. Our protocol also does slightly better in terms of bandwidth, saving roughly a factor of two on both broadcast and point-to-point bandwidth.

3.5 Further Improvements

3.5.1 Deriving more target secrets. While the protocol from Fig. 3 offers a large improvement over naive Pedersen, in many contexts it can be amortized a lot better via standard techniques. Below we mention briefly two of these standard techniques.

Packed secret sharing. If we have a larger honest majority among the shareholders, $n = 2t + p$ for some $p > 1$, then we can pack p secrets in each polynomial rather than just one [15]. Namely, for any $\ell \in [[m]]$, a dealer D_i with p secrets $\vec{s}_{i\ell} = (s_{i1\ell}, \ldots, s_{ip\ell})$ choose a random polynomial $f_{i\ell}(\cdot)$ of degree $t+p-1$ subject to (say) $f_{i\ell}(1-k) = s_{ik\ell}$ for $k = 1, 2, \ldots, p$. The shares are computed as usual $\sigma_{ij\ell} = f_{i\ell}(j)$ for $j = 1, 2, \ldots, n$. Since the polynomial has higher degree, then no set of t or less shareholders has any information about the secrets.

To compute the target secrets, we use the application-dependent linear combination $\vec{\lambda} = L(Q)$ as before. But now for every $\ell \in [m]$ we can derive p independent target secrets, $\vec{z}_\ell = \sum_{i \in Q} \lambda_i \vec{s}_{i\ell} \in \mathbb{Z}_q^p$. Hence we can produce pm target secrets rather than just m of them, for almost the same complexity. (Later it will be convenient to denote this by $\vec{z}_\ell = \vec{\lambda} \cdot [s]_\ell$, for a matrix $[s]_\ell \in \mathbb{Z}_q^{|Q| \times p}$ that has rows $\vec{s}_{i\ell}$ for all $i \in Q$.)

Super-invertible matrices. In settings where we have at least d honest dealers, we can sometime produce dm target secrets rather than just m without any increase in complexity. The Multi-secret VSS remains exactly the same as in Fig. 3, but rather than an application-dependent vector λ we will now use a matrix. Denoting $Q = \{i_1, \ldots, i_u\}$, we have a u-by-d matrix $[\lambda] = L(Q)$. For every $\ell \in [[m]]$ we then have a d-vector of target secrets, $\vec{z}_\ell = [\lambda] \cdot (s_{i_1 \ell}, \ldots, s_{i_u \ell})^T$.

If Λ is super-invertible [20] (i.e., every d-by-d matrix is invertible), then there is a 1-to-1 mapping between the original secrets of the d honest dealers and the d target secrets. This means that if the honest dealers choose random secrets to share, then the target secrets will also be random and independent.

Of course, these two techniques can be combined, if we have d honest dealers and a large honest majority of shareholders $n = 2t+p$. In that case we can compute for each $\ell \in [m]$ a d-by-p matrix of target secrets, $[z]_\ell = [\lambda] \times [s]$. This can be used, for example, in the application for mass distributed DSA-type signatures, where we want to generate many random group elements while sharing their discrete logarithm between the parties.

We note that the proof of security from above carries almost verbatim to these cases, using vectors/matrices instead of scalars as appropriate.

3.5.2 Faster handling of accusations. As described, the protocol from Fig. 3 has worst-case complexity $O(n^3)$ if there are many accusations (assuming $m, n', t = \Theta(n)$). In some cases this can be improved to $O(n^2)$, by using broadcast to implement also the private channels. If PKI is available, then one could implement the private channels by broadcasting encrypted messages. In this case, each private message from dealer to the verifier leaves a public record, which can be helpful in resolving accusations.

Specifically, the dealer can use committing encryption over broadcast to send the shares to the V_j's. For every accusation $V_j \to D_i$, the dealer can open the encryption, and everyone can check if the shares that were sent were valid. This means that whenever an accusation is resolved, either the dealer or the verifier will be disqualified. Hence at most $n + n'$ accusations would ever need to be checked, so the overall complexity of handling them all is reduced to $O(n^2)$.

3.5.3 Beyond honest majority. It is easy to see that the protocol above works right out of the box in settings where there is a mix of semi-honest and malicious parties. If there are s semi-honest and t malicious parties, the protocol ensures both privacy and correctness as long as $n > 2t + s$. Note that this is a weaker condition than $n > 2(t + s)$, so in some settings we can get security even when $n - t - s < t + s$. In other words, even if there are more dishonest than honest parties.

4 MULTI-DEALER VSS IN THE YOSO SETTING

Next we extend techniques from Section 3 and show how to utilize them to get efficient (proactive) VSS also in the more challenging YOSO model [18].

4.1 The basic Multi-Dealer VSS Protocol

We begin with a basic protocol for *multi-dealer VSS*, where we have n dealers D_i, each holding a single secret s_i, that they want to verifiably share among the same new set of n shareholders P_ℓ. This basic protocol lies at the core of many applications, including proactive refreshing, distributed key generation, distributed ephemeral randomness generation for Schnorr-like signatures, and more. Later, in Section 4.3, we show how to build on this basic tool in order to implement a share-refresh protocol with public keys.

As we mentioned in the introduction, we need to make quite a few changes in order to adapt the protocol from Fig. 3 to the YOSO model. We being with a rather high-level description of these changes, and then describe the YOSO protocol in detail in Fig. 5.

Added levels of sharing. As described in the introduction, porting our accusation-based protocol to the YOSO model entails (essentially) two more levels of secret-sharing: Messages of the original protocol must be shared among a committee of verifiers. It is this added level of sharing that necessitates the use of multi-secret VSS techniques even in a one-secret-per-dealer setting. Furthermore, each message to a verifier must be shared among a committee of responders.

Broadcast-only communication. Another difference is that in typical YOSO applications, all communication must be implemented via broadcast. In our protocol we take advantage of this trait and implement the fast accusation resolution trick from Section 3.5.2. This trick helps us offset to some degree the $n\times$ overhead that comes from the extra level of sharing.

Secret sharing made short. Another important optimization has to do with the dealer-to-responders-committee bandwidth. Recall that we need both another level of sharing via the verifiers V_j, and that every message $D_i \rightarrow V_j$ *must be shared among the responders committee.* Naively, this would imply yet another $n\times$ factor in communication.

Luckily, the broadcast channel saves us here too, by using Krawczyk's "secret sharing made short" technique [21]. To send a (long) message from D_i to V_j, the dealer chooses a short symmetric key ϵ_{ij}, uses it to encrypt the message, and broadcasts the ciphertext. Then it privately sends ϵ_{ij} to V_j, and also shares it among the responders committee members.

4.2 Analysis of the YOSO Multi-Dealer VSS protocol

The protocol is described in Fig. 5. We note that just as in the non-YOSO protocol from Section 3, all the honest shareholders agree on the set Q of qualified dealers. (However, they may disagree on the set of remaining verifiers.)

In terms of secrecy, we point out that as long as there are at most t bad parties in each of the committees, the adversary's view in this protocol is essentially the same as in the non-YOSO protocol. (Where the multiple secrets of the dealer in the protocol from Section 3 are replaced by the shares $\sigma_{i\ell}$.)

As stated the group elements that are sent on the broadcast channel in the $\mathbb{A}_i$ include the shares rather than coefficients, which is equivalent since one can be computed efficiently from the other. Then there are some commitments and ciphertexts that are sent and never opened, and the ones that are opened correspond to accusations, which means that either sender or receiver must be bad. Finally, there are the shares of the bad parties, which are uniformly random (since there are at most t of them in each committee).

A formal simulation proof (using the UC-formulation from [18]) will be similar to Theorem A.3, and is deferred to later work.

For soundness, we prove the following lemma, formalizing the informal argument that the only way for V_j to fool P_ℓ is to have committed (via the hash) to different shares that happen by chance to agree with the real shares on a random linear combination. Namely, V_j must commit to values $(\sigma'_{1j\ell}, \ldots, \sigma'_{nj\ell})$ that satisfy $\sum_{i=1}^{n} e_{ij}(\sigma'_{ij\ell} - \sigma_{ij\ell}) = 0$, which is enforced by the checks 3c and 3d performed by P_ℓ. If the shares are not all the same, then the probability of choosing e_{ij}'s that satisfy this equality is $1/q$.

In the proof below, we ignore the Fiat-Shamir aspect of the protocol and analyze it as if it was an interactive protocol. That is, we consider instead a protocol in which V_j first sends to P_ℓ the shares $\sigma'_{ij\ell}$, then the coefficients e_{ij} are chosen at random, and then V_j uses them to compute the R_{jr}'s. (In the protocol, the e_{ij}'s are computed instead by applying a random oracle to $h_{j\ell} = Hash(\sigma_{ij\ell})$ and some other h'es that V_j broadcasts and P_ℓ doesn't check.)

For every honest shareholder P_ℓ, and every (not necessarily honest) dealer and verifier D_i, V_j, let $\sigma'_{ij\ell}$ be the alleged share received by P_ℓ from D_i via V_j. (This is either the value sent from V_j if there was no accusation, or the value that was reconstructed by the W_k's if V_j accused D_i, or 0 is there was an accusation but no value was reconstructed.) We also let $\sigma_{ij\ell}$ denote "the correct share", which is defined as follows:

- If D_i is honest then this is just the value of $f_{ij}(\ell)$ that was sent to V_j (and shared among the W_k's).
- If D_i is dishonest and V_j is honest, then $\sigma_{ij\ell}$ is either the value that V_j received (if V_j did not accuse D_i, or the value that was reconstructed from the W_k's (if V_j did accuse D_i), or 0 (if there was an accusation but no value was reconstructed).
- If neither D_i nor V_j are honest, then $\sigma_{ij\ell}$ is defined as the interpolation of the "real shares" $\sigma_{ij'\ell}$ for all the honest verifiers $V_{j'}$, if they lie on a degree-t polynomial, or 0 if they do not.

We note that the values $\sigma_{ij\ell}, \sigma'_{ij\ell}$ are all well defined by the time that P_ℓ runs the disqualification procedure. Note also that by definition, $\sigma'_{ij\ell} = \sigma_{ij\ell}$ if V_j accused D_i and some value was reconstructed by the W_k's.

Lemma 4.1. *Fix an honest shareholder P_ℓ and some (not necessarily honest) verifier V_j. Assume that the coefficients $(e_{ij})_{i\in NA_j}$ are chosen at random after all the $\sigma'_{ij\ell}$ (but before the R_{jr}'s). Let $Q \subseteq [n]$ denote the set of indexes of qualified dealers D_i at the end of the protocol.*

Then, the probability that there exists $D_{i^*} \in Q$ such that $\sigma'_{i^*j\ell} \neq \sigma_{i^*j\ell}$, and yet V_j is not ignored by P_ℓ, is negligible.

Proof. By the same argument as in Lemma A.1 (as applied to the YOSO protocol), for every $i \in Q, \ell \in [n]$, all the shares $\sigma_{i1\ell}, \sigma_{in\ell}$ lie on a degree-t polynomial and can be correctly interpolated from the shares of the honest verifiers.

Recall that NA_j is the set of parties D_i that were not accused by V_j. Note that dealers accused by V_j may end up qualified, and some that V_j did not accuse may end up being disqualified. Below we denote the last set by $NQ_j = NA_j \setminus Q$, and let $Q_j^+ = Q \cup NQ_j$.

In Step 6 in the protocol, an honest V_j sets $\rho_{j\ell} = \sum_{i\in NA_j} e_{ij}\sigma_{ij\ell}$ and $R_{j\ell} = \rho_{j\ell}G_\ell$. For us it will be convenient to always sum over Q_j^+, so we define $e_{ij} = 0$ for all $i \in NQ_j$. For an honest V_j we therefore have $\rho_{j\ell} = \sum_{i\in Q_j^+} e_{ij}\sigma_{ij\ell}$. Even when V_j is dishonest, we still have the value $R_{j\ell}$ that it broadcast, and we can use the NIZK-POK extractor to extract from it also a scalar ρ_{ij} such that $R_{j\ell} = \rho_{j\ell}G_\ell$.

Let $\rho^*_{j\ell}$ denote $\sum_{i\in Q} e_{ij}\sigma_{ij\ell}$ and let $\delta_{j\ell} = \rho_{j\ell} - \rho^*_{j\ell}$. (I.e., ρ^* is computed from the "correct shares" $\sigma_{ij\ell}$, while ρ related to the "alleged shares" $\sigma'_{ij\ell}$ that P_ℓ gets from V_j.) Note that $\sum_{\ell=0}^{n} \rho^*_{j\ell}G_\ell = \sum_{i\in Q} e_{ij}C_{ij}$. Thus, we have

$$\begin{aligned}\sum_{\ell=0}^{n} \delta_{j\ell}G_\ell &= \sum_{\ell=0}^{n} \rho_{j\ell}G_\ell - \sum_{\ell=0}^{n} \rho^*_{j\ell}G_\ell \\ &= \sum_{i\in Q_j^+} e_{ij}C_{ij} - \sum_{i\in Q} e_{ij}C_{ij} = \sum_{i\in NQ_j} e_{ij}C_{ij}\end{aligned}$$

Parameters: $n \geq 2t+1$, group $\mathbb{G}$, generators $G_0, \ldots, G_n \in \mathbb{G}$.
Parties: Dealers D_i, verifiers V_j, responders W_k, shareholders P_ℓ.
Inputs: Each D_i, $i \in [n]$, has input $s_i \in \mathbb{Z}_q$.

Dealing. Each dealer D_i, $i \in [n]$, does the following:

1. Choose a random degree-t polynomial $f_i(\cdot)$ s.t. $f_i(0) = s_i$.
2. Set $s_{i\ell} = f_i(\ell)\ \forall \ell \in [n]$, and choose a random s_{i0}.
3. $\forall \ell \in [[n]]$, choose a random degree-t polynomial $f_{i\ell}(\cdot)$ s.t. $f_{i\ell}(0) = s_{i\ell}$. Set $\sigma_{ij\ell} = f_{i\ell}(j)$ for all $j \in [n]$.
4. $\forall j \in [n]$, denote $\vec{\sigma}_{ij} = (\sigma_{ij0}, \ldots, \sigma_{ijn})$ and then do:
   (a) Compute column-check values $C_{ij} = \vec{\sigma}_{ij} \bullet \vec{G} = \sum_{\ell=0}^{n} \sigma_{ij\ell} G_\ell$.
   (b) Choose a random degree-t polynomial $g_{ij}(\cdot)$, and set $\epsilon_{ij} = g_{ij}(0)$ and $\epsilon_{ijk} = g_{ij}(k)$ for all $k \in [n]$.
   (c) Encrypt ϵ_{ij} with V_j's public key, ciphertext is E'_{ij}.
   (d) Encrypt $\vec{\sigma}_{ij}$ with symmetric key ϵ_{ij}, ctxt is E_{ij}.
5. $\forall k \in [n]$, denote $\vec{\epsilon}_{ik} = (\epsilon_{i1k}, \ldots, \epsilon_{ink})$.
   (a) Compute a vector commitment γ_{ik} to $\vec{\epsilon}_{ik}$, and let ρ_{ik} be the randomness needed to open it.
   (b) Encrypt $(\vec{\epsilon}_{ik}, \rho_{ik})$ with W_k's public key, ctxt is F_{ik}.
6. Broadcast $\{C_{ij}, E_{ij}, E'_{ij}\}_{j\in[n]}$ and $\{\gamma_{ik}, F_{ik}\}_{k\in[n]}$.

Accusations. Each verifier V_j, $j \in [n]$, does the following:

1. For all $i \in [n]$:
   (a) Decrypt E'_{ij} and then E_{ij} to recover the shares $\vec{\sigma}_{ij}$.
   (b) If $C_{ij} \neq \vec{\sigma}_{ij} \bullet \vec{G}$, broadcast an *accusation* against D_i.
2. Define the set $NA_j = \{i : D_i \text{ was not accused}\}$.
3. $\forall \ell \in [[n]]$, denote $\vec{\sigma}^*_{j\ell} = (\sigma_{ij\ell} : i \in NA_j)$.
4. Let $h_{j\ell}$ be a commitment to $\vec{\sigma}^*_{j\ell}$ and $\tau_{j\ell}$ is the opening. Denote $\vec{h}_j = (h_{j0}, \ldots, h_{jn})$.
5. Set $\vec{e}_j = (e_{ij} : i \in NA_j) \in \mathbb{Z}_q^{|NA_j|}$ as $\vec{e} \leftarrow O(j, \vec{h}_j, NA_j)$, where O is modeled as a random oracle.
6. $\forall \ell \in [n]$ set $\rho_{j\ell} = \vec{\sigma}^*_{j\ell} \bullet \vec{e}_{j\ell} \in \mathbb{Z}_q$ and row-check values $R_{j\ell} = \rho_{j\ell} G_\ell \in \mathbb{G}$.
7. Compute a NIZK-POK for $\rho_{j\ell} = DL_{G_\ell}(R_{j\ell})\ \forall \ell \in [[n]]$.
8. For $\ell \in [n]$, send $(\vec{\sigma}^*_{j\ell}, \tau_{j\ell})$ privately to P_ℓ.
9. For $\ell \in [[n]]$, broadcast $(h_{j\ell}, R_{j\ell}, NIZK_{j\ell})$.

Verifiers that broadcast more than t accusations are ignored. Dealers that received more than t accusations are disqualified.

Response. Responder W_k does the following for every remaining accusation $V_j \to D_i$:

1. Decrypt F_{ik} to recover $(\vec{\epsilon}_{ik}, \rho_{ik})$.
2. Use ρ_{ik} to open the commitment to ϵ_{ijk} inside of $\vec{\epsilon}_{ik}$
3. Broadcast ϵ_{ijk} and its opening.

Disqualifications. Every shareholder P_ℓ does the following:

1. For each non-disqualified dealer D_i, check that $C_{i1}, \ldots, C_{in}$ lie in the linear subspace of evaluations of degree-t polynomials. Disqualify D_i if the test fails.
2. As long as there remain accusations $V_j \to D_i$:
   (a) Check all the commitments γ_{ik} against the opening to ϵ_{ijk} that the W_k's broadcasted. Disqualify D_i if less than $n-t$ are valid, or if not all the valid ϵ_{ijk}'s lie on a degree-t polynomial.
   (b) Otherwise use the shares ϵ_{ijk}'s to recover ϵ_{ij}, and use ϵ_{ij} to decrypt E_{ik} and recover $\vec{\sigma}_{ij}$.
   (c) If $C_{ij} \neq \vec{\sigma}_{ij} \bullet \vec{G}$ then disqualify D_i. Otherwise ignore V_j and all of its accusations and shares.
3. For each remaining verifier V_j with received shares $\sigma'_{1j\ell}, \ldots, \sigma'_{nj\ell}$ and commitment opening $\tau_{j\ell}$ check the following (and ignore V_j if any check fails):
   (a) Check $\vec{\sigma}^*_{j\ell}, \tau_{j\ell}$ against the commitment $h_{j\ell}$.
   (b) Verify all the proofs $\{NIZK_{jr}\}_{r\in[[n]]}$.
   (c) Set $\vec{e} \leftarrow O(j, \vec{h}_j, NA_j)$ and $\rho'_{j\ell} = \sum_{i\in NA_j} e_{ij}\sigma'_{ij\ell}$. Check that $R_{j\ell} = \rho'_{j\ell} G_\ell$.
   (d) Check that $\sum_{r=0}^{n} R_{jr} = \sum_{i\in NA_j} e_{ij} C_{ij}$.

Shares. P_ℓ aborts if it has less than $t+1$ remaining verifiers. Let $J = \{j_0, \ldots, j_t\}$ be its first $t+1$ remaining shareholders and Q be the qualified dealers, and denote by $\{\sigma'_{ij\ell} : i \in Q, j \in J\}$ their shares as received by P_ℓ. Let $(\lambda_j : j \in J)$ be the Lagrange interpolation coefficients for J.
For every qualified dealer D_i, $i \in Q$, P_ℓ computes its share of the secret s_i as $s_{i\ell} = \sum_{j\in J} \lambda_j \sigma'_{ij\ell}$.

Figure 5: The basic YOSO Muti-Dealer VSS protocol

where we have used the verification steps 3c and 3d applied by P_ℓ. Thus, $\sum_{\ell=0}^{n} \delta_{j\ell} G_\ell = \sum_{i\in NQ_j} e_{ij} C_{ij}$, where the $\delta_{j\ell}$ are values extractable by the protocol and are defined uniquely in the sense that it would be infeasible to compute another set of $\delta'_{j\ell}$ that will have this property.

For the values $\sigma'_{ij\ell}$ that P_ℓ gets to pass verification we need $\sum_{i\in Q^+_j} e_{ij}\sigma'_{ij\ell} G_\ell = R_{j\ell} = \rho_{j\ell} G_\ell = (\delta_{j\ell} + \rho^*_{j\ell}) G_\ell$, where the first equality is verified by P_ℓ. This holds if and only if $\sum_{i\in Q^+_j} e_{ij}\sigma'_{ij\ell} = \delta_{j\ell} + \rho^*_{j\ell} = \delta_{j\ell} + \sum_{i\in Q} e_{ij}\sigma_{ij\ell}$. Equivalently,

$$\sum_{i\in Q} e_{ij}(\sigma'_{ij\ell} - \sigma_{ij\ell}) = \delta_{j\ell} - \sum_{i\in NQ_j} e_{ij}\sigma'_{ij\ell} \tag{2}$$

If there is $i' \in Q$ such that $\sigma'_{i'j\ell} \neq \sigma_{i'j\ell}$ then the probability that (2) holds for random $e_{i'j}$ is $1/q$. Indeed, $e_{i'j}$ can assume a single value determined by an expression where $\sigma'_{i'j\ell} - \sigma_{i'j\ell}$ is a divisor. Note that it is important that the right-hand side in (2) does not depend on $i' \in Q$; in particular, $\delta_{j\ell}$ (which satisfies $\sum_{\ell=0}^{n} \delta_{j\ell} G_\ell = \sum_{i\in NQ_j} e_{ij} C_{ij}$) only depends on e_{ij} for $i \in NQ_j$, and the corresponding C_{ij}. ◻

4.3 Proving consistent sharing and correct public keys/commitments

We now describe how to extend the basic multi-dealer VSS from Fig. 5 to get a proactive re-sharing protocol, with public verification keys. To that end, we need to have the dealers prove that the secrets that they share are "the right ones", and we need to add a procedure for computing public verification values for the shares.

In more detail, in our context "the right secrets" are themselves shares of a single global secret, and each share s_i has an associated public value $S_i = s_i G$. At the beginning of the protocol each dealer holds a share s_i as before , but now we assume that the s_i's lie on a degree-t polynomial, and in addition all the values $S_i = s_i G$ are publicly known. Our goal is to maintain that invariant, so that the same will hold also at the end of the protocol, for new shares s'_ℓ that are held by the shareholders.

Before we begin, we note that *if the dealers share the "right secrets"* (i.e. valid shares of a global secret), then the shareholders in the protocol from Fig. 5 can indeed compute new shares of the same global secret. To see that, assume that the original secrets indeed lie on a degree-t polynomial. Namely, $s_i = f(i)$ for some degree-t polynomial $f(\cdot)$, and the global secret is $s = f(0)$.

Recall that each honest dealer D_i chooses a degree-t polynomial f_i such that $f_i(0) = s_i = f(i)$, and that at the conclusion of the protocol each honest shareholder P_ℓ holds the shares $\sigma_{i\ell} = f_i(\ell)$ for every qualified dealer $i \in Q$.

Assume that $|Q| \geq t+1$ and let $I = \{i_0, \ldots, i_t\} \subseteq Q$ be the first $t+1$ qualified dealers. Let $\mu_{i_0}, \ldots, \mu_{i_t}$ be the Lagrange interpolation coefficients for the set I. Namely $p(0) = \sum_{i\in I} \mu_i p(i)$ for every degree-t polynomial p. Consider then the degree-t polynomial $g = \sum_{i\in I} \mu_i f_i$. On the one hand, each shareholder P_ℓ knows $f_i(\ell)$ for all $i \in I$, so it can compute its share $s'_\ell = g(\ell) = \sum_{i\in I} \mu_i f_i(\ell)$. On the other hand, we have

$$g(0) = \sum_{i\in I} \mu_i f_i(0) = \sum_{i\in I} \mu_i f(i) = f(0).$$

Hence the scalars s'_ℓ are indeed shares of the same secret as the original s_i's. We now turn to the task at hand, having the dealers prove that they shared the right secrets, and computing the public verification elements.

4.3.1 *A first attempt.*

A natural approach for proving correct re-sharing goes as follows: Each dealer D_i, after choosing the polynomial f_i and computing a value $\sigma_{i\ell} = f_i(\ell)$ that it wants to communicate to P_ℓ's (via the V_j's), would broadcast all the corresponding values $Z_{i\ell} = \sigma_{i\ell} G$ (along with NIZK proofs of correctness). Once the set of qualified dealers Q is established (and hence the Lagrange interpolation coefficients μ_i are known), this will let everyone compute the public value $S'_\ell = \sum_i \mu_i Z_{i\ell}$. The same derivation as above implies that this is the correct public value. Namely $S'_\ell = s'_\ell G$ for the share s'_ℓ that P_ℓ recovered.

Unfortunately, this approach suffers from the randomness-biasing drawback that was pointed out by Gennaro et al. [17]: After seeing the public values $Z_{i\ell}$, the adversary can influence the set of qualified dealers (by making verifiers that it controls accuse/not accuse dealers that it controls). It can therefore bias the public values S'_ℓ (e.g., ensuring that S'_i begins with a '0'), by trying different combinations for Q until it finds one that yields the desired result.

4.3.2 *Preventing biasing attacks.*

To overcome this drawback, we need to depart somewhat from the protocol above, and move closer to classical Pedresen VSS. Instead of setting $Z_{i\ell} = \sigma_{i\ell} G$, the dealer will broadcast $Z_{i\ell} = \sigma_{i\ell} G + r_{i\ell} H$ (together with proofs), for some randomizer $r_{i\ell}$. The dealer D_i will then communicate *both* $\sigma_{i\ell}$ *and* $r_{i\ell}$ to P_ℓ via the V_j's. Note, that like the $\sigma_{i\ell}$'s the $r_{i\ell}$'s also sit on a polynomial of degree t.

As before, once the qualified set and Lagrange coefficients are determined, anyone can compute $\tilde{S}_\ell = \sum_i \mu_i Z_{i\ell}$. By the same argument as above, we have $Z_\ell = s'_\ell G + r'_\ell H$, where s'_ℓ and r'_ℓ are the share and randomness that P_ℓ computes via interpolation. Hence Z_i does not leak any information on s'_ℓ or the corresponding S'_ℓ. While the adversary can still influence the set of qualified dealers, it has no information about the way that different Q's affect S'_ℓ.

Later, when P_ℓ gets to speak, it will broadcast the real public value $S'_\ell = s'_\ell G$ and $r'_\ell H$, along with a NIZK proofs that it knows value s'_ℓ and r'_ℓ.

This solution entails more or less a 2× increase in complexity. This is because now D_i needs to communicate two values to each shareholder P_j (and each of those must be communicated via the V_j's and W_k's). It remains an interesting problem to achieve the same result without this extra overhead.

4.3.3 *The protocol.*

The invariant that we maintain is that at the beginning of each step, each honest dealer D_i knows s_i, r_i, and everyone knows the corresponding public value $Z_i = s_i G + r_i H$. Moreover, the s_i's of the honest dealers lie on a degree-t polynomial.

The protocol now uses twice as many generators G_ℓ, H_ℓ, $\ell \in [[n]]$ (and also the designated G, H that are mentioned above). The G_ℓ's will be used for the "real shares" $\sigma_{ij\ell}$, and the H_ℓ's will be used for the randomizers. The protocol from Fig. 5 is modified as follows:

- Each dealer D_i repeats all the actions from Fig. 5 twice, once for s_i wrt the G_ℓ's and once for r_i wrt the H_ℓ's. Let $\sigma_{i\ell}$ be the shares of s_i and $\sigma_{ij\ell}$ be the shares of $\sigma_{i\ell}$. Similarly $\rho_{i\ell}$ are the shares of r_i and $\rho_{ij\ell}$ are the shares of $\rho_{i\ell}$. As before, we denote $\vec{\sigma}_{ij} = (\sigma_{ij0} \ldots \sigma_{ijn})$ and $\vec{\rho}_{ij} = (\rho_{ij0} \ldots \rho_{ijn})$.
- Since everything is repeated twice, then in particular D_i broadcasts two sets of column-checking values, $C_{ij} = \sum_{\ell=0}^{n} \sigma_{ij\ell} G_\ell$ and $C'_{ij} = \sum_{\ell=0}^{n} \rho_{ij\ell} H_\ell$.
- D_i broadcasts $Z_{i\ell} = \sigma_{i\ell} G + \rho_{i\ell} H$ for all $\ell \in [[n]]$.
- D_i also broadcasts a NIZK of knowledge for:
  - Representations $\vec{\sigma}_{ij}$ of C_{ij} in the bases $\vec{G}$ for all $j \in [n]$;
  - Representations $\vec{\rho}_{ij}$ of C'_{ij} in the bases $\vec{H}$ for all $j \in [n]$;
  - Representations $(\sigma_{i\ell}, \rho_{i\ell})$ for $Z_{i\ell}$ wrt G, H $\forall \ell \in [n]$;

  such that the following conditions hold:
  - $(\sigma_{i\ell}, \sigma_{i0\ell}, \ldots, \sigma_{in\ell})$ lie on a degree-t polynomial $\forall \ell \in [n]$;
  - $(\rho_{i\ell}, \rho_{i0\ell}, \ldots, \rho_{in\ell})$ lie on a degree-t polynomial $\forall \ell \in [n]$.
- Finally, D_i also broadcasts $S_i = s_i G$ and a NIZK of knowledge that $s_i = DL_G(S_i)$ and the representation that it knows $Z_i = s_i G + r_i H$ agree on s_i.

4.3.4 *The NIZK proofs.*

The proofs that we need are proofs of representations that lie in some linear subspaces. These are standard, and can be implemented by standard tools. However, the witnesses contain $O(n^2)$ scalars in $\mathbb{Z}_q$, so a naive approach will take $O(n^2)$ scalar-point multiplications to prove/verify each of them. Instead, we show now that they can be proved/verified with only $O(n)$ scalar-point multiplications.

- Since the Z_i's and the $Z_{i\ell}$'s are all with respect to the same generators G, H, then everyone can verify on their own that they lie on a degree-t polynomial (using a standard linearity test). This has complexity $O(n)$ scalar-point multiplications.

- In addition to $Z_{i\ell} = \sigma_{i\ell}G + \rho_{i\ell}H$ for all $\ell \in [[n]]$, D_i also broadcasts $Z'_{i\ell} = \sigma_{i\ell}G_\ell + \rho_{i\ell}H_\ell$, and prove that these representations are the same. Proving it for all $\ell \in [[n]]$ takes a total of $O(n)$ scalar-point multiplications.
- D_i publishes also $C_{i0} = \sum_{\ell=0}^{n} \sigma_{i\ell}G_\ell$ and $C'_{i0} = \sum_{\ell=0}^{n} \rho_{i\ell}H_\ell$. The parties verify that their sum is correct, $C_{i0} + C'_{i0} = \sum_{\ell=0}^{n} Z'_{i\ell}$.

  D_i also proves it knows representations $(\sigma_{i\ell})_{\ell\in[[n]]}$ and $(\rho_\ell)_{\ell\in[[n]]}$ for C_{i0}, C'_{i0} in the bases $\vec{G}, \vec{H}$, respectively. This can be done with $O(n)$ scalar-point multiplications.

  This, together with the linearity test on $C_{i0}, C_{i1}, \dots, C_{in}$ (which takes another $O(n)$ scalar-point multiplications), ensures that not only the sharing of s_i into $s_{i1}, \dots, s_{in}$ is correct, but that the resharing of these n values is correct too. .

5 IMPLEMENTATION

We implemented the protocol in Section 4.3 with a few minor changes. Concretely, only a Pedersen commitment $S_i = s_iG + r_iH$ is made public instead of the Feldman commitment $S_i = s_iG$. This removes n NIZK proofs of knowledge of discrete logarithm and is useful in variants where s_iG does not need to be public. Adding back those n NIZK adds very limited computational and communication complexity). The protocol is also slightly optimized by removing completely s_{i0}, σ_{ij0}, r_{i0}, and ρ_{ij0} that are redundant in the Pedersen case, and by using $C_{ij} = \sum_{\ell=0}^{n} \sigma_{ij\ell}G_\ell + \sum_{\ell=0}^{n} \rho_{ij\ell}H_\ell$ instead of $C_{ij} = \sum_{\ell=0}^{n} \sigma_{ij\ell}G_\ell$ and $C'_{ij} = \sum_{\ell=0}^{n} \rho_{ij\ell}H_\ell$. Finally, the Lagrangian coefficients are cached to avoid recomputing them every time the same set of parties is used for reconstruction. Security follows easily from the security proof of the original protocol.

The implementation is in Go (version 1.17.6) for most of the code and in C/C++ for the elliptic curve and field operations.

We use a modified version of libsodium [14] (version 1.0.18) for the curve operations and most of the field operations. Modifications include use of non-compressed representation of elliptic curve points (to avoid overhead from decompression), support for multi-scalar-point multiplication (both a constant-time version for operations with secret scalars, and a variable-time version for operations with non-secret scalars), fixed-based scalar-point multiplication (and variant with two bases for Pedersen commitments), matrix multiplications over the scalar field, scalar polynomial evaluation, and other small additional functions. The code is pure portable C without any assembly optimizations. We also use NTL [26] for the generation of the parity-check matrix for the Shamir secret sharing (to do the linear tests). This operation needs to only be performed one time in the whole duration of the system (for a given number of parties n and threshold t).

The communication layer (i.e., the broadcast channels) is simulated using Go channels. (Channels are assumed to be authenticated.) Real networking communication layer can be built and used as a drop-in replacement of these simulated channels. Serialization is fully implemented and uses a canonical version of msgpack [16].

Benchmarking was done on a cloud VM with 50 cores and 128GB of RAM (CPU: AMD EPYC™ 7601, 2.2GHz). The OS was Ubuntu 20.04. For efficiency reasons, only the steps of the first party in each committee are fully executed, the other parties are partially simulated (to provide the required inputs for the first party of the other committees).

Table 1: Performance of the proactive YOSO resharing protocol in Section 4.3 (computation time and size of the broadcasted message of the first party in each committee)

		Dealing		Accusations		Disqualifications & Shares
t	n	Time	Size	Time	Size	Time
64	129	1.5 s	3.5 MB	1.0s	1.2 MB	3.2 s
128	257	7.2 s	14 MB	4.1s	4.6 MB	12.8 s
256	513	42 s	54 MB	17s	18 MB	52 s

We used a VM with 50 cores and 128GB of RAM just to be able to simulate all the other parties. A single party workload use a very small amount of RAM (less than a few GB) and *a single thread*. The workload of a single party is embarrassingly parallel and the use of multiple threads should give almost linear scaling.

Computational and communication complexity are reported in Table 1. Since all verifiers are honest, there are no accusations and the work of the responders is negligible, hence not reported. The "disqualifications & shares" step include verifying the NIZK made by the dealers and computing the new commitments $S'_i = s'_iG + r'_iH$. From the performance numbers in the table, we can see that all the steps appear to have a quadratic computation and communication complexity (except for the computation complexity of dealing): every time t doubles, the computational and communication complexity is multiplied by 4. Dealing becomes of cubic complexity for high t and n because the scalar operations (to actually generate the shares) become dominant (compared to the scalar-point multiplication that are quadratic and dominant for small t and n).

Micro-benchmarking for the Non-YOSO MultiVSS

In Appendix B we provide performance results for the non-YOSO MultiVSS protocol from Fig. 3 showing its performance for a varying number of servers and batched secrets, and highlighting the significant speedup as compared to a basic implementation of the classical Pedersen VSS protocol.

REFERENCES

[1] Joshua Baron, Karim El Defrawy, Joshua Lampkins, and Rafail Ostrovsky. 2014. How to withstand mobile virus attacks, revisited. In *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*, Magnús M. Halldórsson and Shlomi Dolev (Eds.). ACM, 293–302. https://doi.org/10.1145/2611462.2611474

[2] Joshua Baron, Karim El Defrawy, Joshua Lampkins, and Rafail Ostrovsky. 2015. Communication-Optimal Proactive Secret Sharing for Dynamic Groups. In *Applied Cryptography and Network Security - 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers (Lecture Notes in Computer Science)*, Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis (Eds.), Vol. 9092. Springer, 23–41. https://doi.org/10.1007/978-3-319-28166-7_2

[3] Mihir Bellare, Juan A. Garay, and Tal Rabin. 1996. Distributed Pseudo-Random Bit Generators - A New Way to Speed-Up Shared Coin Tossing. In *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing, Philadelphia, Pennsylvania, USA, May 23-26, 1996*, James E. Burns and Yoram Moses (Eds.). ACM, 191–200. https://doi.org/10.1145/248052.248090

[4] Mihir Bellare, Juan A. Garay, and Tal Rabin. 1998. Fast Batch Verification for Modular Exponentiation and Digital Signatures. In *EUROCRYPT'98 (LNCS)*, Kaisa Nyberg (Ed.), Vol. 1403. Springer, Heidelberg, 236–250. https://doi.org/10.1007/BFb0054130

[5] Fabrice Benhamouda, Craig Gentry, Sergey Gorbunov, Shai Halevi, Hugo Krawczyk, Chengyu Lin, Tal Rabin, and Leonid Reyzin. 2020. Can a Public Blockchain Keep a Secret?. In *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part I (Lecture Notes in Computer Science)*, Rafael Pass and Krzysztof Pietrzak (Eds.), Vol. 12550. Springer, 260–290. https://doi.org/10.1007/978-3-030-64375-1_10

[6] Erica Blum, Jonathan Katz, Chen-Da Liu-Zhang, and Julian Loss. 2020. Asynchronous Byzantine Agreement with Subquadratic Communication. In *TCC 2020, Part I (LNCS)*, Rafael Pass and Krzysztof Pietrzak (Eds.), Vol. 12550. Springer, Heidelberg, 353–380. https://doi.org/10.1007/978-3-030-64375-1_13

[7] Ignacio Cascudo and Bernardo David. 2017. SCRAPE: Scalable Randomness Attested by Public Entities. In *ACNS 17 (LNCS)*, Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi (Eds.), Vol. 10355. Springer, Heidelberg, 537–556. https://doi.org/10.1007/978-3-319-61204-1_27

[8] Dario Catalano and Dario Fiore. 2013. Vector Commitments and Their Applications. In *PKC 2013 (LNCS)*, Kaoru Kurosawa and Goichiro Hanaoka (Eds.), Vol. 7778. Springer, Heidelberg, 55–72. https://doi.org/10.1007/978-3-642-36362-7_5

[9] Arka Rai Choudhuri, Aarushi Goel, Matthew Green, Abhishek Jain, and Gabriel Kaptchuk. 2021. Fluid MPC: Secure Multiparty Computation with Dynamic Participants. In *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part II (Lecture Notes in Computer Science)*, Tal Malkin and Chris Peikert (Eds.), Vol. 12826. Springer, 94–123. https://doi.org/10.1007/978-3-030-84245-1_4

[10] Ronald Cramer. 1996. *Modular Design of Secure yet Practical Cryptographic Protocols.* Ph.D. Dissertation. CWI and University of Amsterdam.

[11] Ivan Damgård. 2010. On Σ Protocols. https://cs.au.dk/%7Eivan/Sigma.pdf. (2010).

[12] Ivan Damgård and Yuval Ishai. 2006. Scalable Secure Multiparty Computation. In *CRYPTO 2006 (LNCS)*, Cynthia Dwork (Ed.), Vol. 4117. Springer, Heidelberg, 501–520. https://doi.org/10.1007/11818175_30

[13] Ivan Damgård, Yuval Ishai, Mikkel Krøigaard, Jesper Buus Nielsen, and Adam Smith. 2008. Scalable Multiparty Computation with Nearly Optimal Work and Resilience. In *CRYPTO 2008 (LNCS)*, David Wagner (Ed.), Vol. 5157. Springer, Heidelberg, 241–261. https://doi.org/10.1007/978-3-540-85174-5_14

[14] Frank Denis. 2022. The Sodium cryptography library. (2022). https://download.libsodium.org/doc/

[15] Matthew K. Franklin and Moti Yung. 1992. Communication Complexity of Secure Computation (Extended Abstract). In *24th ACM STOC*. ACM Press, 699–710. https://doi.org/10.1145/129712.129780

[16] Sadayuki Furuhashi. 2013. MessagePack Serialization Format. (2013). https://msgpack.org/

[17] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. 2007. Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. *Journal of Cryptology* 20, 1 (Jan. 2007), 51–83. https://doi.org/10.1007/s00145-006-0347-3

[18] Craig Gentry, Shai Halevi, Hugo Krawczyk, Bernardo Magri, Jesper Buus Nielsen, Tal Rabin, and Sophia Yakoubov. 2021. YOSO: You Only Speak Once / Secure MPC with Stateless Ephemeral Roles. In *CRYPTO 2021, to appear.* https://ia.cr/2021/210.

[19] Craig Gentry, Shai Halevi, and Vadim Lyubashevsky. 2021. Practical Non-interactive Publicly Verifiable Secret Sharing with Thousands of Parties. *IACR Cryptol. ePrint Arch.* (2021), 1397. https://eprint.iacr.org/2021/1397

[20] Martin Hirt and Jesper Buus Nielsen. 2006. Robust Multiparty Computation with Linear Communication Complexity. In *CRYPTO 2006 (LNCS)*, Cynthia Dwork (Ed.), Vol. 4117. Springer, Heidelberg, 463–482. https://doi.org/10.1007/11818175_28

[21] Hugo Krawczyk. 1994. Secret Sharing Made Short. In *CRYPTO'93 (LNCS)*, Douglas R. Stinson (Ed.), Vol. 773. Springer, Heidelberg, 136–146. https://doi.org/10.1007/3-540-48329-2_12

[22] Stephan Krenn, Thomas Lorünser, and Christoph Striecks. 2017. Batch-verifiable Secret Sharing with Unconditional Privacy. In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy, ICISSP 2017, Porto, Portugal, February 19-21, 2017*, Paolo Mori, Steven Furnell, and Olivier Camp (Eds.). SciTePress, 303–311. https://doi.org/10.5220/0006133003030311

[23] Yehuda Lindell and Ariel Nof. 2018. Fast Secure Multiparty ECDSA with Practical Distributed Key Generation and Applications to Cryptocurrency Custody. In *ACM CCS 2018*, David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang (Eds.). ACM Press, 1837–1854. https://doi.org/10.1145/3243734.3243788

[24] Torben P. Pedersen. 1991. A Threshold Cryptosystem without a Trusted Party (Extended Abstract) (Rump Session). In *EUROCRYPT'91 (LNCS)*, Donald W. Davies (Ed.), Vol. 547. Springer, Heidelberg, 522–526. https://doi.org/10.1007/3-540-46416-6_47

[25] Torben P. Pedersen. 1992. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *CRYPTO'91 (LNCS)*, Joan Feigenbaum (Ed.), Vol. 576. Springer, Heidelberg, 129–140. https://doi.org/10.1007/3-540-46766-1_9

[26] Victor Shoup. 2022. NTL: A Library for doing Number Theory. (2022). https://libntl.org/

A PROOFS

A.1 Soundness of the Single-Dealer/Muti-Secret Protocol

Soundness of this protocol is established by showing that if the dealer was not disqualified, then the shares of all the honest parties must agree with some set of t-degree polynomials $f_{i\ell}(\cdot)$, $\ell = 0, 1, \ldots, m$. In more detail, we show that finding any set of shares $\{\sigma_{ik\ell}\}_{k,\ell}$ that pass verification according **??** but do not all lie on the same degree-d polynomials $f_{i\ell}(\cdot)$, implies finding a non-trivial representation of the element $0 \in \mathbb{G}$ in the bases $G_0, G_1, \ldots, G_m$. This only happens with a negligible probability, assuming that finding discrete logarithms in $\mathbb{G}$ is hard.

Lemma A.1. *There is an efficient transformation that computes a non-trivial representation of zero, $\sum_{\ell \in [[m]]} \lambda_\ell G_\ell = 0 \in \mathbb{G}$ with some of the λ_ℓ's non-zero, when given as input a subset $\mathcal{P} \subset [n]$ of size at least $t+1$ and shares $\{\sigma_{ij\ell} : j \in \mathcal{P}, \ell \in [[m]]\}$ that satisfy:*

- *For all $j \in \mathcal{P}$, $\sum_{\ell=0}^{m} \sigma_{ij\ell} \cdot G_\ell = C_{ij}$; but*
- *The shares $\{\sigma_{ij\ell}\}_{j\in\mathcal{P},\ell\in[[m]]}$ do not all lie on the same ℓ degree-t polynomials $f_{i\ell}(\cdot)$.*

Before proving the lemma, we note that when the dealer is not disqualified, taking $\mathcal{P}$ to be the set of honest parties implies that all the honest parties have their shares lying on the same degree-t polynomials. Also, taking $\mathcal{P}$ as the union of the honest parties and all the parties in the reconstruction set, implies that the secrets $s_{i\ell}$ that can be reconstructed are fixed after the dealing phase, regardless of the reconstruction set.

Proof. The lemma is vacuously true for $\mathcal{P}$ of size $t+1$, since any set of $t+1$ values in $\mathbb{Z}_q$ are on a polynomial of degree t. Consider, therefore, a larger set $\mathcal{P}$. Let $\{j_0, j_1, \ldots, j_{t+1}\} \subseteq \mathcal{P}$ be $t+2$ members of $\mathcal{P}$ whose shares *do not lie on ℓ degree-t polynomials as above.*

Note that the evaluations of degree-t polynomials at the points $j_0, j_1, \ldots, j_{t+1}$ is a linear space of degree $t+1$ over $\mathbb{Z}_q$. Hence there exist coefficients $\gamma_0, \ldots, \gamma_t \in \mathbb{Z}_q$ such that for any polynomial $p(\cdot)$ of degree $\le t$ it holds that, $p(j_{t+1}) = \sum_{u=0}^{t} \gamma_u \cdot p(j_u)$.

By the soundness of the linearity-test in the Accusations step, we have that $C_{ij} = p(j)G$ holds for all $j \in [n]$, for some generator $G \in \mathbb{G}$ and degree-t polynomial $p(\cdot)$ over $\mathbb{Z}_q$. In particular it means that $C_{i,j_{t+1}} = \sum_{u=0}^{t} \gamma_u C_{ij_u}$.

Furthermore, since $\sum_{\ell=0}^{m} \sigma_{ij\ell} \cdot G_\ell = C_{ij}$ for all $j \in \mathcal{P}$ (and in particular for j_t), then

$$\begin{aligned}\sum_{\ell=0}^{m} \sigma_{i,j_{t+1},\ell} G_\ell &= C_{i,j_{t+1}} = \sum_{u=0}^{t} \gamma_u C_{ij_u} \\ &= \sum_{u=0}^{t} \gamma_u \left(\sum_{\ell=0}^{m} \sigma_{i,j_u,\ell} G_\ell \right) = \sum_{\ell=0}^{m} \left(\sum_{u=0}^{t} \gamma_u \sigma_{i,j_u,\ell} \right) G_\ell.\end{aligned}$$

If $\sigma_{i,j_{t+1},\ell} = \sum_{u=0}^{t} \gamma_u \sigma_{i,j_u,\ell}$ for all ℓ, then the points $\{\sigma_{ij_u\ell} : u \in [[t]]\}$ all lie on a polynomial of degree t, contradicting our choice of this set of point.

We therefore conclude that for some $\ell \in [[m]]$ we have $\sigma_{i,j_{t+1},\ell} \neq \sum_{u=0}^{t} \gamma_u \sigma_{i,j_u,\ell}$, and so we get the non-trivial representation that we need, $\lambda_\ell = \sigma_{i,j_{t+1},\ell} - \sum_{u=0}^{t} \gamma_u \sigma_{i,j_u,\ell}$. □

A.2 Soundness of the Multi-Dealer/Muti-Secret Protocol

Turning to the analysis of the protocol from Fig. 3, we begin by proving in Theorem A.2 a simple soundness property of this protocol. Then we describe an ideal functionality that captures the security properties of this protocol. Finally we use Theorems A.1 and A.2 to prove that the protocol from Fig. 3 realizes our ideal functionality.

For soundness, we prove that a shareholder P_j that wasn't disqualified during the sub-protocol for generating public keys, must "know" $z'_{j\ell} \in \mathbb{Z}_q$ for all $\ell \in [[m]]$ such that $Z'_{j\ell} = z'_{j\ell} G_\ell$, and moreover it holds that $z'_{j\ell} = \sum_{i\in Q} \lambda_i \sigma_{ij\ell}$.

Lemma A.2. *Fix an adversary $\mathcal{A}$ and for all $j \in [n]$ denote by $\epsilon_j = \Pr[P_j \in Q']$. (I.e. the probability that the shareholder P_j is not disqualified.)*

There exists an efficient extractor E such that for any $j \in [n]$ with noticeable ϵ_j, E extracts from P_j with overwhelming probability values $\sigma_{j\ell}$ (for all $\ell \in [[m]]$) that satisfy $Z_{i\ell} = \sigma_{i\ell} G_\ell$, where $Z_{j\ell}$ are the ones that P_j broadcasted in Step 6.

Moreover, for every $i \in Q$ and $\ell \in [[m]]$, define $\sigma^*_{ij\ell} = f_{i\ell}(j)$ where $f_{i\ell}(\cdot)$ is the degree-t polynomial on which all the shares of the honest parties lie (cf. Theorem A.1). Then the extracted values satisfy $\sigma_{j\ell} = \sum_{i\in Q} \lambda_i \sigma^*_{ij\ell}$.

Proof. Fix some $j \in [n]$ with noticeable ϵ_j. By Theorem A.1, for every $i \in Q$ and $\ell \in [[m]]$, the shares of all the honest shareholders lie on a degree-t polynomial, except with a negligible probability. Denote that polynomial by $f_{i\ell}(\cdot)$.

Let us set $\sigma^*_{ij\ell} = f_{i\ell}(j)$ for the shareholder of interest P_j, even if it is dishonest. These let us define the values that P_j would have computed in the protocol if it was honest, namely $C^*_{ij} = \sum_{\ell=0}^m \sigma^*_{ij\ell} G_\ell$, $\sigma^*_{j\ell} = \sum_{i\in Q} \lambda_i \sigma^*_{ij\ell}$, $Z^*_{j\ell} = \sigma^*_{j\ell} G_\ell$, and $Z^*_j = \sum_{\ell=0}^m Z^*_{j\ell} = \sum_{\ell=0}^m \sigma^*_{j\ell} G_\ell$. All these values depend only on the honest shareholders and can be recovered from their state. In particular, we can recover a representation of Z^*_j in the bases G_ℓ from the state of the honest parties.

Let $F_i(\cdot)$ be the polynomial that agrees with the $f_{i\ell}(\cdot)$'s that were determined by the state of the honest parties (say, wrt the generator G_0). Namely $C_{ij'} = F_i(j')G_0$ for every honest party, which implies that $C_{ij} = F_i(j)G_0$ also for the shareholder of interest P_j. This is due to the linearity test in the Accusations step, and since there are more than t honest parties, that all the values $C_{ij'}$ lie on a degree-t polynomial (if D_i was not disqualified). Of course, the same holds for C^*_{ij} by definition, so $C_{ij} = C^*_{ij}$ whether or not P_j is honest.

Next, denote $Z_j = \sum_{\ell=0}^m Z_{j\ell}$, and recall that everyone checks during Shareholder-disqualification that $Z_j = \sum_{i\in Q} \lambda_i C_{ij}$. If P_j is not disqualified, then it must satisfy that equality. At the same time, we also have $Z^*_j = \sum_{i\in Q} \lambda_i C^*_{ij} = \sum_{i\in Q} \lambda_i C_{ij} = Z_j$. We conclude that if P_j is not disqualified then $Z_j = Z^*_j$, regardless of whether or not it is honest.

Since P_j successfully proved knowledge of the discrete logs of the $Z_{j\ell}$ that it broadcasted, the knowledge-extractor can extract these values from it. Denote the extracted values by $\sigma_{j\ell}$. Hence, $Z_j = \sum_{\ell=0}^m Z_{j\ell} = \sum_{\ell=0}^m \sigma_{j\ell} G_\ell$, which also yields a representation of $Z_j = Z^*_j$ in the bases G_ℓ.

Under the hardness of DL in $\mathbb{G}$, these two representations of Z_j must be the same, except with a negligible probability. Namely, with overwhelming probability we have $\sigma_{j\ell} = \sigma^*_{j\ell} = \sum_{i\in Q} \lambda_i \sigma^*_{ij\ell}$, as needed. □

A.3 Proof of Simulation

Theorem A.3. *The protocol from Fig. 3 realizes the MultiVSS functionality $\mathcal{F}_{MultiVSS}$.*

Proof. On a high level, the simulator will mostly run the honest-dealer code pretending that the secrets are all zero, and we will show that the resulting view of the adversary is what it needs to be. In more detail, the simulator $\mathcal{S}$ interacts with the functionality and the real-world adversary $\mathcal{A}$ as depicted in Fig. 4. The simulation consists of the following steps:

Sharing. The simulator begins by sending to $\mathcal{A}$ all the shares $\sigma_{ij\ell}$ on behalf of good dealers $i \notin \mathcal{B}$ to bad shareholders $j \in \mathcal{B}'$ (with $\ell \in [[m]]$). It also sends all the column check values C_{ij} of the good dealers $i \notin \mathcal{B}$ (for $j \in [n]$). To do this, the simulator simply runs the honest protocol code on behalf of all the good dealers, pretending that the original secrets are all zero, $s'_{i\ell} = 0$ for all $i \notin \mathcal{B}$ and $\ell \in [[m]]$.

We prove in Theorem A.4 that the view of any set $\mathcal{B}'$ of cardinality $\leq t$ is independent of the original secrets that were used in the Dealing step, hence the view that $\mathcal{S}$ generates is the correct one.

Once this is done, $\mathcal{A}$ responds with the shares $\sigma_{ij\ell}$ and column values C_{ij} of the bad dealers $i \in \mathcal{B}$.

Accusations and responses. Next the simulator needs to send the accusations messages, and respond to the accusations from $\mathcal{A}$. Here too it just follows the prescribed protocol using the state from the previous step.

Note that the accusations that the simulator sends are a deterministic function of the shares and column values that it received from $\mathcal{A}$, and the response to accusations from $\mathcal{A}$ consists only of broadcasting some the same shares $\sigma_{ij\ell}$ that it already sent to $\mathcal{A}$. Hence there is no new information that $\mathcal{A}$ gets in this step.

Qualified dealers and adversarial input. The simulator then computes the qualified dealer set Q as in the protocol. This is a deterministic function of the communication between $\mathcal{S}$ and $\mathcal{A}$ so far. Let $P = Q \cap \mathcal{B}$ be the set of qualified dealers that are controlled by the adversary.

If the shares received from $\mathcal{A}$ for some dealer $i \in P$ do not all lie on degree-t polynomials (after accusations and response), then the simulator aborts. By Theorem A.1, this only happens with a negligible probability.

Otherwise, the simulator interpolates the original secrets $s_{i\ell}$ of the dealers $i \in P$. The simulator sends P to the functionality, as well as the secrets $s_{i\ell}$ of all $i \in P$ and all the shares $\sigma_{ij\ell}$ of the shareholders $j \in B'$ (and $\ell \in [m]$).

Public values. Next, the simulator gets from the functionality the public values $Z_{j\ell}$ for the honest shareholders $j \notin \mathcal{B}$ and $\ell \in [m]$. It then needs to compute also the values Z_{j0} (corresponding to $\ell = 0$) so as to ensure a consistent transcript. This is done as follows:

Let $\vec{\lambda} = L(Q)$. For all j, the simulator uses the column values C_{ij} for $i \in Q$ to compute $Z_j \sum_{i\in Q}^k \lambda_i C_{ij}$, and then $Z_{j0} = Z_j - \sum_{\ell=1}^m Z_{j\ell}$.

By Theorem A.5, the values of $Z_{j\ell}$, including Z_{j0}, are consistent with everything the entire view of $\mathcal{A}$ in the Dealing step above, and only depend on the target secrets (and not on the original secrets that went into computing them).

Next, $\mathcal{S}$ uses the NIZK-simulator to construct valid proofs of knowledge of DL for all the $Z_{j\ell}$'s, $j \notin \mathcal{B}'$ $\ell \in [[m]]$, and sends to $\mathcal{A}$. The adversary $\mathcal{A}$ replies with $Z_{j\ell}$'s and NIZK proofs for $j \in \mathcal{B}'$.

Qualified shareholders and output. The simulator $\mathcal{S}$ verifies the $Z_{j\ell}$'s and associated NIZK proofs as in the protocol to get the qualified set of shareholders $\mathcal{Q}'$. It sends to the functionality the set $P' = \mathcal{Q}' \cap \mathcal{B}'$, and the functionality returns the appropriate output to each qualified shareholder $j \in \mathcal{Q}'$.

We relied on Theorem A.4 to prove that the view of $\mathcal{A}$ is consistent with the honest dealers' inputs and with the $Z_{j\ell}$'s (and hence also the shares) of the honest shareholders. But we still must argue that the shares $\sigma_{j\ell}$ (and public values $Z_{j\ell}$) for *dishonest shareholders* $j \in P'$ are also consistent. Here we rely on Theorem A.2, it asserts exactly that the $Z_{j\ell}$'s that $\mathcal{A}$ sends are indeed consistent with the correct shares (called $\sigma^*_{j\ell}$ in the proof of that lemma), which is what the functionality returns. □

A.4 Secrecy

The technical lemmas that we prove next say that (a) the adversary's view of the Dealing step is independent of the original secrets of the honest dealers, and (b) its view in the Shareholder verification step only depends on the original secrets via the resulting target secrets. Let us first establish some notations.

Denote by $\mathcal{H} = [n'] \setminus \mathcal{B}$ the set of honest dealers, of cardinality $|\mathcal{H}| = h$. Let $\mathcal{B}' \subset [n]$ be an arbitrary set of t shareholders (e.g., the set of bad shareholders). The view of the shareholders in $\mathcal{B}'$ of the Dealing step of the dealers in $\mathcal{H}$ consists of all their shares, $\sigma_{ij\ell}$ for $i \in \mathcal{H}, j \in \mathcal{B}', \ell \in [[m]]$, and all the column check values of these dealers, C_{ij} for $i \in \mathcal{H}, j \in [n]$.

Let $V_{\mathcal{B}'}$ be one such fixed view, we say that it is *consistent* if it is consistent with some set of original secrets, $\{s_{i\ell} : i \in \mathcal{H}, \ell \in [m]\}$. Our first lemma asserts that a consistent view $V_{\mathcal{B}'}$ is consistent with all possible sets of original secrets, and is equally likely to come from any of them.

Lemma A.4. *Fix any consistent view $V_{\mathcal{B}'}$ and any set of original secrets $\{s_{i\ell} : i \in \mathcal{H}, \ell \in [m]\}$. Then there is one unique setting for the matrices $\{\mathbb{A}_i : i \in \mathcal{H}\}$ which is consistent with the secrets $s_{i\ell}$ in their respective rows, that would result in the view $V_{\mathcal{B}'}$.*

Proof. As $V_{\mathcal{B}'}$ is a consistent view, there is some set of original secrets $\{s^*_{i\ell} : i \in \mathcal{H}, \ell \in [m]\}$ which is consistent with it. For every i and $\ell \geq 1$, the t shares $\sigma_{ij\ell}, j \in \mathcal{B}'$, together with the original secret $s^*_{i\ell}$, uniquely define the polynomials $f^*_{i\ell}$ and therefore the rows $\ell = 1, 2, \ldots, m$ of the matrices $\mathbb{A}_i$. Together with (the DL of) the column check values C_{ij}, this also uniquely define the top rows of these matrices ($\ell = 0$), as follows:

Let c_{ij}'s be the discrete-logarithm of the C_{ij}'s to the generator G_0, namely $C_{ij} = c_{ij} G_0$. Also let c_i be the polynomial such that $c_i(j) = \sum_k c_{ij}$. $c_{(\cdot)}$ is a degree-t polynomial, since the C_{ij}'s pass the linearity test. The only setting of the top row of $\mathbb{A}_i$ which is consistent with all the other rows and with the C_{ij}'s is the polynomial $f^*_{i0} = c_i - \sum_{\ell=1}^m u_\ell \cdot f^*_{i\ell}$, where u_i is the discrete-logarithm of G_ℓ to the base G_0.

Hence we now have a unique setting of the matrices $\mathbb{A}_i$ which is consistent with the shares of $j \in \mathcal{B}'$ for rows $\ell \geq 1$ and the column check values C_{ij}. What makes the view $V_{\mathcal{B}'}$ consistent is that also the shares σ_{ij0} for row 0 agree with these polynomials, specifically $\sigma_{ij0} = f^*_{i0}(j)$.

Consider now any other set of original secrets $\{s'_{i\ell} : i \in \mathcal{H}, \ell \in [m]\}$, and we show that there is also a unique setting of the matrices $\mathbb{A}_i$ which is consistent with $V_{\mathcal{B}'}$ and these shares. Denote by $\Lambda(\cdot)$ the unique degree-t polynomial such that $\Lambda(j) = 0$ for all $j \in \mathcal{B}'$ and $\Lambda(0) = 1$. Also let $\delta_{i\ell} = s'_{i\ell} - s^*_{i\ell}$.

The main technical observation in this lemma is that the only setting for row ℓ in $\mathbb{A}_i$ which is consistent with the shares $\sigma_{ij\ell}$ and the secret $s'_{i\ell}$, is the polynomial $f'_{i\ell} = f^*_{i\ell} + \delta_{i\ell} \cdot \Lambda$. Indeed this polynomial has $f'_{i\ell}(j) = f^*_{i\ell}(j) + \delta\Lambda(j) = f^*_{i\ell}(j) = \sigma_{ij\ell}$ for all $j \in \mathcal{B}'$, and also $f'_{i\ell}(0) = f^*_{i\ell}(0) + \delta\Lambda(0) = f^*_{i\ell}(0) + \delta = s'_{i\ell}$.

Together with the polynomial c_i, this again uniquely define the row-0 polynomial

$$\begin{aligned} f'_{i0} &= c_i - \sum_{\ell=1}^m u_\ell \cdot f'_{i\ell} = c_i - \sum_{\ell=1}^m u_\ell \cdot (f^*_{i\ell} + \delta_{i\ell} \cdot \Lambda) \\ &= c_i - \left(\sum_{\ell=1}^m u_\ell \cdot f^*_{i\ell} \right) - \Lambda \cdot \sum_{\ell=1}^m u_\ell \delta_{i\ell} \ = \ f^*_{i\ell} - \Lambda \cdot \rho_i \end{aligned}$$

(where $\rho_i = \sum_{\ell=1}^m u_\ell \delta_{i\ell} \in \mathbb{Z}_q$). This means that $f'_{i0}(\cdot)$ is also consistent with the shares σ_{ij0}, since $\Lambda(j) = 0$ for $j \in \mathcal{B}'$, and so $f'_{i0}(j) = f^*_{i0}(j) - \Lambda(j) \cdot \rho_i = f^*_{i0}(j) = \sigma_{ij0}$. □

Next, fix the randomness of the adversary and a particular view $V_{\mathcal{B}'}$ of the bad shareholders in the Dealing step as above. These in turn fix the adversary's messages, and therefore also the set of qualified shareholders $\mathcal{Q}$ and the linear combination $\vec{\lambda} = L(\mathcal{Q})$.

Lemma A.5. *Fix a view $V_{\mathcal{B}'}$ of the bad shareholders and the adversary's randomness, and consider two sets of honest-dealers' original secrets $\{s_{i\ell}\}$ and $\{s'_{i\ell}\}$ that agree on all the target secrets. Namely $\sum_{i \in \mathcal{H}} \lambda_i s_{i\ell} = \sum_{i \in \mathcal{H}} \lambda_i s'_{i\ell} = z_\ell$ for all $\ell \in [m]$.*

Conditioned on $V_{\mathcal{B}'}$, the view of the adversary in the subprotocol for "Shareholder verification values" is identical for both sets of original secrets.

Proof. The messages of the honest shareholders in the "Shareholder verification values" depend only on their shares of the target secrets z_ℓ, $\ell = 0, 1, \ldots, m$. So we need to show that conditioned on $V_{\mathcal{B}'}$, these shares are the same for both sets of original secrets.

We note that since the adversary's randomness and view in the Dealing protocol is fixed, then the original-secret shares that honest shareholders get from the adversary are the same, regardless of the original secrets of the honest dealers. But of course, the original-secret shares that they get from the honest dealers will not be the same.

Let $\mathbb{A}_i$ be the unique matrices consistent with the original secrets $s_{i\ell}$ and the view $V_{\mathcal{B}'}$, and $f_{i\ell}$ be the polynomial in the ℓ'th row of $\mathbb{A}_i$. Similarly, $\mathbb{A}'_i$ is the unique matrices consistent with the original secrets $s'_{i\ell}$ and the view $V_{\mathcal{B}'}$, and $f'_{i\ell}$ be the polynomial in the ℓ'th row of $\mathbb{A}_i$

Recall from before that $f_{i0} = a_i - \sum_{\ell=1}^{m} u_\ell f_{i\ell}$ and $f'_{i0} = a_i - \sum_{\ell=1}^{m} u_\ell f'_{i\ell}$ for some polynomials a_i and scalars u_ℓ. Also as before, let $\delta_{i\ell} = s_{i\ell} - s'_{i\ell}$ for $\ell \geq 1$, and recall that $f'_{i\ell} = f_{i\ell} + \delta_{i\ell} \cdot \Lambda$ (for some polynomial Λ). Also, since the two sets of original secrets agree on all the target secrets, then we know that $\sum_{i \in \mathcal{H}} \lambda_i \delta_{i\ell} = 0$ for all $\ell \geq 1$.

We first observe that for $\ell \geq 1$, not just the target secret z_ℓ but also its sharing is identical for the two sets of original secrets. Indeed, the secret z_ℓ is shared via the polynomial $g_\ell = \sum_{i \in Q} \lambda_i f_{i\ell}$ for the first set of original secrets and via the polynomial $g'_\ell = \sum_{i \in Q} \lambda_i f'_{i\ell}$ for the second, and

$$\begin{aligned} \forall \ell \geq 1,\ g'_\ell - g_\ell \quad &= \quad \sum_{i \in Q} \lambda_i (f'_{i\ell} - f_{i\ell}) = \sum_{i \in Q} \lambda_i \delta_{i\ell} \cdot \Lambda \\ &\overset{(a)}{=} \quad \Lambda \cdot \left(\sum_{i \in \mathcal{H}} \lambda_i \delta_{i\ell} \right) \overset{(b)}{=} 0. \end{aligned} \qquad (3)$$

Equality (a) above holds since the original secrets of the adversary are the same in both cases (so $\delta_{i\ell} = 0$ for $i \notin \mathcal{H}$), and equality (b) holds since the parenthesized expression sums up to zero.

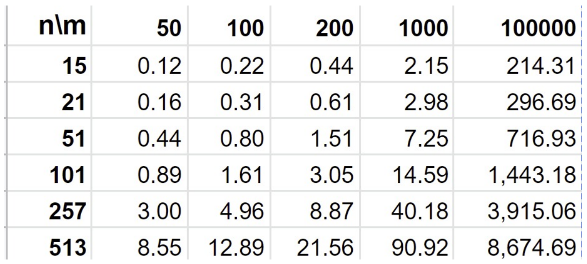
The only thing left to check is that the sharing of the additional "target secrets" z_0 and z'_0 for $\ell = 0$ are also the same for both sets of original secrets. These are shared via the polynomials $g_0 = \sum_{i \in Q} \lambda_i f_{i0}$ and $g'_0 = \sum_{i \in Q} \lambda_i f'_{i0}$, respectively. Hence

$$\begin{aligned} g'_0 - g_0 \quad &= \quad \sum_{i \in Q} \lambda_i (f'_{i0} - f_{i0}) \\ &= \quad \sum_{i \in Q} \lambda_i \Big(\big(a_i - \sum_{\ell=1}^{m} u_\ell f'_{i\ell}\big) - \big(a_i - \sum_{\ell=1}^{m} u_\ell f_{i\ell}\big) \Big) \\ &= \quad \sum_{\ell=1}^{m} u_\ell \sum_{i \in Q} \lambda_i (f'_{i\ell} - f_{i\ell}) = 0, \end{aligned}$$

where the last equality follows from Eq. (3). □

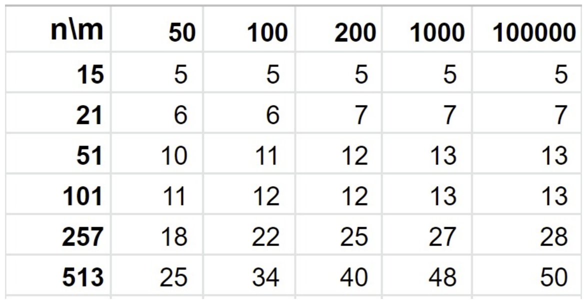
B MICRO-BENCHMARKING FOR THE NON-YOSO MULTIVSS PROTOCOL

We provide in Figs. 6 and 7 some performance results for the non-YOSO MultiVSS protocol from Fig. 3. These results were obtained by running micro-benchmarking on this protocol. As we can see, our protocol provides significant speedup as compared to a basic implementation of the classical Pedersen VSS protocol.



n\m	50	100	200	1000	100000
15	0.12	0.22	0.44	2.15	214.31
21	0.16	0.31	0.61	2.98	296.69
51	0.44	0.80	1.51	7.25	716.93
101	0.89	1.61	3.05	14.59	1,443.18
257	3.00	4.96	8.87	40.18	3,915.06
513	8.55	12.89	21.56	90.92	8,674.69

Figure 6: Single-threaded computation per party (seconds) for the MultiVSS protocol, with n dealers and m secrets per dealer.



n\m	50	100	200	1000	100000
15	5	5	5	5	5
21	6	6	7	7	7
51	10	11	12	13	13
101	11	12	12	13	13
257	18	22	25	27	28
513	25	34	40	48	50

Figure 7: The speedup factor of the MultiVSS protocol against naive Pedersen, with n dealers and m secrets per dealer.