Publicly Verifiable Secret Sharing over Class Groups and Applications to DKG and YOSO

Ignacio Cascudo[1]* and Bernardo David[2]**

[1] IMDEA Software Institute, Madrid, Spain. ignacio.cascudo@imdea.org
[2] IT University of Copenhagen, Copenhagen, Denmark. bernardo@bmdavid.com

Abstract. Publicly Verifiable Secret Sharing (PVSS) allows a dealer to publish encrypted shares of a secret so that parties holding the corresponding decryption keys may later reconstruct it. Both dealing and reconstruction are non-interactive and any verifier can check their validity. PVSS finds applications in randomness beacons, distributed key generation (DKG) and in YOSO MPC (Gentry *et al.* CRYPTO'21), when endowed with suitable publicly verifiable re-sharing as in YOLO YOSO (Cascudo *et al.* ASIACRYPT'22).

We introduce a PVSS scheme over class groups that achieves similar efficiency to state-of-the art schemes that only allow for reconstructing *a function* of the secret, while our scheme allows the reconstruction of the original secret. Our construction generalizes the DDH-based scheme of YOLO YOSO to operate over class groups, which poses technical challenges in adapting the necessary NIZKs in face of the unknown group order and the fact that efficient NIZKs of knowledge are not as simple to construct in this setting.

Building on our PVSS scheme's ability to recover the original secret, we propose two DKG protocols for discrete logarithm key pairs: a biasable 1-round protocol, which improves on the concrete communication/computational complexities of previous works; and a 2-round unbiasable protocol, which improves on the round complexity of previous works. We also add publicly verifiable resharing towards anonymous committees to our PVSS, so that it can be used to efficiently transfer state among committees in the YOSO setting. Together with a recent construction of MPC in the YOSO model based on class groups (Braun *et al.* CRYPTO'23), this results in the most efficient full realization (*i.e.* without assuming receiver anonymous channels) of YOSO MPC based on the CDN framework with transparent setup.

1 Introduction

Publicly Verifiable Secret Sharing [37] (PVSS) allows for a dealer to publish encrypted secret shares in such a way that any verifier can check their validity. Moreover, after the parties holding the corresponding decryption keys reconstruct the secret, any verifier can also check the secret's validity with respect to the encrypted shares (typically by checking the consistency between the encrypted and plaintext shares used for reconstruction). Many PVSS schemes are known [23,36,4,35,30,8,9,26], but the state-of-the-art constructions [10] based on number theoretic assumptions only allow for reconstructing g^s, where $g \in \mathbb{G}$ is the generator of a cyclic group $\mathbb{G}$ and $s \in \mathbb{Z}_p$ is the secret. This limitation can be circumvented [11] by sharing a random secret s' with the PVSS and publishing a one-time pad of the actual secret s with a key derived (e.g. via a random oracle) from the reconstructable secret $g^{s'}$. However, this solution limits the efficiency of a number of PVSS applications. In particular, the secret sharing scheme derived in this way is no longer linear.

Distributed Key Generation (DKG). Besides randomness beacons (*e.g.* [8,9]), one of the main applications of PVSS schemes is in constructing Distributed Key Generation (DKG) protocols. Such

* Ignacio Cascudo was partially supported by the Spanish Government under the project SecuRing (ref. PID2019-110873RJ-I00) and the PRODIGY Project (TED2021-132464B-I00), both funded by MCIN/AEI/10.13039/501100011033/. PRODIGY is also funded by the European Union NextGenerationEU/PRTR. He was also partially supported by the European Union under GA 101096435 (CONFIDENTIAL-6G). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

** Bernardo David was supported by the Independent Research Fund Denmark (IRFD) grant number 0165-00079B.

protocols [33,22,32,24,29,28] allow for parties to obtain Shamir shares sk_i of a secret key $\mathsf{sk} \in \mathbb{Z}_p$ and the corresponding public key g^{sk} while revealing nothing else. The recent unbiasable DKG protocol of [11] builds on the PVSS scheme of [9] to achieve higher efficiency than previous protocols in terms of round/computational complexities (and in many cases [22,24,28] also better communication complexity). However, even though it requires only 2 rounds in case there is no cheating, it still falls short of round optimality [32] in case a malicious party triggers a dispute phase that requires 2 extra rounds. This issue stems from the fact that, in order to allow the parties to retrieve s_i, the DKG of [11] must publish a separate encryption of shares s_i apart from the original PVSS [9] encrypted shares, since those can only be reconstructed to g^{s_i}. In case the PVSS encrypted shares are not consistent with the extra encryption, the dispute phase must be triggered to avoid bias.

The YOSO model. The recent introduction of the You Only Speak Once (YOSO) model for multiparty computation (MPC) protocols [25] and related models [17,1,19] has sparked a renewed interest in PVSS schemes with added properties. In the YOSO model, each round of the protocol is executed by a fresh randomly selected committee of parties who remain anonymous until they send their first message, after which they no longer participate in the execution. This is interesting as it improves scalability because small committees are sufficient to execute each round, as well as resulting in protocols resistant to adaptive corruptions, given that the adversary does not know who to corrupt. However, due to the ephemeral nature of these committees, each of them must transfer their secret state to the next, which is hard given their anonymity. In the YOSO model, it is assumed that all parties have access to ideal receiver anonymous communication channels (RACC), which allow for sending messages to an anonymous party to be randomnly chosen at a later point. Hence, protocols in the YOSO model assume RACCs as setup but aim at minimizing their use. In particular, it was observed in [25] that the Cramer-Damgård-Nielsen (CDN) [18] approach to MPC via threshold encryption is particularly well suited to this setting, as secret key shares are the only secret state maintained by parties. Only very recently, Braun *et al.* proposed a YOSO MPC protocol [6] following the CDN approach without assuming pre-distribution of secret key shares as trusted setup. This protocol assumes access to ideal RACCs in order to realize a threshold encryption scheme over class groups with a matching DKG and a protocol for re-sharing the secret key at every round.

PVSS in the YOSO model. A number of tools [3,27,7] have been proposed to implement RACCs but only recently an efficient publicly verifiable (re-)sharing scheme compatible with such techniques was proposed in YOLO YOSO [10]. The YOLO YOSO scheme allows for parties to share secrets by publishing publicly verifiable encrypted shares and then re-share those secrets into a fresh set of shares for the next anonymous committee without assuming access to an ideal RACC, thus providing a way to realize the communication infrastructure of the YOSO model using only a random oracle and a Public Key Infrastructure (PKI) as setup (*i.e.* a transparent setup). However, besides suffering from the issue that only g^s can be reconstructed from encrypted shares of s, YOLO YOSO is based on DDH and not directly compatible with class groups. Hence, if YOLO YOSO was used to realize the RACC setup required in the protocol of [6] one would need to rely on freakishly large groups where DDH is hard and be prepared to rely both on DDH and on hardness assumptions over class groups.

1.1 Our Contributions

We introduce an efficient PVSS scheme based on class groups that allows for reconstructing the original secret, enabling applications to DKG and YOSO MPC. Our main results are summarized as follows:

PVSS over class groups: We construct a PVSS scheme over class groups [15] that allows for reconstructing the original secret achieving similar efficiency[3] as previous works [9,10] that only allowed for recovering functions of the secret. Moreover, our scheme achieves a stronger security guarantee. In addition to this, privacy is based solely on the DDH-f assumption [16], known to be implied by both DDH and hard subgroup membership on class groups.

Efficient NIZKs of encrypted share validity: Our design differs from the schemes of [9,10], overcoming the hurdles of avoiding extracting witnesses and adapting the SCRAPE [8] share validity test to the class group setting.

[3] Up to a constant due to the time for group operations and size for group elements in class groups being higher than those for DDH-hard groups based on elliptic curves.

DKG protocols for Discrete Logarithm key pairs: We show how our PVSS can be used to construct a 1-round biasable DKG protocol that outperforms the state-of-the-art [28]. We also construct a 2-round unbiasable protocol which is round-optimal [32], improving on the state-of-the-art [11].

Full realization of Efficient YOSO MPC with transparent setup: Our PVSS can be endowed with publicly verifiable re-sharing towards anonymous committees, lifted from YOLO YOSO [10] via our new NIZKs of share validity. Using this efficient realization of the RACC setup needed by the communication-efficient protocol of [6] yields the most efficient full realization (*i.e.* without assuming ideal RACCs) of YOSO MPC with transparent setup based solely on class groups.

When constructing our PVSS scheme, we face the main technical hurdle of constructing an efficient NIZK of share validity over class groups. Similarly to [9,10], we start from Shamir's secret sharing, encrypt the shares (using encryption over class groups) and want to apply the SCRAPE [8] test to verify share validity. However, we cannot apply the SCRAPE test directly, since the security analysis of this technique crucially relies on the group order, which is unknown for class groups. Moreover, current techniques [12,13,6] for zero knowledge proof systems over class groups do not allow for efficient proofs of knowledge for complex relations such as that of share validity via the SCRAPE test. In order to overcome both of these difficulties we make the following main technical contributions: 1. a new analysis of the SCRAPE test for encrypted shares over groups of unknown order (*i.e.* class groups); 2. a new efficient NIZK proof of share validity (but *not of knowledge*) based on the SCRAPE test for Shamir shares encrypted over class groups; 3. a new proof strategy for our PVSS scheme based on a NIZK that does not allow for extracting adversarial shares.

We construct our 2-round unbiasable DKG protocol as a direct application of our new PVSS protocol. First, all parties publish encrypted shares of random secrets along with proofs of share validity, which are checked so that invalid share vectors and their creators are ignored. Next, honest parties decrypt the shares they received and combine them to generate their share of the secret key and a partial public key, which is published along with a correctness proof so that all parties may compute the final public key. While a similar approach was taken in Mt. Random [11], that DKG requires two extra rounds in case of cheating. The ALBATROSS [9] PVSS used in [11] only allows parties to share g^{s_i}, not the original share s_i, requiring an extra ciphertext containing s_i to be published. When the share in the separate ciphertext differs from the PVSS encrypted share, a dispute phase consisting of 2 extra rounds is executed. We eliminate the dispute phase and achieve a round-optimal [32] protocol by relying on the fact that we can recover the original s_i in our new PVSS scheme's encrypted shares.

Our 1-round DKG protocol publishes the information needed for computing public key shares and the final public key along with the encrypted shared of our PVSS scheme. Doing so avoids the need for the second round where this information is revealed but allows for an adversary to bias the public key by observing the shares published by the honest parties before publishing its own shares. While this bias is unavoidable in 1-round DKG protocols [32], it does not pose a problem when DKG is used in many applications (see e.g. [29]). Our approach cannot be implemented by a simple modification of our 2-round protocol, since it is necessary to prove consistency between encrypted shares used to derive the secret key and public shares used to derive the public key. In order to do so, we design an efficient NIZK proving this relation for our PVSS scheme. Our 1-round protocol requires computing less group operations and communicating less group elements than the work of Kate *et al.* [31], which in turn is shown to be more efficient than the Groth [28] DKG. Hence, our 1-round DKG improves on the concrete efficiency of [31,28].

Another application of our new PVSS scheme is in efficiently realizing publicly verifiable (re-)sharing towards anonymous committees selected at random, which is crucial in the YOSO model. We adapt our techniques for proving encrypted share validity over class groups to obtain an efficient NIZK of encrypted re-sharing data validity. We remark that efficiently and non-interactively proving re-sharing validity is the main hurdle when constructing secret sharing schemes for the YOSO model, where all parties must do re-sharing at every round. This extended PVSS with re-sharing can be combined with the shuffle-based encryption to the future scheme from YOLO YOSO [10], which only requires a publicly verifiable mixnet, known to be realizable with proofs of shuffle correctness [2] for linearly homomorphic encryption schemes (*e.g.* based on class groups [15]). The resulting publicly verifiable secret (re-)sharing scheme towards anonymous committees implements the communication infrastructure needed for the efficient YOSO MPC protocol proposed in [6], which follows the CDN [18] approach to reduce the size of secret state transferred among anonymous committees to a minimum and is based on class groups to achieve transparent setup. However, our solution does not require assuming ideal receiver anonymous communication channels (RACC) as in [6], while improving on the efficiency of the proof of resharing, which in [6] requires executing two instances of an inefficient PVSS-like protocol. Hence, combining our results with the protocol of [6] yields the most efficient full realization of YOSO MPC with transparent setup.

1.2 Related Works

Cryptography over Class Groups: The Castagnos-Laguillaumie (CL) framework for encryption based on class groups was introduced in [15] and later refined in [6,12,13,14,16,38]. This framework creates a finite group of unknown order where the discrete logarithm is assumed hard to compute, together with a cyclic subgroup where the discrete logarithm is actually easy. This allows for constructing additively homomorphic ElGamal-style encryption, where it is possible to encode plaintexts into the group where the discrete logarithm is easy, compute linear operations on multiple ciphertexts and obtain the result m instead of g^m.

Publicly Verifiable Secret Sharing: Many PVSS schemes based on different techniques for proving share validity are known [37,23,36,4,35,30]. SCRAPE [8] was the first scheme to achieve $O(n)$ complexity for share validity verification, allowing for executions with tens of thousands of parties. ALBATROSS [9] built on the SCRAPE techniques to construct a compact NIZK for share validity and also achieved sharing of large batches of secrets. This NIZK was generalized and further improved in YOLO YOSO [10], where support for re-sharing and anonymous committees was also efficiently achieved for the first time. Recently, Mt. Random [11] extended ALBATROSS, showing how to slowly release sub-batches of secrets. While these previous works build on number theoretical assumptions, an efficient PVSS scheme from lattice-based assumptions is constructed in [26].

Distributed Key Generation: Most DKG protocols use secret sharing in a similar way as ours, the key difference being how parties prove the correctness of their shares and public information. The classic DKG by Pedersen [33], employs Feldman's VSS, resulting in a protocol with 1 round in case of no disputes, and 2 extra rounds if there are disputes. Fouque and Stern [22] proposed a one-round DKG based on the Paillier cryptosystem that still allows the adversary to bias public keys. Groth [28] proposed a 1-round protocol based on pairings. Recently, Katz [32] showed that all 1-round protocols are biasable and proposed round-optimal protocols. Gennaro *et al.* [24] were the first to observe that Pedersen's DKG is biased and made it unbiasable by introducing a new round of interaction and a new round of dispute resolution. Gurkan *et al.* [29] introduces a pairing-based DKG based on the notion of aggregation via gossip. Cascudo *et al.* [11] introduce the Mt. Random DKG, which follows a similar approach as our constructions but is based on the ALBATROSS [9] PVSS, requiring 2 extra conflict resolution rounds to avoid bias. Recently, Kate *et al.* introduced a DKG based on class groups improving on the performance of Groth [28].

YOSO MPC: The original YOSO MPC model and the first constructions were introduced in [25], while similar models with less stringent restrictions on interaction and matching protocols were introduced in Fluid MPC [17] and in SCALES [1]. A similar model without anonymity but stricter interaction restrictions and matching protocols were introduced in [19]. Further protocols for the Fluid MPC and YOSO MPC models were proposed in [34] and [6], respectively. Suitable receiver anonymous communication channels for original YOSO model (where parties remain anonymous until they act) were first constructed in [3] and [27], respectively suffering from a low corruption threshold (less than 1/4 of parties) and from high complexity. Towards solving this issue, the notion of Encryption to the Future (EtF) was introduced in [7] and efficient DDH-based EtF schemes with matching PVSS (and re-sharing) were introduced in [10].

Independent Work: Several 2-round (*i.e.* round optimal) unbiasable DKG protocols based on generic secret sharing, encryption and NIZK schemes are proposed in [32]. However, the core technical issue of obtaining efficient concrete instantiations is not addressed. In [31], the authors propose 1-round (biasable) and 2-round (unbisable) DKG protocols based on "leaky" non-interactive VSS (NI-VSS) protocol. This NI-VSS achieves a weaker security notion than our PVSS as it leaks information about the secret, which is shown to be sufficient for their DKG constructions but is clearly insufficient for general use (*e.g.* our YOSO MPC application). Moreover, the NIZK of share validity of [31] is based on a NIZK of exponent knowledge and requires more communication/computation than our NIZKs, which circumvent the need for extracting witnesses.

2 Preliminaries

For $m, n \in \mathbb{Z}$, we denote $[m, n] := \{m, m+1, \ldots, n\}$. Moreover, we write $[n] = [1, n] = \{1, \ldots, n\}$. For a finite set S, we denote by $x \leftarrow_{\$} S$ the selection of a uniformly random element in S. If we are sampling from a non-necessarily uniform distribution $\mathcal{D}$, then we write $x \leftarrow \mathcal{D}$. In this paper q will always denote a prime number and then $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$ is the field of integers modulo q. $\mathbb{Z}_q[X]_{\leq t}$ denotes the set of polynomials in $\mathbb{Z}_q[X]$ of degree at most t. Let $S \subseteq \mathbb{N}$ a finite set and $A = \{\alpha_i : i \in S\}$ be a set of pairwise distinct points contained in a field $\mathbb{F}$. For $i \in S$, we define the Lagrange interpolation polynomial $\mathrm{Lag}_{i,S,A}(X) := \prod_{j \in S \setminus \{i\}} \frac{X-\alpha_j}{\alpha_i - \alpha_j}$. Recall that $L(X) = \sum_{i \in S} y_i \cdot \mathrm{Lag}_{i,S,A}(X)$ is the unique polynomial in $\mathbb{F}[X]$ of degree at most $|S| - 1$ with $L(\alpha_i) = y_i$ for all $i \in S$.

Relations are written as $\mathcal{R} = \{(x; w) : R(x, w) = 1\}$ where x is the statement, w is the witness and R is some predicate. We write $\mathtt{NIZK}(\mathcal{R})$ (respectively $\mathtt{NIZKPoK}(\mathcal{R})$) to denote a generic non-interactive zero knowledge proof (respectively proof of knowledge) for relation $\mathcal{R}$, without instantiating it at that point.

2.1 Publicly Verifiable Secret Sharing(PVSS)

We first present our definitions of a publicly verifiable secret sharing scheme and security properties, where we mainly adopt the definitions from [10]. After that, we recall the SCRAPE test [8] which has been of great utility in several works on publicly verifiable secret sharing and applications [8,9,10,29].

Model A PVSS scheme consists of the following algorithms.

- *Setup*
  - $\mathsf{Setup}(1^\lambda, \mathsf{ip}) \to \mathsf{pp}$ outputs public parameters pp. The initial parameters ip contain information about number of parties, privacy and reconstruction thresholds and spaces of secrets and shares. The public parameters include a description of spaces of private and public keys SK and PK and the relation $\mathcal{R}_{\mathsf{Key}} \subseteq \mathsf{PK} \times \mathsf{SK}$ describing valid key pairs.
  - $\mathsf{KeyGen}(\mathsf{pp}, id) \to (\mathsf{sk}, \mathsf{pk}, \mathsf{Pf}_{\mathsf{pk}})$, where $(\mathsf{pk}; \mathsf{sk}) \in \mathcal{R}_{\mathsf{Key}}$ and $\mathsf{Pf}_{\mathsf{pk}}$ is a proof meant to assert that pk is a valid public key.
  - $\mathsf{VerifyKey}(\mathsf{pp}, id, \mathsf{pk}, \mathsf{Pf}_{\mathsf{pk}}) \to 0/1$ (as a verdict on whether pk is valid).
- *Distribution*
  - $\mathsf{Dist}(\mathsf{pp}, (\mathsf{pk}_i)_{i \in [n]}, \mathsf{s}) \to ((C_i)_{i \in [n]}, \mathsf{Pf}_{\mathsf{Sh}})$ where $\mathsf{s} \in \mathcal{S}$ is a secret, outputs "encrypted shares" C_i and a proof $\mathsf{Pf}_{\mathsf{Sh}}$ of sharing correctness.
- *Distribution Verification*
  - $\mathsf{VerifySharing}(\mathsf{pp}, (\mathsf{pk}_i, C_i)_{i \in [n]}, \mathsf{Pf}_{\mathsf{Sh}}) \to 0/1$ (as a verdict on whether the sharing is valid).
- *Reconstruction*
  - $\mathsf{DecShare}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{sk}_i, C_i) \to (A_i, \mathsf{Pf}_{\mathsf{Dec}i})$, outputs a decrypted share A_i and a proof $\mathsf{Pf}_{\mathsf{Dec}i}$ of correct decryption.
  - $\mathsf{Rec}(\mathsf{pp}, \{A_i : i \in \mathcal{T}\})$ for some $\mathcal{T} \subseteq [n]$ outputs an element of the secret space $\mathsf{s}' \in \mathcal{S}$ or an error symbol $\perp$.
- *Reconstruction Verification*
  - $\mathsf{VerifyDec}(\mathsf{pp}, \mathsf{pk}_i, C_i, A_i, \mathsf{Pf}_{\mathsf{Dec}i}) \to 0/1$ (as a verdict on whether A_i is a valid decryption of C_i).

Security properties

Correctness with r-reconstruction. The correctness with r-reconstruction requirement ensures that if everybody is honest, then all proofs involved pass and any set of at least r participants can reconstruct the secret from their shares (by first having each party decrypt their share and then jointly applying the reconstruction algorithm Rec).

Definition 1. *For a set $\mathcal{T} \subseteq [n]$, and a probability distribution $\mathcal{D}_{\mathcal{S}}$ over the secret space, define the following experiment $\mathtt{ExpCorr}_{\mathcal{T},\mathcal{D}_{\mathcal{S}}}(1^\lambda)$.*

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{ip})$
- $\forall\ i \in [n], \quad (\mathsf{sk}_i, \mathsf{pk}_i, \mathsf{Pf}_{\mathsf{pk}_i}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, i)$
- $\mathbf{s} \leftarrow \mathcal{D}_{\mathcal{S}}$
- $((C_i)_{i\in[n]}, \mathsf{Pf}_{\mathsf{Sh}}) \leftarrow \mathsf{Dist}(\mathsf{pp}, \{\mathsf{pk}_i : i \in [n]\}, \mathbf{s})$
- $\forall\ i \in \mathcal{T}, \quad (A_i, \mathsf{Pf}_{\mathsf{Dec}i}) \leftarrow \mathsf{DecShare}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{sk}_i, C_i)$
- $\mathbf{s}' \leftarrow \mathsf{Rec}(\mathsf{pp}, \{A_i : i \in \mathcal{T}\})$, *where* $\mathbf{s}' \in \mathcal{S} \cup \{\perp\}$
- *Output* $\big(\mathsf{pp}, (\mathsf{pk}_i, \mathsf{Pf}_{\mathsf{pk}_i}, C_i)_{i\in[n]}, \mathsf{Pf}_{\mathsf{Sh}}, (\mathsf{Pf}_{\mathsf{Dec}i})_{i\in\mathcal{T}}, \mathbf{s}, \mathbf{s}'\big)$

Definition 2. *We say that the PVSS is correct with r-reconstruction if for all $\mathcal{T} \subseteq [n]$ of size at least r, any probability distribution $\mathcal{D}_{\mathcal{S}}$ over the secret space,*

$$\begin{aligned}\Pr\Big[\ & \mathsf{VerifyKey}(\mathsf{pp}, i, \mathsf{pk}_i, \mathsf{Pf}_{\mathsf{pk}_i}) = 1\ \forall\ i \in [n]\ \wedge\ \mathsf{VerifySharing}(\mathsf{pp}, (\mathsf{pk}_i, C_i)_{i\in[n]}, \mathsf{Pf}_{\mathsf{Sh}}) = 1 \\ & \wedge\ \mathsf{VerifyDec}(\mathsf{pp}, \mathsf{pk}_i, C_i, A_i, \mathsf{Pf}_{\mathsf{Dec}i}) = 1\ \forall\ i \in \mathcal{T}\ \wedge\ \mathbf{s}' = \mathbf{s} \\ & \mid\ \big(\mathsf{pp}, (\mathsf{pk}_i, C_i)_{i\in[n]}, \mathsf{Pf}_{\mathsf{Sh}}, (\mathsf{Pf}_{\mathsf{Dec}i})_{i\in\mathcal{T}}, \mathbf{s}, \mathbf{s}'\big) \leftarrow \mathtt{ExpCorr}_{\mathcal{T},\mathcal{D}_{\mathcal{S}}}(1^\lambda)\ \Big] = 1\end{aligned}$$

Verifiability. The verifiability properties assert that passing the verification procedures $\mathsf{VerifyKey}$, $\mathsf{VerifySharing}$ and $\mathsf{VerifyDec}$ guarantee respectively that the key pairs are well constructed, that the set of encrypted shares is indeed a correct sharing of a secret and that the shares have been correctly decrypted.

Definition 3 (Verifiability of Key Generation). *A PVSS satisfies* verifiability of key generation *for $\mathcal{R}_{\mathsf{Key}}$ if for all PPT $\mathcal{A}$,*

$$\begin{aligned}\Pr\Big[\ & \mathsf{VerifyKey}(\mathsf{pp}, id, \mathsf{pk}, \mathsf{Pf}_{\mathsf{pk}}) = 1 \wedge \nexists\ \mathsf{sk} \in \mathsf{SK}\ \ s.t.\ (\mathsf{pk}; \mathsf{sk}) \in \mathcal{R}_{\mathsf{Key}}\ \Big| \\ & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ (id, \mathsf{pk}, \mathsf{Pf}_{\mathsf{pk}}) \leftarrow \mathcal{A}(\mathsf{pp})\Big]\ \textit{is negligible in } \lambda.\end{aligned}$$

Definition 4 (Verifiability of Sharing Distribution). *The PVSS satisfies verifiability of sharing distribution if for every PPT $\mathcal{A}$,*

$$\begin{aligned}\Pr\Big[\ & \mathsf{VerifySharing}(\mathsf{pp}, (\mathsf{pk}_i, C_i)_{i\in[n]}, \mathsf{Pf}_{\mathsf{Sh}}) = 1\ \wedge \\ & \nexists \mathbf{s} \in \mathcal{S}\ \ s.t.\ ((C_i)_{i\in[n]}, \cdot) \leftarrow \mathsf{Dist}(\mathsf{pp}, \{\mathsf{pk}_i : i \in [n]\}, \mathbf{s})\ \Big| \\ & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), ((C_i)_{i\in[n]}, \mathsf{Pf}_{\mathsf{Sh}}) \leftarrow \mathcal{A}(\mathsf{pp})\Big]\ \textit{is negligible in } \lambda.\end{aligned}$$

Definition 5 (Verifiability of Share Decryption). *The PVSS satisfies verifiability of share decryption if the following is satisfied: For every PPT $\mathcal{A}$,*

$$\begin{aligned}\Pr\Big[\ & \mathsf{VerifyDec}(\mathsf{pp}, \mathsf{pk}, C, A, \mathsf{Pf}_{\mathsf{Dec}}) = 1\ \wedge \\ & \nexists \mathsf{sk} \in \mathsf{SK}\ \ s.t.\ (A, \cdot) \leftarrow \mathsf{DecShare}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, C)\ \Big| \\ & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{pk}, C, A, \mathsf{Pf}_{\mathsf{Dec}}) \leftarrow \mathcal{A}(\mathsf{pp})\Big]\ \textit{is negligible in } \lambda.\end{aligned}$$

Privacy (t-indistinguishability). We define now indistinguishability of secrets against an adversary corrupting t parties. We follow the notions from [30,35]. In our definition, the adversary is allowed to decide the public keys of the corrupted parties after seeing those of the honest parties. Then, provided two secrets ($\mathbf{s}_0$, $\mathbf{s}_1$) and a sharing of a random secret $\mathbf{s}_b$, the adversary has negligible advantage in guessing which secret was shared. In this paper we choose the IND2-privacy flavor where the adversary can choose $\mathbf{s}_0$, $\mathbf{s}_1$. This is stronger than IND1-privacy (used e.g. in [8,9,10]) where the challenger chooses the secrets at random.

Definition 6 (t-(IND2-privacy), based on [30]). *The PVSS is t-IND2-private if for any* $\mathsf{poly}(\lambda)$*-time adversary $\mathcal{A}_{\mathsf{Priv}}$ corrupting t parties (w.l.o.g. $\mathcal{A}_{\mathsf{Priv}}$ corrupts $[n-t+1,n]$), we have*

$$\Pr\left[\mathrm{Game}^{\mathsf{ind\text{-}secrecy},0}_{\mathcal{A}_{\mathsf{Priv}},\mathsf{PVSS}}(\lambda)=1\right]-\Pr\left[\mathrm{Game}^{\mathsf{ind\text{-}secrecy},1}_{\mathcal{A}_{\mathsf{Priv}},\mathsf{PVSS}}(\lambda)=1\right]=\mathsf{negl}(\lambda)$$

where for $b=0,1$, $\mathrm{Game}^{\mathsf{ind\text{-}secrecy},b}_{\mathcal{A}_{\mathsf{Priv}},\mathsf{PVSS}}(\lambda)$ is the following game against a challenger:

- *The challenger runs $\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda)$ and sends pp to $\mathcal{A}_{\mathsf{Priv}}$.*
- *For $i\in[n-t]$, the challenger runs $(\mathsf{sk}_i,\mathsf{pk}_i,\mathsf{Pf}_{\mathsf{pk}_i})\leftarrow\mathsf{KeyGen}(\mathsf{pp},i)$ and sends all created $(\mathsf{pk}_i,\mathsf{Pf}_{\mathsf{pk}_i})$ to $\mathcal{A}_{\mathsf{Priv}}$*
- *$\mathcal{A}_{\mathsf{Priv}}$ creates $(\mathsf{pk}_i,\mathsf{Pf}_{\mathsf{pk}_i})_{i\in[n-t+1,n]}\leftarrow\mathcal{A}_{\mathsf{Priv}}(\mathsf{pp},(\mathsf{pk}_i,\mathsf{Pf}_{\mathsf{pk}_i})_{i\in[n-t]})$ for the corrupted parties and sends them to the challenger, together with two values $\mathbf{s}_0,\mathbf{s}_1$ in $\mathcal{S}$.*
- *The challenger runs $\mathsf{VerifyKey}(\mathsf{pp},i,\mathsf{pk}_i,\mathsf{Pf}_{\mathsf{pk}_i})$ for $i\in[n-t+1,n]$. If any of these output 0 (reject), the challenger sends $\perp$ to $\mathcal{A}_{\mathsf{Priv}}$*
- *Otherwise, if all proofs accept, the challenger runs $(C_1,\ldots,C_n,\mathsf{Pf}_{\mathsf{Sh}})\leftarrow\mathsf{Dist}(\mathsf{pp},\{\mathsf{pk}_i:i\in[n]\},\mathbf{s}_b)$ (a sharing of $\mathbf{s}_b$), and sends $(C_1,\ldots,C_n,\mathsf{Pf}_{\mathsf{Sh}})$ to $\mathcal{A}_{\mathsf{Priv}}$.*
- *$\mathcal{A}_{\mathsf{Priv}}$ outputs a guess $b'\in\{0,1\}$.*

The SCRAPE test We recall the SCRAPE test from [8]. Given fixed evaluation points $\alpha_1,\ldots,\alpha_n$ in a finite field $\mathbb{F}$, the SCRAPE test allows to check whether a vector $\mathbf{y}=(y_1,...,y_n)\in\mathbb{F}^n$ is of the form $(p(\alpha_1),\ldots,p(\alpha_n))$ for some $p(X)\in\mathbb{F}[X]_{\leq d}$, by computing the inner product of $\mathbf{y}$ with a vector sampled uniformly at random from a certain set.[4] This is summed up in Theorem 1.

Theorem 1 (SCRAPE test, [8]). *Let $\mathbb{F}$ be a finite field, $\alpha_1,\ldots,\alpha_n$ pairwise distinct elements of $\mathbb{F}$, $y_1,\ldots,y_n$ arbitrary elements of $\mathbb{F}$, $0\leq d\leq n-2$ an integer. Let $v_i=\prod_{j\in[n]\setminus\{i\}}(\alpha_i-\alpha_j)^{-1}$.*

Let $m^*(X):=m_0+m_1X+\cdots+m_{n-d-2}X^{n-d-2}\leftarrow_\$\mathbb{F}[X]_{\leq n-d-2}$ and

$$T:=\sum_{i=1}^{n}v_i m^*(\alpha_i)y_i$$

1. *If there exists a polynomial $p\in\mathbb{F}[X]$ of degree $\leq d$ such that $y_i=p(\alpha_i)$ for all $i\in[n]$, then* $\Pr[T=0]=1$.
2. *Otherwise,* $\Pr[T=0]=1/|\mathbb{F}|$.

where the probability is over the uniform choice of $m^*(X)$.

For completion, we provide a proof of this theorem in Appendix A.1

2.2 Background on Class Groups

The CL Framework [15]. We first provide some background on the CL framework for encryption based on class groups. First, there is a probabilistic algorithm CLGen which is given security parameter λ, and some prime $q>2^\lambda$, and outputs $\mathsf{pp}_{\mathsf{CL}}=(q,\bar{s},\hat{G},F,f,g_q,\rho)\leftarrow\mathsf{CLGen}(1^\lambda,q;\rho)$. Here $\rho\in\{0,1\}^\lambda$ is the randomness used by CLGen and it is included in the output to signify that it can be publicly known. We will omit it from the argument of CLGen when it is not important. The (non-necessarily cyclic) group

[4] In coding-theoretic this set is the dual code to the Reed-Solomon code formed by the evaluations of polynomials of degree $\leq d$.

$\hat{G}$ has odd cardinality $q \cdot \hat{s}$ where $\gcd(q, \hat{s}) = 1$, and where $\hat{s}$ is unknown but we know an upper bound $\bar{s}$, i.e. $\hat{s} \le \bar{s}$. For technical reasons, we also assume without loss of generality $\gcd(q, \bar{s}) = 1$.

Having $F = \langle f \rangle$ denote the subgroup of cardinality q, $\hat{G}$ is a direct product $\hat{G} = \hat{G}^q \times F$, where $\hat{G}^q$ is the group of containing the q-th powers of elements in $\hat{G}$, and is of order $\hat{s}$. $\hat{G}$ is not necessarily cyclic, but there is a cyclic subgroup $G \subseteq \hat{G}$ of order $q \cdot s$ (again s is unknown) that factors as $G = G^q \times F$ where $G^q = \langle g_q \rangle$ again contains the q-th powers of elements in G. Note then that $G = \langle g \rangle$ with $g = f \cdot g_q$. Given some element in $\hat{G}$ it is not known how to determine if it is in G efficiently.

An key feature of this framework is that for subgroup $F \le G$ there is an efficient deterministic discrete logarithm algorithm $\mathsf{CLSolve}$ that given $f' \in F$ computes the unique $x \leftarrow \mathsf{CLSolve}(\mathsf{pp}_{CL}, f')$, with $x \in [0, q-1]$ such that $f^x = f'$.

We will need distributions $\mathcal{D}, \mathcal{D}_q$, over the integers such that $\{g^x : x \leftarrow \mathcal{D}\}$ and $\{g_q^x : x \leftarrow \mathcal{D}_q\}$ are statistically close to the uniform distributions in G and G^q, respectively. $\mathcal{D}, \mathcal{D}_q$ can be instantiated by either uniform or discrete Gaussian distributions [14,15,38]: in particular, choosing $\mathcal{D}$ (resp. $\mathcal{D}_q$) to be the uniform distribution in $[q\bar{s}2^{\kappa-2}]$ (resp. $[\bar{s}2^{\kappa-2}]$) leads to distributions whose statistical distances to the uniform distributions in the respective groups are at most $2^{-\kappa}$. Note also that the distribution $\{g_q^{x'} \cdot f^a : x' \leftarrow \mathcal{D}_q, a \leftarrow_\$ \mathbb{Z}_q\}$ is almost uniform in G, as a consequence of the factorization $G = G^q \times F$.

Based on this framework, Castagnos and Laguillaumie construct a linearly homomorphic encryption scheme for messages in $\mathbb{Z}_q$ in [15]. Later, variations of this encryption scheme were presented in [16]. In particular, the share distribution in our PVSS is closely related to one of the schemes presented in [16]: concretely the scheme where $\mathsf{sk} \leftarrow \mathcal{D}_q$, $\mathsf{pk} = g_q^{\mathsf{sk}}$ and the encryption of $m \in \mathbb{Z}_q$ under pk and randomness $r \leftarrow \mathcal{D}_q$ is the pair $(c_1, c_2) = (g_q^r,\ \mathsf{pk}^r f^m) \in G^q \times G$. The message can then be decrypted as $m = \mathsf{CLSolve}(c_2 \cdot c_1^{-\mathsf{sk}})$. The scheme was proved IND-CPA secure under the hard subset membership (`HSM`) assumption, described below. We describe first the assumptions we will need directly for our proofs.

First there is the `DDH-f` assumption from [16]

Definition 7 (`DDH-f` assumption, [16]). *For a PPT $\mathcal{A}$, let $\mathsf{Adv}_{\mathcal{A}}^{\mathtt{DDH\text{-}f}}(\lambda)$ be*

$$\Big| \Pr\Big[\ b^* = b \mid \mathsf{pp}_{CL} \leftarrow \mathsf{CLGen}(1^\lambda, q),\ x, y \leftarrow_\$ \mathcal{D},\ u \leftarrow_\$ \mathbb{Z}_q,\ X = g^x,\ Y = g^y,$$
$$b \leftarrow_\$ \{0,1\},\ Z_0 = g^{xy},\ Z_1 = g^{xy} f^u, b^* \leftarrow \mathcal{A}(\mathsf{pp}_{CL}, X, Y, Z_b)\Big] \ - 1/2\Big|.$$

`DDH-f` is hard for CLGen if for all PPT $\mathcal{A}$, $Adv_{\mathcal{A}}^{\mathtt{DDH\text{-}f}}(\lambda)$ is negligible in λ.

Second, the more recent rough order assumption from [6].

Definition 8 (Rough Order assumption, [6]). *For a natural number $C \in \mathbb{N}$ and security parameter λ, consider $\mathcal{D}_C^{rough}$ the uniform distribution in the set $\{\rho \in \{0,1\}^\lambda : \mathsf{pp}_{CL} \leftarrow \mathsf{CLGen}(1^\lambda, q; \rho) \wedge \forall$ prime $p < C,\ p \nmid \mathrm{ord}(\hat{G})\}$. Let*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathtt{RO}_C}(\lambda) = \Big|\Pr\Big[b = b^* \Big| \rho_0 \leftarrow_\$ \{0,1\}^\lambda, \rho_1 \leftarrow \mathcal{D}_C^{rough}, b \leftarrow_\$ \{0,1\}, b^* \leftarrow \mathcal{A}(1^\lambda, \rho_b)\Big] - 1/2\Big|$$

$\mathtt{RO}_C$ is hard for CLGen if for all PPT $\mathcal{A}$, the $\mathsf{Adv}_{\mathcal{A}}^{\mathtt{RO}_C}(\lambda)$ is negligible in λ.

We refer to [6] for the discussion of why this assumption is plausible. Moreover, we remark, as was also done in [6], that the assumption involves an inefficient challenger (as we do not know how to sample from $\mathcal{D}_C^{\mathsf{rough}}$ efficiently). However, also as [6] does, we will only use the assumption inside a security proof, namely that of Theorem 7,[5] to argue that if an adversary successfully attacks a protocol, it would be able to determine that that given class group has a low order element, contradicting the assumption.

Other hardness assumptions on class groups The standard Decisional Diffie-Hellman (DDH) assumption on G states that distinguishing tuples (g^x, g^y, g^{xy}) from tuples (g^x, g^y, g^z) where x, y, z are sampled independently from $\mathcal{D}$ is hard. More precisely:

Definition 9 (`DDH`-assumption on G). *For a PPT $\mathcal{A}$, let $\mathsf{Adv}_{\mathcal{A}}^{\mathtt{DDH}}(\lambda)$ be*

$$\Big| \Pr\Big[b^* = b \mid \mathsf{pp}_{CL} \leftarrow \mathsf{CLGen}(1^\lambda, q),\ x, y, z \leftarrow_\$ \mathcal{D},\ X = g^x,\ Y = g^y,$$
$$b \leftarrow_\$ \{0,1\},\ Z_0 = g^{xy},\ Z_1 = g^z, b^* \leftarrow \mathcal{A}(\mathsf{pp}_{CL}, X, Y, Z_b)\Big] \ - 1/2\Big|$$

[5] As well as for using the ZK proof protocol from [6] which we show in next section

We say that the DDH *problem is hard for* CLGen *if for all PPT* $\mathcal{A}$, $Adv^{\mathtt{DDH}}_{\mathcal{A}}(\lambda)$ *is negligible in* λ.

The HSM assumption states that it is hard to distinguish elements sampled from G from elements sampled from G^q (using $\mathcal{D}$ and $\mathcal{D}_q$ respectively).

Definition 10 (Hard Subgroup Membership (HSM) assumption, [16]). *For a PPT* $\mathcal{A}$, *let* $\mathsf{Adv}^{\mathtt{HSM}}_{\mathcal{A}}(\lambda)$ *be*

$$\Big|\Pr\Big[\ b = b^* |\ \mathsf{pp}_{CL} \leftarrow \mathsf{CLGen}(1^\lambda, q), x \leftarrow \mathcal{D}, y \leftarrow \mathcal{D}_q,$$
$$b \leftarrow_\$ \{0,1\}, Z_0 = g^x, Z_1 = g_q^y, b^* \leftarrow \mathcal{A}(\mathsf{pp}_{\mathsf{CL}})\Big] - 1/2\Big|$$

We say that the HSM *problem is hard for* CLGen *if for all PPT* $\mathcal{A}$, $Adv^{\mathtt{HSM}}_{\mathcal{A}}(\lambda)$ *is negligible in* λ.

2.3 Zero Knowledge Proofs for Class Groups

In this section, we recall some proofs for statements involving discrete logarithms in class groups from recent works. In this paper we will need both proofs of knowledge of discrete logarithm and proofs of discrete logarithm equality. However, in the second case we will not need the proofs to be proofs of knowledge.

Proofs of knowledge of discrete logarithm We consider two proofs of knowledge of discrete logarithm, introduced respectively in [12] and [13].[6]. Let $\mathcal{R}_{\mathsf{DL}} = \{((h, x); w) \in (G \times G) \times \mathbb{Z} : h^w = x\}$. For this and all the proofs below to be statistically honest-verifier zero knowledge, we will require the witness w to be in an interval $[-S, S]$ for some public bound S (the proofs require to set parameters which depend on S). We remark that the soundness does not guarantee that the witness is in that interval.

There is a tradeoff between both proofs in terms of complexity and security assumptions: the first proof is less efficient but does not require any assumption; the second one is more efficient but is based on the hardness assumptions (i.e. it is an argument of knowledge) $\mathtt{LO}_C$ and SR; perhaps more importantly, it requires h to be uniformly random, and in particular not decided by the adversary. We give a brief description of both proof systems and we refer the readers to [12] and [13] for more details.

The proof (Figure 1) is parametrized by natural numbers A and ℓ. We will refer to its non-interactive version via Fiat-Shamir with the name Π_{DL1}.

Theorem 2 (Adapted from [12]). *The interactive proof in Figure 1 is a Proof of Knowledge for* $\mathcal{R}_{\mathsf{DL}}$ *with knowledge soundness* $2^{-\ell}$, *and it is statistically zero-knowledge as long as* $w \in [-S, S]$, ℓ *is polynomial and* $\ell S/A$ *is negligible. By the Fiat-Shamir heuristic, the non-interactive version has the same properties in the random oracle model.*

In Figure 2 we present a proof of knowledge of discrete logarithm from [13]. We denote the non-interactive version as Π_{DL2}. As mentioned before, it does not only rely on $\mathtt{LO}_C$ and SR but also requires h to be uniformly random; typically is used in a setting where we can take h to be a random power of g_q (since the g_q outputted by CLGen itself is not uniformly random). Finally the relation for which the proof has knowledge soundness is not exactly $\mathcal{R}_{\mathsf{DL}}$ but $\mathcal{R}'_{\mathsf{DL}} = \{((h, x); (w_0, w_1)) \in (G^q \times \hat{G}) \times \mathbb{Z}^2 : h^{2^{-w_0} w_1} = x\}$, since this is what can be extracted from the proof. Still, the protocol assumes that the honest prover uses an *integer* $w = 2^{-w_0} \cdot w_1$.

Theorem 3 (Adapted from [13]). *Under the* $\mathtt{LO}_C$ *and* SR *assumption, and assuming* h *is uniformly random in a large enough subset in* G^q, *the protocol in Figure 2 is a computationally sound proof of knowledge for* $\mathcal{R}'_{\mathsf{DL}}$ *with knowledge soundness error* $4/C$, *complete if* $w \in [-S, S]$, *and statistically special honest-verifier zero knowledge as long as* $w \in [-S, S]$ *and* SC/A *is negligible. By the Fiat-Shamir heuristic, the non-interactive version has the same properties in the random oracle model.*

[6] The proofs were in fact introduced for slightly more involved relations, but for simplicity we adapt them for just proving knowledge of discrete logarithm

Proof of knowledge of discrete logarithm from [12]

Proof of knowledge for $\mathcal{R}_{\mathsf{DL}} = \{((h,x);w) \in (G \times G) \times \mathbb{Z} : h^w = x\}$

Interactive version:
Repeat ℓ times in parallel:
- The prover chooses $r \leftarrow_\$ [0,A]$ and sends $t = h^r$ to the verifier
- The verifier chooses $b \leftarrow_\$ \{0,1\}$
- The prover answers with $u = r + bw$
- The verifier accepts if $u \in [-S, A+S]$ and $h^u = t \cdot x^b$

Non-interactive (Fiat-Shamir) version:
Let $\mathcal{H} : \{0,1\}^* \to \{0,1\}^\ell$ be a random oracle.
$\underline{\mathit{\Pi}_{\mathsf{DL1}}.\text{Prove}((h,x);w)}$:
- The prover chooses $\mathbf{r} := (r_1,\ldots,r_\ell) \leftarrow_\$ [A]^\ell$, constructs $t_1 = h^{r_1},\ldots,t_\ell = h^{r_\ell}$ and computes $\mathbf{b} := (b_1,\ldots,b_\ell) = \mathcal{H}(h,x,t_1,\ldots,t_\ell)$ and $\mathbf{u} = \mathbf{r} + w\mathbf{b}$ with the sum and scalar product operating componentwise.
- Output $\mathsf{Pf} = (\mathbf{b}, \mathbf{u})$.

$\underline{\mathit{\Pi}_{\mathsf{DL1}}.\text{Verify}((h,x),\mathsf{Pf})}$:
Check $\mathbf{u} \in [A+S]^\ell$, compute $t'_j = x^{-b_j} \cdot h^{u_j}$ for $j \in [\ell]$, check that $\mathbf{b} = \mathcal{H}(h,x,t'_1,\ldots,t'_\ell)$ and accept the proof if all checks accept.

Fig. 1. Proof of knowledge of discrete logarithm from [12]

Proof of knowledge of discrete logarithm from [13]

Proof of knowledge for $\mathcal{R}'_{\mathsf{DL}} = \{((h,x);(w_0,w_1)) \in (G^q \times \hat{G}) \times \mathbb{Z}^2 : h^{2^{-w_0}w_1} = x\}$ where a honest prover uses integer $w = 2^{-w_0}w_1 \in \mathbb{Z}$.
The proof is parametrized by integers A and C and presented in Figure 2
Interactive version:
- The prover chooses $r \leftarrow_\$ [A]$ and sends $t = h^r$ to the verifier.
- The verifier chooses $c \leftarrow_\$ [C]$
- The prover answers with $u = r + cw$.
- The verifier accepts if $u \in [-SC, SC + A]$ and $h^u = t \cdot x^c$.

Non-Interactive version:
Let $\mathcal{H} : \{0,1\}^* \to [C]$ be a random oracle.
$\underline{\mathit{\Pi}_{\mathsf{DL2}}.\text{Prove}((h,x);w)}$:
- Choose $r \leftarrow_\$ [A]$, construct $t = h^r$ and compute $c = \mathcal{H}(h,x,t)$ and $u = r + cw$.
- Output $\mathsf{Pf} = (u,c)$.

$\underline{\mathit{\Pi}_{\mathsf{DL2}}.\text{Verify}((h,x),\mathsf{Pf})}$:
Checks that $u \in [-SC, SC+A]$, compute $t' = x^{-c}h^u$, check that $c = \mathcal{H}(h,x,t')$ and accept the proof if both checks accept.

Fig. 2. Proof of knowledge of discrete logarithm from [13]

Sound Proofs of discrete logarithm equality and linear relations The proofs of knowledge above are either somewhat inefficient, in the first case, or require that the basis is not controlled by the adversary, in the second. In several cases we will need proofs of discrete logarithm equality where we can settle for proofs with soundness, instead of proofs of knowledge. We can instantiate these from the proofs for linear relations in a class group introduced in [6]. In this case, soundness requires the rough-order assumption also introduced in [6].

Consider the following relation $\mathcal{R}_{\mathsf{LinCL}}$ given by[7]

$$\{((X_{i,j})_{i\in[n],j\in[m]}, (Y_i)_{i\in[n]}; (w_j)_{j\in[m]}) \in (G)^{nm+n} \times \mathbb{Z}^m : Y_i = \prod_{j=1}^{m} X_{i,j}^{w_j} \ \forall i \in [n]\}$$

For $m = 1$, this is in fact the discrete logarithm equality relation

$$\mathcal{R}_{\mathsf{DLEQ}} = \{((X_i)_{i\in[n]}, (Y_i)_{i\in[n]}; w) \in (G)^{2n} \times \mathbb{Z} : Y_i = X_i^w \ \forall i \in [n]\}$$

A Σ-protocol for $\mathcal{R}_{\mathsf{LinCL}}$, parametrized by $A, C \in \mathbb{N}$, is given in Figure 3.
We denote $\mathbf{X} = (X_{i,j})_{i\in[n],j\in[m]}, \mathbf{Y} = (Y_i)_{i\in[n]}, \mathbf{w} = (w_j)_{j\in[m]}$.

Proof of linear class group relations from [6]

Proof for $\mathcal{R}_{\mathsf{LinCL}} = \left\{(\mathbf{X}, \mathbf{Y}; \mathbf{w}) \in G^{nm+n} \times \mathbb{Z}^m : Y_i = \prod_{j=1}^m X_{i,j}^{w_j} \ \forall i \in [n]\right\}$

Interactive version:

1. The prover chooses $(r_1, \ldots, r_m) \leftarrow_\$ [A]^m$, constructs $T_i = \prod_{j=1}^m X_{i,j}^{r_j}$ for $i \in [n]$ and sends $(T_1, \ldots, T_n)$.
2. The verifier chooses $c \leftarrow_\$ [C]$ and sends it to the prover.
3. The prover computes $u_j = r_j + cw_j$ for $j \in [m]$ and sends them to the verifier.
4. The verifier checks $u_j \in [-SC, SC + A]$ for all $j \in [m]$, and also $T_i \cdot Y_i^c = \prod_{j=1}^m X_{i,j}^{u_j}$ and accepts if all checks pass.

Non-interactive version:
Let $\mathcal{H} : \{0,1\}^* \to [C]$
$\Pi_{\mathsf{LinCL}}.\mathsf{Prove}(\mathbf{X}, \mathbf{Y}; \mathbf{w})$:

- Choose $\mathbf{r} = (r_1, \ldots, r_m) \leftarrow_\$ [0, A]^m$, construct $T_i = \prod_{j=1}^m X_{i,j}^{r_j}$ for $i \in [n]$, compute $c = \mathcal{H}(\mathbf{X}, \mathbf{Y}, \mathbf{T})$, and $\mathbf{u} = \mathbf{r} + c\mathbf{w}$ (coordinatewise)
- Output $\mathsf{Pf} = (\mathbf{u}, c)$

$\Pi_{\mathsf{LinCL}}.\mathsf{Verify}((\mathbf{X}, \mathbf{Y}), \mathsf{Pf})$:
Check $\mathbf{u} \in [-SC, SC + A]^m$, compute $T_i = Y_i^{-c} \cdot \prod_{j=1}^m X_{i,j}^{u_j}$ for $i \in [n]$, check $c = \mathcal{H}(\mathbf{X}, \mathbf{Y}, \mathbf{T})$, accept if both checks accept.

Fig. 3. Proof of linear class group relations from [6]

Lemma 1 ([6]). *The interactive proof in Figure 3 is complete, computationally sound with soundness error $1/C + \mathsf{negl}$ under the RO_C assumption and statistically special honest-verifier zero knowledge if SC/A is negligible. By the Fiat-Shamir heuristic, the non-interactive version has the same properties in the random oracle model.*

Remark 1. By the result of [21], the non-interactive version of the proof in Figure 3 obtained via the Fiat-Shamir transform in the random oracle model is simulation sound.

3 PVSS over Class Groups

3.1 The PVSS scheme

Our PVSS is similar to the DHPVSS scheme in YOLO YOSO [10], which we recall in Appendix B for comparison. In particular we replace the El Gamal encryption used there by the Castagnos-Laguillaumie-Tucker encryption from [16]. The benefit we obtain over DHPVSS is that in our scheme parties can reconstruct the share "field" secret $s \in \mathbb{Z}_q$, where in DHPVSS they can only reconstruct g^s (with g being a

[7] Notation: To avoid confusion with the group G^q of q-th powers of elements from G, we denote the direct product of m copies of G, for $m \in \mathbb{N}$, as $(G)^m$

generator of the DDH-hard group $\mathbb{G}$). This is fine for applications of PVSS such as distributed randomness beacons [8] and can also be turned into a PVSS for $\mathbb{Z}_q$ by defining the secret to be $s' = H(g^s) + a$, for some efficiently computable $H : \mathbb{G} \to \mathbb{Z}_q$ and an element a published by the dealer. But then the PVSS is no longer linear, which makes it harder to be used for MPC-related applications and DKG.

In exchange, there arise some technical challenges with respect to [10]. First, several steps of the construction need ZK proofs, and ZK proofs of knowledge are somewhat inefficient or only applicable under certain conditions (see remarks above and Section 2.3) . Fortunately, we show that we only really need proofs of knowledge in the key generation algorithm. This means that using less efficient PoKs (Π_{DL1} in Section 2.3) may not be so problematic as key generation can be carried out long before the PVSS takes place; but also one can use the more efficient proof Π_{DL2} from [13] (Section 2.3) by randomizing the generator g_q. The second issue will be the construction of an efficient (constant in the number of parties n) proof of correct sharing, but we defer this discussion to Section 3.2.

We present our scheme for a general case where the space of secrets is $\mathbb{Z}_q^k$. For applications in this paper we only need $k = 1$, but the general case is not much harder to present an in addition PVSS with larger secrets have been considered for some applications e.g. in [9].

PVSS Scheme qCLPVSS. Let λ be the security parameter and $q > 2^\lambda$ prime. Let k (size of the secret), t (privacy threshold) and n (number of parties) be natural numbers, with $k, t, n = \mathsf{poly}(\lambda)$ (and hence we can assume $n + k \leq q$) and $k + t \leq n$. Our scheme qCLPVSS consists of the tuple of algorithms ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{VerifyKey}, \mathsf{Dist}, \mathsf{VerifySharing}, \mathsf{DecShare}, \mathsf{Rec}, \mathsf{VerifyDec}$) below:

- $\mathsf{qCLPVSS.Setup}(1^\lambda, q, k, t, n)$:
  1. Specify a set of pairwise distinct points $\{\beta_1, \ldots, \beta_k, \alpha_1, \ldots, \alpha_n\} \subset \mathbb{Z}_q$.
     Let $\mathsf{pp}_{\mathsf{Sh}} = (q, k, t, n, (\beta_j)_{j \in [k]}, (\alpha_i)_{i \in [n]})$
  2. Run $\mathsf{pp}_{\mathsf{CL}} := (q, \bar{s}, f, g_q, \hat{G}, F, \rho) \leftarrow \mathsf{CLGen}(1^\lambda, q)$.
  3. The output is then $\mathsf{pp} = (\mathsf{pp}_{\mathsf{Sh}}, \mathsf{pp}_{\mathsf{CL}})$.

- $\mathsf{qCLPVSS.KeyGen}(\mathsf{pp}, i)$:
  1. Sample $\mathsf{sk}_i \leftarrow \mathcal{D}_q$ and compute $\mathsf{pk}_i = g_q^{\mathsf{sk}_i}$.
  2. Create proof $\mathsf{Pf}_{\mathsf{pk}_i} = \mathtt{NIZKPoK}_{\mathsf{DL}}.\mathsf{Prove}(\{(g_q, \mathsf{pk}_i); \mathsf{sk}_i : \mathsf{pk}_i = g_q^{\mathsf{sk}_i}\})$
  3. Output$(\mathsf{sk}_i, \mathsf{pk}_i, \mathsf{Pf}_{\mathsf{pk}_i})$.

- $\mathsf{qCLPVSS.VerifyKey}(\mathsf{pp}, i, \mathsf{pk}_i, \mathsf{Pf}_{\mathsf{pk}_i})$:
  Run $\mathtt{NIZKPoK}_{\mathsf{DL}}.\mathsf{Verify}$ on $\mathsf{Pf}_{\mathsf{pk}_i}$ with respect to statement (g_q, pk_i) and output its result.

- $\mathsf{qCLPVSS.Dist}(\mathsf{pp}, (\mathsf{pk}_i)_{i \in [n]}, \mathbf{s})$, where $\mathbf{s} = (s_1, \ldots, s_k) \in \mathbb{Z}_q^k$:
  1. Create a Shamir sharing of $\mathbf{s}$: sample a polynomial $p(X) \in \mathbb{Z}_q[X]_{\leq t+k-1}$ with $p(\beta_j) = s_j$i for $j \in [k]$ and set $\sigma_i = p(\alpha_i)$ for $i \in [n]$.
  2. Sample $r \leftarrow \mathcal{D}_q$ and compute $R = g_q^r$.
  3. Create $B_i = \mathsf{pk}_i^r \cdot f^{\sigma_i}$.
  4. Create the sharing proof (not necessarily of knowledge)
  $$\mathsf{Pf}_{\mathsf{Sh}} = \mathtt{NIZK}_{\mathsf{Sh}}.\mathsf{Prove}(\{(f, g_q, (\mathsf{pk}_i)_{i=1}^n, R, (B_i)_{i=1}^n); (p(X), r) :$$
  $$\deg p(X) \leq t + k - 1,\ R = g_q^r,\ B_i = \mathsf{pk}_i^r \cdot f^{p(\alpha_i)}\ \forall i \in [n]\})$$
     We show how to instantiate $\mathtt{NIZK}_{\mathsf{Sh}}$ in Section 3.2.
  5. Output $(R, B_1, \ldots, B_n, \mathsf{Pf}_{\mathsf{Sh}})$. To make it syntactically consistent with our definition in Section 2.1, we define $C_i := (R, B_i)$ for all $i \in [n]$, and notice that $(R, B_1, \ldots, B_n, \mathsf{Pf}_{\mathsf{Sh}})$ contains the same information as $(C_1, \ldots, C_n, \mathsf{Pf}_{\mathsf{Sh}})$.

- $\mathsf{qCLPVSS.VerifySharing}(\mathsf{pp}, (\mathsf{pk}_i)_{i \in [n]}, (C_1, \ldots, C_n, \mathsf{Pf}_{\mathsf{Sh}}))$, where $C_i = (R, B_i)$:
  Run $\mathtt{NIZK}_{\mathsf{Sh}}.\mathsf{Verify}$ on $\mathsf{Pf}_{\mathsf{Sh}}$ with respect to statement $(f, g_q, (\mathsf{pk}_i)_{i=1}^n, R, (B_i)_{i=1}^n)$ and output its result.

- $\mathsf{qCLPVSS.DecShare}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{sk}_i, C_i)$, where $C_i = (R, B_i)$:
  1. Compute $f_i = B_i \cdot R^{-\mathsf{sk}_i}$, $A_i = \mathsf{CLSolve}(f_i)$ and $M_i = f_i^{-1} \cdot B_i$.

2. Compute $\mathsf{Pf}_{\mathsf{Dec}i} = \mathtt{NIZK}_{\mathsf{DLEQ}}.\mathsf{Prove}(\{(g_q, R, \mathsf{pk}_i, M_i); \mathsf{sk}_i : g_q^{\mathsf{sk}_i} = \mathsf{pk}_i, R^{\mathsf{sk}_i} = M_i\})$. Again this does not need to be a proof of knowledge.
3. Output $(A_i, \mathsf{Pf}_{\mathsf{Dec}i})$.

- $\mathsf{qCLPVSS.Rec}(\mathsf{pp}, \{A_i : i \in \mathcal{T}\})$:

1. If $|\mathcal{T}| < t + k$, output $\perp$.
2. Otherwise select $\mathcal{T}' \subseteq \mathcal{T}$, with $|\mathcal{T}'| = t + k$ (e.g. the first $t + k$ indices in $\mathcal{T}$).
3. For each $j \in [k]$, define $s'_j = \sum_{i \in \mathcal{T}'} A_i \cdot L_i(\beta_j)$ where $L_i(X) = \text{Lag}_{i,\mathcal{T}',\{\alpha_i : i \in \mathcal{T}'\}}$.[8]
4. Output $\mathbf{s}' = (s'_1, \ldots, s'_k)$.

- $\mathsf{qCLPVSS.VerifyDec}(\mathsf{pp}, C_i, A_i, \mathsf{Pf}_{\mathsf{Dec}i})$ where $C_i = (R, B_i)$:
  Compute $M_i = f^{-A_i} \cdot B_i$ and run $\mathtt{NIZK}_{\mathsf{DLEQ}}.\mathsf{Verify}$ on $\mathsf{Pf}_{\mathsf{Dec}i}$ with respect to statement $(g_q, R, \mathsf{pk}_i, M_i)$, and output the result of the verification.

We now show that the PVSS above guarantees the security properties from Section 2.1. In particular there is $t + k$-reconstruction and t-IND2-privacy.

Theorem 4. $\mathsf{qCLPVSS}$ *is a correct PVSS with* $t + k$*-reconstruction*

Proof. The proof is quite immediate, we give a detailed proof in Appendix A.2

Theorem 5. – *If* $\mathtt{NIZKPoK}_{DL}$ *is a proof of knowledge with knowledge error negligible in* λ *then* $\mathsf{qCLPVSS}$ *has verifiability of key generation.*
– *If* $\mathtt{NIZK}_{\mathsf{Sh}}$ *is a proof with soundness error negligible in* λ *then* $\mathsf{qCLPVSS}$ *has verifiability of sharing distribution.*
– *If* $\mathtt{NIZK}_{\mathsf{DLEQ}}$ *with soundness error negligible in* λ *then* $\mathsf{qCLPVSS}$ *has verifiability of share decryption.*

Proof. Trivial as the statements proved by the NIZK proofs exactly guarantee correct key generation, sharing distribution and share decryption, respectively.

In order to prove t-IND2-secrecy, we need to introduce a modified hardness assumption, and show it is implied by `DDH-f`. The new assumption, `DDH-qf` is very similar to `DDH-f` but the generator of G is replaced by the generator of G^q.

Definition 11 (`DDH-qf` hardness assumption). *For a PPT* $\mathcal{A}$ *let*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathit{DDH\text{-}qf}}(\lambda) := \big|Pr[b^* = b | \mathsf{pp}_{CL} \leftarrow \mathsf{CLGen}(1^\lambda, q), x, y \leftarrow \mathcal{D}_q, u \leftarrow_{\$} \mathbb{Z}_q, X = g_q^x,$$

$$Y = g_q^y, b \leftarrow_{\$} \{0,1\}, Z_0 = g_q^{xy}, Z_1 = g_q^{xy} f^u, b^* \leftarrow \mathcal{A}(\mathsf{pp}_{CL}, X, Y, Z_b)] - 1/2\big|$$

`DDH-qf` is hard for CLGen *if* $\forall$ *PPT* $\mathcal{A}$*,* $Adv_{\mathcal{A}}^{\mathit{DDH\text{-}qf}}(\lambda)$ *is negligible in* λ*.*

Lemma 2. *If `DDH-f` is hard for* CLGen*, then `DDH-qf` is hard for* CLGen*.*

Proof. Appendix A.4

Theorem 6. $\mathsf{qCLPVSS}$ *is t-IND2-secret under `DDH-f`, assuming* $\mathtt{NIZK}_{\mathsf{Sh}}$*,* $\mathtt{NIZK}_{\mathsf{DLEQ}}$ *are zero-knowledge proofs and* $\mathtt{NIZKPoK}_{\mathsf{DL}}$ *is a zero-knowledge proof of knowledge.*

Proof. The proof is presented in Apppendix A.3.

---

[8] Recall, that by definition of Lag, $L_i(X) = \prod_{j \in \mathcal{T}' \setminus \{i\}} \frac{X - \alpha_j}{\alpha_i - \alpha_j}$

3.2 Instantiating the proofs

Sharing proof We discuss how to instantiate the sharing proof $\mathsf{Pf}_{\mathsf{Sh}}$, which we consider the main technical challenge of the PVSS construction. Recall this is a zero knowledge proof for the language

$$\{(f, g_q, (\mathsf{pk}_i)_{i=1}^n, R, (B_i)_{i=1}^n); (p(X), r) : \deg p(X) \le t, R = g_q^r, B_i = \mathsf{pk}_i^r f^{p(\alpha_i)} \ \forall i \in [n]\}.$$

As we have mentioned before, we use the overall idea from YOLO YOSO [10], which in turn consists in using the SCRAPE check from Theorem 1 in an efficient way which yields a constant size (in n) proof, but we will need to do adjustments to this strategy.

The idea from [10], translated to our class group framework, is as follows: if we sample a random polynomial $m^* \in \mathbb{Z}_q[X]_{\le n-t-k+1}$ then for any correct sharing ($\sigma_i = p(\alpha_i)$ with $\deg p(X) \le t$) we must have $\sum_{i=1}^n \sigma_i \cdot v_i \cdot m^*(\alpha_i) = 0$ in $\mathbb{Z}_q$ for the v_i's defined in Theorem 1.

We embed $w_i = v_i \cdot m^*(\alpha_i) \in \mathbb{Z}_q$ as integers in $[q-1]$, and compute the products $U = \prod_{i=1}^n \mathsf{pk}_i^{w_i}$ and $V = \prod_{i=1}^n B_i^{w_i}$. If the B_i's are correct then $V = \prod_{i=1}^n {\mathsf{pk}_i}^{rw_i} f^{\sigma_i w_i}$ but the second term cancels out because $\sum_{i=1}^n \sigma_i w_i = 0 \mod q$ (recall f is of order q). So then $V = U^r$ which can be proved using a proof of discrete logarithm equality with $R = g_q^r$. If the σ_i's are not valid, with large probability $\sum_{i=1}^n \sigma_i w_i \ne 0 \mod q$ (by Theorem 1), the F-part of the product does not cancel out, and the proof will not pass.

However, there is a problem that did not appear in the setting of [10]: it may be that a malicious prover sets $B_i = (H_i \mathsf{pk}_i^r) \cdot f^{\sigma_i}$, with correct shares σ_i but where $H_i \ne 1$ are elements in $\hat{G}^q$ such that, when computing V the product $\prod H_i^{w_i}$ cancels out and this is not caught by the proof.

We solve this problem as follows: we randomize further the values w_i by replacing them with $w_i' = w_i + c_i q$ for some random $c_i \in [C]$. This does not affect the F-part of the equation, as we are adding a multiple of q, but as we will see the prover can only pass this test with high probability by either setting all $H_i = 1$ (and then the shares are correct) or by breaking the rough order assumption from [6]. In addition, this modification does not affect the communication complexity, while the computation only increases slightly by computing n products and sums of integers. The proof Π_{Sh} of correct sharing is presented in Figure 4.

To prove the soundness of Π_{Sh} we first need the following lemma.

Lemma 3. *Let $H_i \in \hat{G}^q$ be elements in $\hat{G}^q$ such that there is at least one element $H_j \ne 1$. Let $w_i \in \mathbb{Z}$. Sample $(c_1, \dots, c_n) \leftarrow_\$ [C]^n$ for some integer $C > 1$. Then if H_j has order $\ge C$, the probability that $\prod_{i=1}^n H_i^{w_i + c_i q} = 1$ is at most $1/C$.*

Proof. Without loss of generality, we assume $j = 1$, i.e. the order of H_1 is at least C. Then fix any $(c_2, \dots, c_n) \in [C]^{n-1}$ and consider the quantities $M_c = H_1^{w_1+cq} \prod_{i=2}^n H_i^{w_i+c_i q}$. Clearly if $M_c = M_{c'}$ for $c \ne c'$ then $1 = M_c \cdot M_{c'}^{-1} = H_1^{(c-c')q}$. But since the order of $\hat{G}^q$ is coprime to q, then $H_1^{(c-c')} = 1$, a contradiction with the order of H_1 (since $|c - c'| \le C - 1$). Therefore at most one M_c can equal 1. Since this is for any $(c_2, \dots, c_{n-1}) \in [C]^{n-1}$ we obtain the lemma.

Theorem 7. *In the random oracle model, and assuming $\mathtt{RO}_C$ is hard for* CLGen, *Π_{Sh} in Figure 4 is a proof for the relation $\mathcal{R}_{\mathsf{Sh}}$ with soundness error $\epsilon_{\mathsf{DLEQ}} + 1/C + 1/q + \mathsf{negl}(\lambda)$, where ϵ_{DLEQ} is the soundness error of $\mathtt{NIZK}_{\mathsf{DLEQ}}$. It is zero knowledge assuming $\mathtt{NIZK}_{\mathsf{DLEQ}}$ is.*

Proof. The proof is shown in Apprendix A.5.

Remark 2. By [21], Π_{Sh} is simulation sound in the random oracle model.

Discrete logarithm knowledge and discrete logarithm equality We have seen that $\mathtt{NIZK}_{\mathsf{Sh}}$, and hence the sharing distribution algorithm $\mathsf{qCLPVSS.Share}$, can be instantiated by Π_{Sh} as long as we have a proof $\mathtt{NIZK}_{\mathsf{DLEQ}}$ of discrete logarithm equality. Moreover, we also need $\mathtt{NIZK}_{\mathsf{DLEQ}}$ for the sharing decryption $\mathsf{DecShare}$. In both cases, we do not need a proof of knowledge of the exponent, so we can use Π_{LinCL} in Figure 3. This proof requires the $\mathtt{RO}_C$ assumption, but we already need this assumption for Π_{Sh} anyway.

Finally, we do need a proof of knowledge $\mathtt{NIZKPoK}_{\mathsf{DL}}$ of discrete logarithm in the key generation algorithm $\mathsf{qCLPVSS.KeyGen}$. We have listed two options in Section 2.3: either we use Π_{DL1}, which has a higher complexity but which does not require hardness assumptions and can be applied regardless of how

Proof Π_{Sh} of correct sharing:

Proof for the relation

$$\mathcal{R}_{\mathsf{Sh}} = \{(f, g_q, (\mathsf{pk}_i)_{i=1}^n, R, (B_i)_{i=1}^n); (p(X), r) : p(X) \in \mathbb{Z}_q[X]_{\le t+k-1},$$

$$r \in \mathbb{Z}, R = g_q^r, B_i = \mathsf{pk}_i^r f^{p(\alpha_i)} \ \forall i \in [n]\}.$$

The proof is parametrized by $C \in \mathbb{Z}$. We assume a random oracle $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q[X]_{\le n-t-k-1} \times [C]^n$ and a $\mathtt{NIZK}$ proof $\mathtt{NIZK}_{\mathsf{DLEQ}}$ for discrete logarithm equality in class groups, given by algorithms $\mathtt{NIZK}_{\mathsf{DLEQ}}.\mathsf{Prove}$, $\mathtt{NIZK}_{\mathsf{DLEQ}}.\mathsf{Verify}$.

$\Pi_{\mathsf{Sh}}.\mathsf{Prove}((f, g_q, (\mathsf{pk}_i)_{i=1}^n, R, (B_i)_{i=1}^n); (p(X), r))$:

1. Compute $(m^*(X), c_1, \ldots, c_n) = \mathcal{H}(\mathsf{pk}_1, \ldots, \mathsf{pk}_n, R, B_1, \ldots, B_n)$. Note $m^*(X) \in \mathbb{Z}_q[X]_{\le n-t-k-1}$ and $c_i \in [C]$ for each $i \in [n]$.
   Let $v_i = \prod_{j \in [n] \setminus \{i\}} (\alpha_i - \alpha_j)^{-1} \in \mathbb{Z}_q$.
2. Define $w_i = m^*(\alpha_i) \cdot v_i$ where the evaluation and product is in $\mathbb{Z}_q$. From now on see w_i as integers (in $[q-1]$).
3. Compute $w_i' = w_i + c_i q$ over the integers.
4. Compute $U = \prod_{i=1}^n \mathsf{pk}_i^{w_i'}$ and $V = \prod_{i=1}^n B_i^{w_i'}$.
5. Compute $\mathtt{NIZK}_{\mathsf{DLEQ}}((g_q, U, R, V); r) : g_q^r = R \wedge U^r = V)$. We write $\mathsf{Pf}_{\mathsf{Sh}} = \mathtt{NIZK}_{\mathsf{DLEQ}}.\mathsf{Prove}((g_q, U, R, V); r)$.
6. Output $\mathsf{Pf}_{\mathsf{Sh}}$.

$\Pi_{\mathsf{Sh}}.\mathsf{Verify}((f, g_q, (\mathsf{pk}_i)_{i=1}^n, R, (B_i)_{i=1}^n), \mathsf{Pf}_{\mathsf{Sh}})$:

1. Compute $(m^*(X), c_1, \ldots, c_n) = \mathcal{H}(\mathsf{pk}_1, \ldots, \mathsf{pk}_n, R, B_1, \ldots, B_n)$.
2. Compute w_i' from $m^*(X)$ and the public information as the dealer does.
3. Compute $U = \prod_{i=1}^n \mathsf{pk}_i^{w_i'}$ and $V = \prod_{i=1}^n B_i^{w_i'}$.
4. Output $\mathtt{NIZK}_{\mathsf{DLEQ}}.\mathsf{Verify}((g_q, U, R, V), \mathsf{Pf}_{\mathsf{Sh}})$.

Fig. 4. Proof for correct PVSS sharing

g_q is chosen; or we use Π_{DL2} which relies on the $\mathtt{LO}_C$ and $\mathtt{SR}$ assumptions and where we need to slightly modify the setup to replace g_q by a randomized $g_q' = g_q^\rho$ for a random ρ which the adversary cannot control. We remark that, although Π_{DL2} only guarantees witness extraction for the slightly different relation $\mathcal{R}'_{DL}$ where only knowledge of integers ρ_0 and ρ_1 with $g_q^{2^{-\rho_0}\rho_1} = \mathsf{pk}_i$ is guaranteed, this is not really a big problem for us: the one place where we need extraction of the exponent is in the proof of Theorem 6, and there we can replace extracted sk_i by ${\rho_{i,1}}^{2^{-\rho_{i,0}}}$ and use the fact that square roots in G^q are computed efficiently.

3.3 Complexity

We focus on the communication complexity of $\mathsf{qCLPVSS}$, since this is usually the main bottleneck in PVSS applications. Let κ be a statistical security parameter for soundness, zero knowledge (so both soundness error and statistical distance in the zero knowledge simulation are bounded by $2^{-\kappa}$), and also so that we instantiate $\mathcal{D}_q$ by sampling uniformly in $[2^\kappa \bar{s}]$ (see Section 2.2). [9]

- $\mathsf{qCLPVSS.KeyGen}$: 1 element in G and $\sim \kappa^2 + \kappa \log \kappa$ bits (using Π_{DL1}) or $\sim 3\kappa + \log(\bar{s})$ (using Π_{DL2}) bits per party.
- $\mathsf{qCLPVSS.Dist}$: $n+1$ elements in G and $\sim 3\kappa + \log(\bar{s})$ bits
- $\mathsf{qCLPVSS.DecShare}$: $\sim 3\kappa + \log(\bar{s}) + \log q$ bits (per party)

Moreover, the encrypted shares are n CL-$\mathtt{HSM}$ ciphertexts (where we only send R once) and may benefit from compression techniques [5].

Although we do not estimate the computational complexity in details, the main point of our construction is that it maintains the linear complexity in terms of group operations that was achieved in

[9] In practice we consider $\kappa = 40$ is reasonable.

previous works [8,9,10]. While group operations on class groups have higher complexity than over groups defined on elliptic curves, the concrete times estimated in [5] show that the overhead in computation time is of about an order of magnitude.

Comparison with YOLO YOSO [10]. The PVSS scheme in YOLO YOSO requires essentially the same amount of group operations computation and group elements communication. Since our scheme operates over class groups, it clearly has an overhead in relation to elliptic curve implementations of YOLO YOSO as estimated in [5]. However, we achieve more flexibility in being able to retrieve the original shared secret s (or the result s' of linear operations on multiple secrets), whereas YOLO YOSO only allows for obtaining g^s (or the result s' of linear operations on multiple secrets). Moreover, we prove the IND-2-security notion from [30], whereas YOLO YOSO only shows the weaker IND-1-security also from [30], although we think it can also be proved IND-2-secure. In contrast, previous (and less efficient) PVSS using similar techniques [8,9]are only IND-1. This is because these are based on a OW-CPA secure encryption scheme that allows for the necessary linear operations used in the NIZKs of sharing correctness.

Comparison with [31]. An independent work [31] constructs a PVSS scheme from class groups, motivated by distributed key generation. The shares are encrypted in the same way as ours (namely the dealer sends $(R, (B_i)_{i=1}^n)$. However, our scheme presents several advantages: the remaining communication of the sharing phase (the size of the proof $\mathsf{Pf}_{\mathsf{Sh}}$) is independent of n and t, while they require to send commitments to the t coefficients of the polynomial, as well as somewhat larger proofs. Moreover, our PVSS achieves the strong IND-2-security property, while their construction does not satisfy the notion of indistinguishability of secrets, but a weaker notion of privacy that allows leakage. This leakage is fine for their DKG application, but it may not be adequate in other applications.

4 Application: Distributed Key Generation

We extend $\mathsf{qCLPVSS}$ to construct a distributed key generation protocol for a given cyclic group $\mathbb{H}$ of prime order q where DDH is assumed to be hard (e.g. an elliptic curve group). We assume an static adversary that can corrupt at most $t \leq \frac{n-1}{2}$ parties. Our goal is for parties to generate partial public keys $\mathsf{tpk}_i = h^{p(\alpha_i)}$ and a global public key $\mathsf{tpk} = h^{p(\beta)}$, where each party i privately knows $\mathsf{tsk}_i = p(\alpha_i)$. The global secret key is implicitly defined as $\mathsf{tsk} = p(\beta)$.

We will present two constructions of discrete key generation: the first one has two rounds of communication but has the property that the public key can not be biased by the adversary. The second is a non-interactive protocol (only one round of communication) but a rushing adversary can bias the public key. Note this is unavoidable for one-round distributed key generation (see [32]).

4.1 Two-round DKG with unbiasable public key

In this section we will implement the functionality $\mathcal{F}_{\mathsf{DKG}}$ in Figure 5. Note that when interacting with this functionality, the adversary can decide on the threshold partial secret keys tsk_i of the corrupted parties. But the global secret key tsk is chosen by the functionality uniformly at random and independently of these tsk_i, and hence the adversary has no control on the threshold public key tpk.

The strategy follows the general template by Katz [32], using our PVSS. Every party PVSSs a contribution s_j to the secret key. This determines a set $\mathcal{Q}$ of parties whose sharing proofs pass the check. Parties define their tsk_i summing the shares received from parties in $\mathcal{Q}$. In the second round, parties publish $\mathsf{tpk}_i = h^{\mathsf{tsk}_i}$ and prove this is consistent with the encrypted shares received before.

Theorem 8. *Under the* `DDH-f` *(for privacy of* $\mathsf{qCLPVSS}$*) and* $\mathtt{RO}_C$ *(for verifiability of* $\mathsf{qCLPVSS}$ *and simulation soundness of* Π_{Sh} *and* Π_{LinCL}*) assumptions the protocol* Π_{DKG} *in Figure 6 realizes* $\mathcal{F}_{\mathsf{DKG}}$ *securely in the random model in the presence of a malicious static adversary corrupting* $t \leq \frac{n-1}{2}$ *parties.*

Proof. Appendix A.6

Functionality $\mathcal{F}_{\mathsf{DKG}}$

$\mathcal{F}_{\mathsf{DKG}}$ is parameterized by a DDH-hard cyclic group $\mathbb{H}$ of prime order q, with generator h. Let n and $1 \leq t \leq (n-1)/2$ be integers. Let $\beta, \alpha_1, \ldots, \alpha_n$ be pairwise distinct elements in $\mathbb{Z}_q$. $\mathcal{F}_{\mathsf{DKG}}$ interacts with parties $ID_1, \ldots, ID_n$ and an adversary $\mathcal{S}$ that corrupts at most t parties. $\mathcal{F}_{\mathsf{DKG}}$ works as follows:

- Upon receiving $(\textsc{Gen}, sid, ID_i)$ from a party ID_i:
  1. If ID_i is honest, forward $(\textsc{Gen}, sid, ID_i)$ to $\mathcal{S}$.
  2. If ID_i is corrupted, wait for $\mathcal{S}$ to send $(\textsc{SetShare}, sid, ID_i, \mathsf{tsk}_i)$ where $\mathsf{tsk}_i \in \mathbb{Z}_q$ and set $\mathsf{tpk}_i = g^{\mathsf{tsk}_i}$.
- Let J be the set of all parties ID_j who sent $(\textsc{Gen}, sid, ID_j)$. If all honest parties are in J, proceed as follows:
  1. Sample a random polynomial p of degree at most t with $p(\alpha_i) = \mathsf{tsk}_i$ for all tsk_i sent by $\mathcal{S}$ in the previous step[a] For every party ID_ℓ for which no tsk_i has been received, set $\mathsf{tsk}_\ell = p(\alpha_\ell)$ and $\mathsf{tpk}_\ell = h^{\mathsf{tsk}_\ell}$.
  2. Set $\mathsf{tpk} = h^{p(\beta)}$. [b]
  3. For all corrupted $ID_c \in J$, send $(\textsc{Keys}, sid, \mathsf{tsk}_c, \{\mathsf{tpk}_j\}_{j \in J}, \mathsf{tpk})$ to $\mathcal{S}$.
  4. Wait for $\mathcal{S}$ to send $(\textsc{Abort}, sid, C)$ where C is a set of corrupted parties.
  5. Send $(\textsc{Keys}, sid, \mathsf{tsk}_j, \{\mathsf{tpk}_k\}_{k \in J \setminus C}, \mathsf{tpk})$ to each honest party ID_j.

---

[a] At least one such polynomial exists because there are at most t corrupted parties.
[b] Note that $p(\beta)$ is uniformly random in $\mathbb{Z}_q$ independently of the tsk_i sent in the previous step, and hence tpk is uniform in $\mathbb{H}$ conditioned to those tsk_i.

Fig. 5. Distributed Key Generation Functionality $\mathcal{F}_{\mathsf{DKG}}$

Two-round DKG protocol Π_{DKG} with Unbiasable Public Key

Let q be a prime and $0 \leq t < n \leq q$ be positive integers. Let $\mathbb{H}$ be a cyclic group of order q generated by h.

Setup:

1. Parties run $\mathsf{pp} \leftarrow \mathsf{qCLPVSS.Setup}(1^\lambda, q, 1, t, n)$
2. Each party i runs $(\mathsf{sk}_i, \mathsf{pk}_i, \mathsf{Pf}_{\mathsf{pk}_i}) \leftarrow \mathsf{qCLPVSS.KeyGen}(\mathsf{pp}, i)$

Only parties who have produced $(\mathsf{sk}_i, \mathsf{pk}_i, \mathsf{Pf}_{\mathsf{pk}_i})$ that pass the verification $\mathsf{qCLPVSS.VerifyKey}$ are accepted to participate in the protocol.

Protocol:

1. Each party $j \in [n]$:
   (a) Samples uniformly random $s_j \in \mathcal{F}_q$
   (b) Runs $(R_j, (B_{j,i})_{i\in[n]}, \mathsf{Pf}_{\mathsf{Sh}j}) \leftarrow \mathsf{qCLPVSS.Share}(\mathsf{pp}, (\mathsf{pk}_i)_{i\in[n]}, s_j)$.
   (c) Publishes $(R_j, (B_{j,i})_{i\in[n]}, \mathsf{Pf}_{\mathsf{Sh}j})$
2. Let $\mathcal{Q}$ be the set of j for which

$$\mathsf{qCLPVSS.VerifySharing}(\mathsf{pp}, (\mathsf{pk}_i)_{i\in[n]}, R_j, (B_{j,i})_{i\in[n]}, \mathsf{Pf}_{\mathsf{Sh}j})) = 1.$$

   Parties compute $R_{\mathcal{Q}} = \prod_{j\in\mathcal{Q}} R_j$, $B_{\mathcal{Q},i} = \prod_{j\in\mathcal{Q}} B_{j,i}$ for all $i \in \mathcal{Q}$.
   Each party $i \in \mathcal{Q}$:
   (a) Computes $f_i = B_{\mathcal{Q},i} \cdot R_{\mathcal{Q}}^{-\mathsf{sk}_i}$, $\mathsf{tsk}_i = \mathsf{CLSolve}(f_i)$ and $\mathsf{tpk}_i = h^{\mathsf{tsk}_i}$.
   (b) Creates a proof $\mathsf{Pf}_{\mathsf{tpk}_i} = \Pi_{\mathsf{LinCL}}.\mathsf{Prove}(\{(f, R_{\mathcal{Q}}, B_{\mathcal{Q},i}, h, \mathsf{tpk}_i, \mathsf{pk}_i); (\mathsf{tsk}_i, \mathsf{sk}_i) : f^{\mathsf{tsk}_i} R_{\mathcal{Q}}^{\mathsf{sk}_i} = B_{\mathcal{Q},i},\ h^{\mathsf{tsk}_i} = \mathsf{tpk}_i,\ g_q^{\mathsf{sk}_i} = \mathsf{pk}_i\})$. (Section 2.3)
   (c) Publishes $(\mathsf{tpk}_i, \mathsf{Pf}_{\mathsf{tpk}_i})$
3. Let I be the set of parties i for which the (public, deterministic) verification of the proof $\mathsf{Pf}_{\mathsf{tpk}_i}$ accepts, and let $\mathcal{T}$ any set of $t+1$ parties in I (e.g. the first $t+1$ with respect to some pre-agreed indexing). The global public key tpk is $\mathsf{tpk} = \prod_{i\in\mathcal{T}} \mathsf{tpk}_i^{\lambda_i}$ where $\lambda_i = \prod_{k\in\mathcal{T}\setminus\{i\}} \frac{\beta-\alpha_j}{\alpha_i-\alpha_j}$

Fig. 6. Two-round DKG protocol Π_{DKG} with Unbiasable Public Key

4.2 One-round biasable public-key version

We now show a protocol that implements the functionality in Figure 7 in one round of communication. In this case, the functionality allows the adversary to bias the public key: the functionality sends some "temporary" public keys tpk, $\{\mathsf{tpk}_i\}_{i\in[n]}$ as well as temporary secret keys tsk_i for the corrupted parties, and then the adversary can choose to update the secret sharing polynomial by adding a contribution

$p'(X)$. This reflects the fact that in a one-round real protocol an adversary can wait until all honest parties have spoken, see all information it is allowed to, and in that moment then make one or more corrupted parties execute the PVSS honestly with sharing polynomials adding to some chosen $p'(X)$.

Functionality $\mathcal{F}_{\mathsf{BDKG}}$

$\mathcal{F}_{\mathsf{BDKG}}$ is parameterized by a DDH-hard cyclic group $\mathbb{H}$ of prime order q, with generator h. Let n and $1 \leq t \leq (n-1)/2$ be integers. Let $\beta, \alpha_1, \ldots, \alpha_n$ be pairwise distinct elements in $\mathbb{Z}_q$. $\mathcal{F}_{\mathsf{BDKG}}$ interacts with parties $ID_1, \ldots, ID_n$ and an adversary $\mathcal{S}$ that corrupts at most t parties. $\mathcal{F}_{\mathsf{BDKG}}$ works as follows:

1. Upon receiving $(\textsc{Gen}, sid, ID_i)$ from a honest party ID_i, forward $(\textsc{Gen}, sid, ID_i)$ to $\mathcal{S}$. When all honest parties have done this, continue.
2. Sample a random polynomial p of degree at most t. For every party $ID_j \in [n]$, set $\mathsf{tsk}_j = p(\alpha_j)$ and $\mathsf{tpk}_j = h^{\mathsf{tsk}_j}$, $\mathsf{tpk} = h^{p(\beta)}$.
3. Send $(\textsc{Keys}, sid, \{\mathsf{tpk}_j\}_{j\in[n]}, \mathsf{tpk}, \{\mathsf{tsk}_j\}_{j\in\mathsf{Corr}})$ to $\mathcal{S}$.
4. Upon receiving $(\textsc{Bias}, sid, p')$ from $\mathcal{S}$, where p' is a polynomial of degree at most t then update $\mathsf{tpk}' = \mathsf{tpk} \cdot h^{p'(\beta)}$, $\mathsf{tpk}'_j = \mathsf{tpk}_j \cdot h^{p'(\alpha_j)}$ and $\mathsf{tsk}'_j = \mathsf{tsk}_j + p'(\alpha_j)$ for all $j \in [n]$.
5. For all parties ID_i, send $(\textsc{Keys}, sid, \mathsf{tsk}'_i, \{\mathsf{tpk}'_j\}_{j\in[n]}, \mathsf{tpk}')$ to ID_i.

Fig. 7. Biasable Distributed Key Generation Functionality $\mathcal{F}_{\mathsf{BDKG}}$

As in the two-round protocol, every party j shares a secret s_j with the PVSS, sending $R_j = g_q^{r_j}$ $B_{j,i} = \mathsf{pk}_i^{r_j} f^{\sigma_{j,i}}$ where $\sigma_{j,i} = p_j(\alpha_i)$ are Shamir shares of $p_j(\beta) = s_j$. But now, they also publish the values $D_{j,i} := h^{\sigma_{j,i}} \in \mathbb{H}$. This allows every party to eventually compute the i-th threshold public key as $\mathsf{tpk}_i = \prod_{j\in\mathcal{Q}} h^{\sigma_{j,i}}$ where $\mathcal{Q}$ is again the set of parties that created the sharing honestly.

To be included in $\mathcal{Q}$, party j needs to prove not only that $(R_j, B_{j,i})$ form a correct PVSS sharing but also that $B_{j,i}$ and $D_{j,i}$ are consistent. In other words, we will need a NIZK proof $\mathsf{Pf}_{\mathsf{ExtSh}}$ for the relation

$$\mathcal{R}_{\mathsf{ExtSh}} = \{(f, g_q, h, R, (\mathsf{pk}_i)_{i\in[n]}, (B_i)_{i\in[n]}, (D_i)_{i\in[n]}; (r, p(X)) :$$
$$\deg p \leq t, R = g_q^r, \text{ and } \forall\, i \in [n],\ B_i = \mathsf{pk}_i^r f^{p(\alpha_i)}, D_i = h^{p(\alpha_i)}\}$$

We show how to accomplish this with a *constant-size proof* next.

As in qCLPVSS, we can reduce testing whether B_i are of the correct form with respect to R (i.e. $B_i = \mathsf{pk}_i^r f^{p(\alpha_i)}$ for $p(X) \in \mathbb{Z}_q[X]_{\leq t}$ and where $r \in \mathbb{Z}$ is such that g_q^r) to a DLEQ proof $g_q^r = R$, $U^r = V$. Moreover, thanks to the SCRAPE test, verifiers can locally check if $D_i = h^{\hat{p}(\alpha_i)}$ for some $\hat{p} \in \mathbb{Z}_q[X]_{\leq t}$.

We still need to guarantee that $p(X) = \hat{p}(X)$, i.e. the shares hidden by B_i and D_i are the same. It is enough to prove that $p(\alpha_i) = \hat{p}(\alpha_i)$ for all $i \in [t+1]$. We can do this by testing $\sum_{i=1}^{t+1} e_i p(\alpha_i) =^? \sum_{i=1}^{t+1} e_i \hat{p}(\alpha_i)$ for random $e_1, \ldots, e_{t+1} \in \mathbb{Z}_q$ sampled via the random oracle. This would guarantee the property with probability $1 - 1/q$ over the random choice of the e_i.

To test this we define $D = \prod_{i=1}^{t+1} D_i^{e_i}$ and $B = \prod_{i=1}^{t+1} B_i^{e_i}$, $M = \prod_{i=1}^{t+1} \mathsf{pk}_i^{e_i}$ (all of which can be computed publicly) and $d = \sum_{i=1}^{t} e_i p(\alpha_i)$ (computed privately by the prover). If the prover has been honest then $M^r f^d = B$. This suggests we can reduce the problem to proving existence of r in $\mathbb{Z}$ and d in $\mathbb{Z}_q$ with $g_q^r = R$, $U^r = V$, $M^r f^d = B$, $h^d = D$. We will indeed prove this is sound. Finally, this last statement can then be addressed with a proof similar to the Π_{LinCL} in Section 2.3, with the only difference that h, D are in a different group and d is in $\mathbb{Z}_q$. We remark this type of "mixed" statements have already been addressed in similar ways in papers such as [12,13,6].

We start by presenting this last proof, which we call Π_{MDLEQ} in Figure 8. Again, as in other similar protocols, the proof is paramtetrized by $C, A \in \mathbb{N}$ and to guarantee zero knowledge, we need that the witness is in an interval $[-S, S]$ and CS/A is negligible.

Theorem 9. *The interactive proof in Figure 8 has soundness error $1/C + \mathsf{negl}(\lambda)$ if the $\mathtt{RO}_C$ assumption holds. It is statistically zero-knowledge if the witness r is in $[-S, S]$ and CS/A is negligible. By the Fiat-Shamir heuristic, the non-interactive version has the same properties in the random oracle model.*

Zero-Knowledge Proof for "Mixed" Discrete Logarithm Equality

Zero Knowledge Proof for relation

$$\mathcal{R}_{\mathsf{MDLEQ}} = \{(g_q, U, M, f, h, R, V, B, D; r, d) : g_q^r = R, U^r = V, M^r f^d = B, h^d = D\}$$

Interactive version:
- Prover samples $r_* \leftarrow_\$ [0, A]$, $d_* \leftarrow_\$ \mathbb{Z}_q$, computes $R_* = g_q^r$, $V_* = U^{r_*}$, $B_* = M^{r_*} f^{d_*} = B$, $D_* = h^{d_*}$, sends R_*, V_*, B_*, D_* to verifier.
- Verifier samples $c \in [C]$.
- Prover computes and sends $u_r = r_* + cr$ (in $\mathbb{Z}$), $u_d = d_* + cd \mod q$.
- Verifier checks $g_q^{u_r} = R_* R^c$, $U^{u_r} = V_* \cdot V^c$, $M^{u_r} f^{u_d} = B_* \cdot B^c$, $h^{u_d} = D_* \cdot D^c$ and accepts if all checks pass.

Non-Interactive version:
Requires Random Oracle $\mathcal{H} : \{0, 1\}^* \to [C]$:
$\underline{\mathit{\Pi}_{\mathsf{MDLEQ}}.\mathsf{Prove}(\mathbf{X}, \mathbf{w})}$ (where $\mathbf{X} = (g_q, U, M, f, h, R, V, B, D)$, $\mathbf{w} = (r, d)$)
- Sample $r_* \leftarrow_\$ [0, A]$, $d_* \leftarrow_\$ \mathbb{Z}_q$, computes $R_* = g_q^r$, $V_* = U^{r_*}$, $B_* = M^{r_*} f^{d_*} = B$, $D_* = h^{d_*}$, $c = \mathcal{H}(\mathbf{X}, \mathbf{Y})$, where $\mathbf{Y} = (R_*, V_*, B_*, D_*)$, and $u_d = d_* + cd \mod q$, $u_r = r_* + cr$ (in $\mathbb{Z}$).
- Output $\mathsf{Pf}_{\mathsf{MDLEQ}} = (c, u_d, u_r)$

$\underline{\mathit{\Pi}_{\mathsf{MDLEQ}}.\mathsf{Verify}(\mathbf{X}, \mathsf{Pf}_{\mathsf{MDLEQ}})}$
Compute $R_* = R^{-c} g_q^{u_r}$, $V_* = V^{-c} U^{u_r}$, $B_* = B^{-c} M^{u_r} f^{u_d}$, $D_* = D^{-c} h^{u_d}$. Define $\mathbf{Y} = (R_*, V_*, B_*, D_*)$). Check $c = \mathcal{H}(\mathbf{X}, \mathbf{Y})$. Accept if that is the case.

Fig. 8. Zero-Knowledge Proof for "Mixed" Discrete Logarithm Equality

Proof. We present the proof in Appendix A.7

We use $\mathit{\Pi}_{\mathsf{MDLEQ}}$ as a building block for the proof $\mathit{\Pi}_{\mathsf{ExtSh}}$, Figure 9.

Theorem 10. *In the random oracle model, and assuming* $\mathtt{RO}_C$ *is hard for* CLGen*,* $\mathit{\Pi}_{\mathsf{ExtSh}}$ *(Figure 9) is a simulation sound proof for the relation* $\mathcal{R}_{\mathsf{ExtSh}}$ *with soundness error* $\epsilon_{\mathsf{MDLEQ}} + 1/C + 3/q + \mathsf{negl}(\lambda)$*, where* $\epsilon_{\mathsf{MDLEQ}}$ *is the soundness error of* $\mathit{\Pi}_{\mathsf{MDLEQ}}$*. If we use the same* C *in* $\mathit{\Pi}_{\mathsf{MDLEQ}}$ *as in this proof, the soundness error is* $2/C + 3/q + \mathsf{negl}(\lambda)$*. Moreover, it is zero-knowledge assuming* $\mathit{\Pi}_{\mathsf{MDLEQ}}$ *is.*

Proof. We present the proof in Appendix A.8

Finally, we present our one-round DKG protocol in Figure 10.

Theorem 11. *Under the* $\mathtt{DDH}\text{-}\mathtt{f}$ *(for privacy of* $\mathsf{qCLPVSS}$*) and* $\mathtt{RO}_C$ *(for verifiability of* $\mathsf{qCLPVSS}$ *and simulation soundness of* $\mathit{\Pi}_{\mathsf{LinCL}}$*) assumptions the protocol* $\mathit{\Pi}_{\mathsf{BDKG}}$ *in Figure 10 realizes* $\mathcal{F}_{\mathsf{BDKG}}$ *securely in the random model in the presence of a malicious static adversary corrupting* $t \leq \frac{n-1}{2}$ *parties.*

Proof. Appendix A.9

4.3 Communication complexity and comparison

In Table 1 we list the communication complexities of our two protocols, and compare them with the best (to the best of our knowledge) round-efficient distributed key generation protocols, both in the case of biasable and unbiasable public keys. In both cases, the comparison point is a scheme based on Paillier encryption. For the one-round, biasable public key case, we use the Fouque-Stern [22] protocol. For the two-round case, we use the suggested instantiation with Paillier of Katz' framework from [32], where we instantiate the NIZKs as in Fouque-Stern. We observe that the communication is dominated by the first summand and that therefore for a moderately large amount of parties, our DKG protocol will communicate less information as long as $k_{\hat{G}}$ is somewhat smaller than $3k_N$. Current security estimations ([20,5]) indicate this is the case for reasonable security parameters, e.g. 128-bit security. In fact, note that the dominating factor in our protocol consists of the n^2 share encryptions (n per party), which

Zero Knowledge Proof for correct "extended" sharing

Non-interactive Proof for the relation

$$\mathcal{R}_{\mathsf{ExtSh}} = \{(f, g_q, h, (\mathsf{pk}_i)_{i=1}^n, R, (B_i)_{i=1}^n), (D_i)_{i=1}^n); (p(X), r) : r \in \mathbb{Z},$$

$$p(X) \in \mathbb{Z}_q[X]_{\leq t}, R = g_q^r, B_i = \mathsf{pk}_i^r f^{p(\alpha_i)} \wedge D_i = h^{p(\alpha_i)} \; \forall i \in [n]\}.$$

The proof is parametrized by $C \in \mathbb{Z}$.
We assume a random oracle $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q[X]_{\leq n-t-2} \times [C]^n \times \mathbb{Z}_q^{t+1}$.
Let $\mathsf{X} := (f, g_q, h, (\mathsf{pk}_i)_{i=1}^n, R, (B_i)_{i=1}^n, (D_i)_{i=1}^n)$, $\mathsf{wit} := (p(X), r)$

$\Pi_{\mathsf{ExtSh}}.\mathsf{Prove}(\mathsf{X}; \mathsf{wit})$:

1. Compute $(m^*(X), c_1, \ldots, c_n, e_1, \ldots, e_{t+1}) = \mathcal{H}(\mathsf{X})$.
   Let $v_i = \prod_{j \in [n] \setminus \{i\}} (\alpha_i - \alpha_j)^{-1} \in \mathbb{Z}_q$.
2. Define $w_i = m^*(\alpha_i) \cdot v_i$ for each $i \in [n]$ where the evaluation and product is in $\mathbb{Z}_q$. From now on see w_i as integers (in $[0, q-1]$).
3. Compute $w_i' = w_i + c_i q$ over the integers for $i \in [n]$.
4. Compute $U = \prod_{i=1}^n \mathsf{pk}_i^{w_i'}$ and $V = \prod_{i=1}^n B_i^{w_i'}$.
5. Compute $d = \sum_{i=1}^{t+1} e_i p(\alpha_i)$, $B = \prod_{i=1}^{t+1} B_i^{e_i}$, $D = \prod_{i=1}^{t+1} D_i^{e_i}$, $M = \prod_{i=1}^{t+1} \mathsf{pk}_i^{e_i}$
6. Output $\mathsf{Pf}_{\mathsf{ExtSh}} = \Pi_{\mathsf{MDLEQ}}.\mathsf{Prove}(g_q, U, M, f, h, R, V, B, D; r, d)$ as in Figure 8. Recall this is a proof for the relations $g_q^r = R, U^r = V, M^r \cdot f^d = B, h^d = D$.

$\Pi_{\mathsf{Ext-Sh}}.\mathsf{Verify}(\mathbf{X}, \mathsf{Pf}_{\mathsf{Sh}})$:

1. Compute $(m^*(X), c_1, \ldots, c_n, e_1, \ldots, e_{t+1}) = \mathcal{H}(\mathsf{X})$.
2. Compute w_i and w_i' from $m^*(X)$ and the public information as the prover does.
3. Check $\prod_{i=1}^n D_i^{w_i} = 1_{\mathbb{H}}$. If not, output reject. Otherwise, continue.
4. Compute U, V, B, D, M from w_i', e_i and public information as the prover does.
5. Output $\Pi_{\mathsf{MDLEQ}}.\mathsf{Verify}((g_q, U, M, f, h, R, V, B, D, \mathsf{Pf}_{\mathsf{ExtSh}})$.

Fig. 9. Zero Knowledge Proof for correct "extended" sharing

are in fact roughly $\frac{1}{2}n^2$ CL-HSM ciphertexts[10], the Paillier based constructions communicate $3n^2 k_N$ bits ($\sim \frac{3}{2}n^2$ Paillier ciphertexts) and [5] estimates each CL-ciphertext to be 1.5 to 2.3 shorter than a Paillier ciphertext depending on the security parameter and for q of 224 bits. This estimation makes our communication 4.5 to 7 times smaller than the alternatives.

5 Application: YOSO MPC

In the YOSO model, parties can only speak once, *i.e.* after each party sends a message it can no longer participate in the execution. Moreover, the next committee of parties that take over the execution is selected at random and remains anonymous until they act. This requires a mechanism for transferring the secret state kept by each party in the comittee responsible for the current round to the committee responsible to the next round. As observed in [25], departing from the protocol [18] is a promising approach for keeping this state to a minimum. In the CDN protocol, the only secret state that parties must hold throughout the execution consists of shares of a secret key for a linearly homomorphic threshold encryption scheme, instead of requiring parties to hold shares of each intermediate gate output. In a recent work [6], linearly homomorphic threshold encryption based on the CL-framework was leveraged to realize this approach with a transparent setup by constructing a suitable DKG and a re-sharing protocol that allows for transferring secret key shares among committees (assuming receiver anonymous communication channels).

As a first step, we endow our PVSS scheme qCLPVSS with a publicly verifiable re-sharing scheme in order to construct an efficient mechanism for transfering secret state among committees in the YOSO model. This re-sharing mechanism already improves on the efficiency of the one proposed in [6]. We only need to publish a set of encrypted shares and a NIZK of re-sharing validity as many elements of $\mathbb{Z}_q$ as

[10] Since the element R_j is common to all encryptions by party j

One-round Distributed Key Generation Π_{BDKG}
Let q be a prime and $0 \leq t < n \leq q$ be positive integers. Let $\mathbb{H}$ be a cyclic group of order q generated by h.

Setup:
1. Parties run $\mathsf{pp} \leftarrow \mathsf{qCLPVSS.Setup}(1^\lambda, q, 1, t, n)$
2. Each party i runs $(\mathsf{sk}_i, \mathsf{pk}_i, \mathsf{Pf}_{\mathsf{pk}_i}) \leftarrow \mathsf{qCLPVSS.KeyGen}(\mathsf{pp}, i)$

Only parties who have produced $(\mathsf{sk}_i, \mathsf{pk}_i, \mathsf{Pf}_{\mathsf{pk}_i})$ that pass the verification $\mathsf{qCLPVSS.VerifyKey}$ are accepted to participate in the protocol.

Protocol:
In the only communication round, each party $j \in [n]$:
1. Samples uniformly random $s_j \leftarrow_\$ \mathbb{Z}_q$
2. Runs $(R_j, (B_{j,i})_{i\in[n]}, \cdot) \leftarrow \mathsf{qCLPVSS.Share}(\mathsf{pp}, (\mathsf{pk}_i)_{i\in[n]}, s_j)$. By this notation we mean we omit the proof of correct sharing, as we replace it by the one below. Let $p_j(X)$ the sharing polynomial, $\sigma_{j,i} = p_j(\alpha_i)$ the share for party i, obtained as part of the PVSS.
3. Computes $D_{j,i} = h^{\sigma_{j,i}}$ for all $i \in [n]$
4. Use $\Pi_{\mathsf{ExtSh}}.\mathsf{Prove}$ to compute a proof $\mathsf{Pf}_{\mathsf{ExtSh}\,j}$ for the statement $\exists r_j \in \mathbb{Z}$, $p_j(X) \in \mathbb{Z}_q[X]_{\leq t}$, such that $R = g_q^{r_j}$, $B_{j,i} = \mathsf{pk}_i^{r_j} \cdot f_j^{p_j(\alpha_i)}$ $\forall i \in [n]$, and $D_{j,i} = h^{p_j(\alpha_i)}$ $\forall i \in [n]$.
5. Publishes $(R_j, (B_{j,i})_{i\in[n]}, (D_{j,i})_{i\in[n]}, \mathsf{Pf}_{\mathsf{ExtSh}\,j})$

Global output:
Let $\mathcal{Q}$ be the set of j for which the (deterministic, public) verification $\Pi_{\mathsf{ExtSh}}.\mathsf{Verify}$ accepts $\mathsf{Pf}_{\mathsf{ExtSh}\,j}$. Then:
- For every $i \in [n]$, tpk_i is defined as $\mathsf{tpk}_i = \prod_{j\in\mathcal{Q}} D_{j,i}$.
- Let $\mathcal{T} = [t+1]$. The global public key tpk is $\mathsf{tpk} = \prod_{i\in\mathcal{T}} \mathsf{tpk}_i^{\lambda_i}$ where λ_i is the Lagrange interpolation coefficient $\lambda_i = \prod_{k\in\mathcal{T}\setminus\{i\}} \frac{\beta-\alpha_k}{\alpha_i-\alpha_k}$.

Private output:
Each party $i \in [n]$:
1. Computes $B_{\mathcal{Q},i} = \prod_{j\in\mathcal{Q}} B_{j,i}, R_{\mathcal{Q}} = \prod_{j\in\mathcal{Q}} R_j$
2. Computes $f_i = B_{\mathcal{Q},i} \cdot R_{\mathcal{Q}}^{-\mathsf{sk}_i}$ and outputs $\mathsf{tsk}_i = \mathsf{CLSolve}(f_i)$.

Fig. 10. One-round Distributed Key Generation (with biasable public key)

encrypted shares, whereas the protocol of [6] has each committee execute one VSS instances towards the next committee and one towards the second next committee. Later one, we show how the efficient encryption to the future scheme of YOLO YOSO can be combined with this approach to realize the full communication infrastructure needed to transfer state among committees.

5.1 Resharing

We consider how a set of parties who have a correct PVSS sharing of a secret with $\mathsf{qCLPVSS}$ can reshare this to a new set of parties. In the following we assume the case $k = 1$ (one secret in $\mathbb{Z}_q$) and we consider a starting set of n_0 parties, with privacy threshold t_0 and we denote their evaluation points $\overline{\alpha_1}, \ldots, \overline{\alpha_{n_0}}$ for the shares and $\overline{\beta}$ for the secret. Moreover, let $\overline{\mathsf{pk}_i}, \overline{\mathsf{sk}_i}$ their keys. Meanwhile for the next set of parties we have respectively $\alpha_1, \ldots, \alpha_{n_1}$, β and pk_i, sk_i respectively. Now given a secret s shared with degree-t_0 Shamir secret sharing, with shares $\overline{\sigma_i}$ for $i \in [n_0]$ we know that, for any set $\mathcal{T}$ of size t_0+1, $s = \sum_{i\in T} \overline{\lambda_i}\overline{\sigma_i}$ where $\overline{\lambda_i} = L_i(\beta)$ for $L_i = \mathrm{Lag}_{i,\mathcal{T},(\overline{\alpha_i})}$, i.e. $\overline{\lambda_i} = \prod_{j\in\mathcal{T}\setminus\{i\}}(\overline{\beta}-\overline{\alpha_j})(\overline{\alpha_i}-\overline{\alpha_j})^{-1}$. Since Shamir secret sharing is linear, it is enough that such a set $\mathcal{T}$ correctly reshare their shares to the new committee of parties: party i, having received $\sigma_{j,i}$ as a share of $\overline{\sigma_j}$ for each $j \in \mathcal{T}$, can then compute $\sum_{i\in\mathcal{T}} \overline{\lambda_i}\sigma_{j,i}$ and by linearity this will form a new sharing of s.

Note that in PVSS, we have the advantage that there is no need for dispute resolution: everyone can compute $\mathcal{T}$ by themselves, provided that there is a proof of correct sharing. This enables its use in the YOSO model, as share receivers do not need to speak at that point. We do need that there are at least $t_0 + 1$ honest parties in the first set, i.e. $2t_0 + 1 \leq n$.

Scheme	Comm. (bits)	Rounds	Bias Resist.	Assump.
Katz[32], using Paillier	$3n^2k_N + 2n^2\kappa$ $+(n^2 + tn + n)k_{\mathbb{H}}$	2	Yes	DCR
$\mathit{\Pi}_{\mathsf{DKG}}$	$(n^2 + n)k_{\hat{G}} + 3n\log(\bar{s})$ $+9n\kappa + nk_{\mathbb{H}}$	2	Yes	$\mathtt{DDH\text{-}f}$,$\mathtt{RO}_C$
Fouque-Stern [22]	$3n^2k_N + 2n^2\kappa$ $+(2n^2 + tn + n)k_{\mathbb{H}}$	1	No	DCR
$\mathit{\Pi}_{\mathsf{BDKG}}$	$(n^2 + n)k_{\hat{G}} + n\log(\bar{s})$ $+3n\kappa + n^2k_{\mathbb{H}}$	1	No	$\mathtt{DDH\text{-}f}$,$\mathtt{RO}_C$

Table 1. Comparison of DKG schemes for a DDH-hard group $\mathbb{H}$ where n is the total number of parties, t is the number of corrupted parties, $k_{\mathbb{H}}$ is the number of bits of an element of $\mathbb{H}$ which to simplify we set to $\log q$, k_N is the number of bits of the Paillier cryptosystem modulus N, $k_{\hat{G}}$ is the number of bits of a representation of an element in $\hat{G}$, $\bar{s}$ is the upper bound for the order of $\hat{G}^q$.

The crux of the protocol is proving a correct resharing. If $(\overline{R}, (\overline{B}_j)_{j\in[n_0]})$ is the original sharing, party j will create a polynomial with $p_j(\beta) = \overline{\sigma_j}$, use qCLPVSS for creating a sharing $(R_j, B_{j,i})$ where the $B_{j,i}$ encrypt $p_j(\alpha_i)$ and show not only correctness of this sharing, but also that $(\overline{R}, \overline{B}_j)$ decrypts to $p_j(\beta)$.

We show the resharing protocol in Figure 11 and later we explain the proof of resharing in more detail below. As for security, note that the IND2 security property of the PVSS directly guarantees that a set containing at most t_0 parties of the first committee and t_1 parties of the second can still not distinguish between sharings of two secrets. The soundness of the proofs will guarantee that a party is included in $\mathcal{Q}$ if they have reshared their share correctly. From $\mathcal{Q}$ parties can then determine $\mathcal{T}$.

Protocol for PVSS Resharing to a new committee

Input: A PVSS $(\overline{R}, (\overline{B_i})_{i\in[n_0]}$ of a secret $s \in \mathbb{Z}_q$

Output: A PVSS $(R, (B_i)_{i\in[n_1]}$ of the same secret $s \in \mathbb{Z}_q$

We assume at most t_0 corrupted parties in the first set and t_1 corrupted parties in the second. Moreover $2t_0 + 1 \leq n_0$ (to guarantee at least $t_0 + 1$ honest parties)

1. Every party $j \in [n_0]$:
   (a) Retrieves $\overline{\sigma}_j \leftarrow \mathsf{qCLPVSS.DecShare}(\mathsf{pp}, \overline{\mathsf{sk}_j}, \overline{R}, \overline{B_j})$
   (b) Chooses $p_j \in \mathbb{Z}_q[X]_{\leq t_1}$ uniformly at random such that $p_j(\beta) = \overline{\sigma_j}$
   (c) Chooses $r_j \leftarrow \mathcal{D}_q$ and computes $R_j = g_q^{r_j}, B_{j,i} = \mathsf{pk}_i^{r_j} f^{p_j(\alpha_i)}$. Let $\mathbf{X}_j := (g_q, h, f, (\mathsf{pk}_i)_{i\in[n_1]}, \overline{\mathsf{pk}_j}, \overline{R}, \overline{B_j}, R_j, (B_{j,i})_{i\in[n_1]})$, $\mathbf{w}_j := (\overline{\mathsf{sk}_j}, r_j, p_j)$.
   (d) Using $\mathit{\Pi}_{\mathsf{Resh}}$ in Figure 12 below compute a proof $\mathsf{Pf}_{\mathsf{Resh}\,j} = \mathit{\Pi}_{\mathsf{Resh}}.\mathsf{Prove}(\mathbf{X}_j; \mathbf{w}_j)$ for the relation given by $\deg p_j \leq t_1, g_q^{\overline{\mathsf{sk}_j}} = \overline{\mathsf{pk}_j}, \overline{B_j} = \overline{R}^{\overline{\mathsf{sk}_j}} \cdot f^{p_j(\beta)}, R_j = g_q^{r_j}$, and $B_{j,i} = \mathsf{pk}_i^{r_j} \cdot f^{p_j(\alpha_i)}\ \forall i \in [n_1]$.
   (e) Output $(R_j, (B_{j,i})_{i\in[n_1]}, \mathsf{Pf}_{\mathsf{Resh}\,j})$.
2. Let $\mathcal{Q}$ the set of parties j in $[n_0]$ for which $\mathsf{Pf}_{\mathsf{Resh}\,j}$ passes. Let $\mathcal{T} \subseteq \mathcal{Q}$ be a subset of $t_0 + 1$ parties. Then define $R = \sum_{j\in\mathcal{T}} R_j^{\overline{\lambda_j}}$, and $B_i = \sum_{j\in\mathcal{T}} B_{j,i}^{\overline{\lambda_j}}$ for $i \in [n_1]$, where $\overline{\lambda_j} = \sum_{k\in\mathcal{T}\setminus\{j\}}(\overline{\beta} - \overline{\alpha_k})(\overline{\alpha_j} - \overline{\alpha_k})^{-1}$ computed over $\mathbb{Z}_q$ and then considered as an integer in $[0, q-1]$.
3. Output $(R, (B_i)_{i\in[n_1]})$.

Fig. 11. Protocol for resharing to a new committee

We now detail the proof of resharing $\mathit{\Pi}_{\mathsf{Resh}}$(Figure 12). Consider

$$\mathcal{R}_{\mathsf{Resh}} = \{(g_q, h, f, (\mathsf{pk}_i)_{i\in[n_1]}, \overline{\mathsf{pk}}, \overline{R}, \overline{B}, R, (B_i)_{i\in[n_1]}); (\overline{\mathsf{sk}}, r, p(X)) : p \in \mathbb{Z}_q[X]_{\leq t},$$
$$r \in \mathbb{Z},\ g_q^{\overline{\mathsf{sk}}} = \overline{\mathsf{pk}},\ \overline{B} = \overline{R}^{\overline{\mathsf{sk}}} \cdot f^{p(\beta)},\ R = g_q^r,\ B_i = \mathsf{pk}_i^r f^{p(\alpha_i)}\ \forall i \in [n_1]\}.$$

Zero Knowledge Proof for correct resharing

Non-interactive Proof for the relation

$$\mathcal{R}_{\mathsf{Resh}} = \{(g_q, h, f, (\mathsf{pk}_i)_{i\in[n_1]}, \overline{\mathsf{pk}}, \overline{R}, \overline{B}, R, (B_i)_{i\in[n_1]}); (\overline{\mathsf{sk}}, r, p(X)) : r \in \mathbb{Z},$$

$$p(X) \in \mathbb{Z}_q[X]_{\le t},\ g_q^{\overline{\mathsf{sk}}} = \overline{\mathsf{pk}},\ \overline{B} = \overline{R}^{\overline{\mathsf{sk}}} \cdot f^{p(\beta)},\ R = g_q^r,\ B_i = \mathsf{pk}_i^r f^{p(\alpha_i)}\ \forall i \in [n_1]\}.$$

The proof is parametrized by $C \in \mathbb{Z}$.
We assume a random oracle $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q[X]_{\le n-t-1} \times [C]^{n+1}$.
Let $\mathsf{X} := (g_q, h, f, (\mathsf{pk}_i)_{i\in[n_1]}, \overline{\mathsf{pk}}, \overline{R}, \overline{B}, R, (B_i)_{i\in[n_1]})$, $\mathsf{wit} := (\overline{\mathsf{sk}}, r, p(X))$. For ease of notation let $\alpha_0 = \beta$.

$\Pi_{\mathsf{Resh}}.\mathsf{Prove}(\mathsf{X};\mathsf{wit})$:

1. Compute $(m^*(X), c_0, c_1, \dots, c_n) = \mathcal{H}(\mathsf{X})$. For $i \in [0,n]$ let $v_i = \prod_{j\in[0,n]\setminus\{i\}}(\alpha_i - \alpha_j)^{-1} \in \mathbb{Z}_q$.
2. Define $w_i = m^*(\alpha_i) \cdot v_i$ for each $i \in [0,n]$ where the evaluation and product is in $\mathbb{Z}_q$. From now on see w_i as integers (in $[0, q-1]$).
3. Compute $w_i' = w_i + c_i q$ over the integers for $i \in [n]$.
4. Compute $U = \prod_{i=1}^n \mathsf{pk}_i^{w_i'}$ and $V = \prod_{i=1}^n B_i^{w_i'}$. Also let $\overline{R_0} = \overline{R}^{w_0'}$ and $\overline{B_0} = \overline{B}^{w_0'}$
5. Compute a proof, using Π_{LinCL} (Figure 3, Section 2.3) of the following relation
$$\{(U, \overline{R_0}, V, g_q, \overline{\mathsf{pk}}, R); (r, \overline{\mathsf{sk}}) : U^r \cdot (R_0)^{\overline{\mathsf{sk}}} = V \cdot B_0,\ g_q^{\overline{\mathsf{sk}}} = \overline{\mathsf{pk}},\ g_q^r = R\}$$
6. Output this proof as $\mathsf{Pf}_{\mathsf{Resh}}$.

$\Pi_{\mathsf{Ext-Sh}}.\mathsf{Verify}(\mathsf{X}, \mathsf{Pf}_{\mathsf{Resh}})$:

1. Compute $(m^*(X), c_0, c_1, \dots, c_n) = \mathcal{H}(\mathsf{X})$.
2. Compute w_i, w_i', U, V, $\overline{R_0}$, $\overline{B_0}$ from $m^*(X)$ and the public information as the prover does.
3. Verify $\mathsf{Pf}_{\mathsf{Resh}}$ is a valid proof for the relation above.

Fig. 12. Zero Knowledge Proof for correct resharing

This is the usual $\mathcal{R}_{\mathsf{Sh}}$ augmented with the fact that the secret $p(\beta)$ is the value committed by $(\overline{\mathsf{pk}}, \overline{B}) = (g_q^{\overline{\mathsf{sk}}}, \overline{R}^{\overline{\mathsf{sk}}} \cdot f^{p(\beta)})$. We will use the SCRAPE test, now applied to the $n+1$ evaluation points $\beta, \alpha_1, \dots, \alpha_n$. We rename $\alpha_0 := \beta$ for simplicity. Then we need to sample m^* of degree $n-t-1$ (rather than $n-t-2$ as before), and define v_i, now for all $i \in [0,n]$ and including α_0. Given $w_i = m^*(\alpha_i) \cdot v_i$, the SCRAPE test implies $\sum_{i=0}^n p(\alpha_i) w_i = 0$ for any p of $\deg p \le t$.

Now if we compute $U = \prod_{i=1}^n \mathsf{pk}_i$ and $V = \prod_{i=1}^n B_i^{w_i}$ as in previous proofs, we can eventually reduce the task to showing existence of $r, \overline{\mathsf{sk}}$ with $g_q^r = R$, $g_q^{\overline{\mathsf{sk}}} = \overline{\mathsf{pk}}$ and $U^r \cdot (\overline{R}^{w_0})^{\mathsf{sk}} = V \cdot \overline{B}^{w_0}$ which can be addressed with the proof Π_{LinCL} (Figure 3, Section 2.3). However, there is the same problem with soundness as in Section 3.2, caused by the fact that the adversary could have concocted $B_i = \mathsf{pk}_i^r \cdot f^{p(\alpha_i)} \cdot H_i$ (and now also $\overline{B} = \overline{R}^{\overline{\mathsf{sk}}} \cdot f^{p(\beta)} \cdot H_0$) so that $\prod_{i=0}^n H_i^{w_i}$ cancels out. This is solved exactly in the same way as in Section 3.2 by randomizing $w_i' = w_i + c_i q$ and using the rough order assumption.

Theorem 12. *In the random oracle model, and assuming $\mathtt{RO}_C$ is hard for* CLGen*, Π_{Resh} in Figure 12 is a proof for the relation $\mathcal{R}_{\mathsf{Resh}}$ with soundness error $\epsilon_{\mathsf{LinCL}} + 1/C + 1/q + \mathsf{negl}(\lambda)$, where $\epsilon_{\mathsf{LinCL}}$ is the soundness error of $\mathtt{NIZK}_{\mathsf{LinCL}}$. It is zero knowledge assuming $\mathtt{NIZK}_{\mathsf{LinCL}}$ is.*

Proof. The proof follows analogously to Theorem 7, with the changes above.

5.2 Realizing Efficient YOSO MPC

Departing from our qCLPVSS PVSS scheme and the associated resharing scheme in Figure 11, we realize an efficient YOSO MPC protocol by combining the DKG and preprocessing/online phases from [6] with our PVSS. The protocol of [6] first generates a shard key for a linearly homomorphic threshold encryption scheme based on the CL-framework, which is then used to generate encrypted Beaver triples. In an online phase, parties use distributed decryption to obtain the necessary information for evaluating private multiplications using the preprocessed encrypted Beaver triples. However, at every round, the current committee of parties must reshare the secret key towards the next committee. We aim at replacing the resharing scheme of [6] with our scheme from Figure 11.

At first, we assume we have public keys for the next committee despite it being anonymous, and later argue about how to remove this assumption. Each party in the first committee to obtain shares of the secret key via the DKG of [6] converts them into shares of our qCLPVSS scheme. This can be done using standard tricks for share conversion or simply by a single execution of an inefficient YOSO MPC that publishes qCLPVSS shares given shares in a different format. Once a committee has qCLPVSS shares of the secret key, it can use our resharing scheme from Figure 11 to efficiently transfer those to the next committee at every round of the MPC protocol of [6].

This simple application of our resharing scheme still requires each committee to know public keys for the next random anonymous committee. While this could be done by means of Random-index RPIR [27] or ideal receiver anonymous communication channels (RACC), we would like to perform the necessary encryption towards the next anonymous committee in a more efficient way. In order to do so, one can use the YOLO YOSO [10] encryption to the future scheme based on mixnets with publicly verifiable proofs of shuffle correctness (and the associated scheme for authententication from the past). These schemes allows for encrypting a message under a public key associated to a randomly chosen party without learning their identity, later allowing the recipient to sign messages by proving that they indeed received the ciphertext. Since the YOLO YOSO construction can be realized from proof of correctness shuffle, it can be implemented in our setting by using a proof system [2] that works over linearly homomorphic encryption schemes, such as those in the CL-framework. Hence, we can obtain a more efficient realization of YOSO MPC based on the protocol of [6] and our PVSS scheme with resharing qCLPVSS that only uses transparent setup and does not require ideal RACCs.

References

1. Anasuya Acharya, Carmit Hazay, Vladimir Kolesnikov, and Manoj Prabhakaran. SCALES - MPC with small clients and larger ephemeral servers. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part II*, volume 13748 of *LNCS*, pages 502–531. Springer, Heidelberg, November 2022.
2. Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 263–280. Springer, Heidelberg, April 2012.
3. Fabrice Benhamouda, Craig Gentry, Sergey Gorbunov, Shai Halevi, Hugo Krawczyk, Chengyu Lin, Tal Rabin, and Leonid Reyzin. Can a public blockchain keep a secret? In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 260–290. Springer, Heidelberg, November 2020.
4. Fabrice Boudot and Jacques Traoré. Efficient publicly verifiable secret sharing schemes with fast or delayed recovery. In Vijay Varadharajan and Yi Mu, editors, *ICICS 99*, volume 1726 of *LNCS*, pages 87–102. Springer, Heidelberg, November 1999.
5. Cyril Bouvier, Guilhem Castagnos, Laurent Imbert, and Fabien Laguillaumie. I want to ride my BICYCL : BICYCL implements cryptography in class groups. *J. Cryptol.*, 36(3):17, 2023.
6. Lennart Braun, Ivan Damgård, and Claudio Orlandi. Secure multiparty computation from threshold encryption based on class groups. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part I*, volume 14081 of *Lecture Notes in Computer Science*, pages 613–645. Springer, 2023.
7. Matteo Campanelli, Bernardo David, Hamidreza Khoshakhlagh, Anders Konring, and Jesper Buus Nielsen. Encryption to the future - A paradigm for sending secret messages to future (anonymous) committees. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part III*, volume 13793 of *LNCS*, pages 151–180. Springer, Heidelberg, December 2022.
8. Ignacio Cascudo and Bernardo David. SCRAPE: Scalable randomness attested by public entities. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *ACNS 17*, volume 10355 of *LNCS*, pages 537–556. Springer, Heidelberg, July 2017.
9. Ignacio Cascudo and Bernardo David. ALBATROSS: Publicly AttestabLe BATched Randomness based On Secret Sharing. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 311–341. Springer, Heidelberg, December 2020.
10. Ignacio Cascudo, Bernardo David, Lydia Garms, and Anders Konring. YOLO YOSO: Fast and simple encryption and secret sharing in the YOSO model. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part I*, volume 13791 of *LNCS*, pages 651–680. Springer, Heidelberg, December 2022.

11. Ignacio Cascudo, Bernardo David, Omer Shlomovits, and Denis Varlakov. Mt. random: Multi-tiered randomness beacons. In Mehdi Tibouchi and Xiaofeng Wang, editors, *Applied Cryptography and Network Security - 21st International Conference, ACNS 2023, Kyoto, Japan, June 19-22, 2023, Proceedings, Part II*, volume 13906 of *Lecture Notes in Computer Science*, pages 645–674. Springer, 2023.
12. Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Two-party ECDSA from hash proof systems and efficient instantiations. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 191–221. Springer, Heidelberg, August 2019.
13. Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Bandwidth-efficient threshold EC-DSA. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 266–296. Springer, Heidelberg, May 2020.
14. Guilhem Castagnos, Laurent Imbert, and Fabien Laguillaumie. Encryption switching protocols revisited: Switching modulo p. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 255–287. Springer, Heidelberg, August 2017.
15. Guilhem Castagnos and Fabien Laguillaumie. Linearly homomorphic encryption from DDH. In Kaisa Nyberg, editor, *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings*, volume 9048 of *Lecture Notes in Computer Science*, pages 487–505. Springer, 2015.
16. Guilhem Castagnos, Fabien Laguillaumie, and Ida Tucker. Practical fully secure unrestricted inner product functional encryption modulo p. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 733–764. Springer, Heidelberg, December 2018.
17. Arka Rai Choudhuri, Aarushi Goel, Matthew Green, Abhishek Jain, and Gabriel Kaptchuk. Fluid MPC: Secure multiparty computation with dynamic participants. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 94–123, Virtual Event, August 2021. Springer, Heidelberg.
18. Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 280–299. Springer, Heidelberg, May 2001.
19. Bernardo David, Giovanni Deligios, Aarushi Goel, Yuval Ishai, Anders Konring, Eyal Kushilevitz, Chen-Da Liu-Zhang, and Varun Narayanan. Perfect MPC over layered graphs. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part I*, volume 14081 of *Lecture Notes in Computer Science*, pages 360–392. Springer, 2023.
20. Samuel Dobson, Steven D. Galbraith, and Benjamin Smith. Trustless unknown-order groups. *Mathematical Cryptology*, 1(2):25–39, 2022.
21. Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the Fiat-Shamir transform. In Steven D. Galbraith and Mridul Nandi, editors, *INDOCRYPT 2012*, volume 7668 of *LNCS*, pages 60–79. Springer, Heidelberg, December 2012.
22. Pierre-Alain Fouque and Jacques Stern. One round threshold discrete-log key generation without private channels. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 300–316. Springer, Heidelberg, February 2001.
23. Eiichiro Fujisaki and Tatsuaki Okamoto. A practical and provably secure scheme for publicly verifiable secret sharing and its applications. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 32–46. Springer, Heidelberg, May / June 1998.
24. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 295–310. Springer, Heidelberg, May 1999.
25. Craig Gentry, Shai Halevi, Hugo Krawczyk, Bernardo Magri, Jesper Buus Nielsen, Tal Rabin, and Sophia Yakoubov. YOSO: You only speak once - secure MPC with stateless ephemeral roles. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 64–93, Virtual Event, August 2021. Springer, Heidelberg.
26. Craig Gentry, Shai Halevi, and Vadim Lyubashevsky. Practical non-interactive publicly verifiable secret sharing with thousands of parties. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 458–487. Springer, Heidelberg, May / June 2022.
27. Craig Gentry, Shai Halevi, Bernardo Magri, Jesper Buus Nielsen, and Sophia Yakoubov. Random-index PIR and applications. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part III*, volume 13044 of *LNCS*, pages 32–61. Springer, Heidelberg, November 2021.
28. Jens Groth. Non-interactive distributed key generation and key resharing. Cryptology ePrint Archive, Paper 2021/339, 2021. `https://eprint.iacr.org/2021/339`.
29. Kobi Gurkan, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. Aggregatable distributed key generation. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 147–176. Springer, Heidelberg, October 2021.

30. Somayeh Heidarvand and Jorge L. Villar. Public verifiability from pairings in secret sharing schemes. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC 2008*, volume 5381 of *LNCS*, pages 294–308. Springer, Heidelberg, August 2009.
31. Aniket Kate, Easwar Vivek Mangipudi, Pratyay Mukherjee, Hamza Saleem, and Sri Aravinda Krishnan Thyagarajan. Non-interactive vss using class groups and application to dkg. Cryptology ePrint Archive, Paper 2023/451, 2023. `https://eprint.iacr.org/2023/451`.
32. Jonathan Katz. Round optimal robust distributed key generation. Cryptology ePrint Archive, Paper 2023/1094, 2023. `https://eprint.iacr.org/2023/1094`.
33. Torben P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract) (rump session). In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 522–526. Springer, Heidelberg, April 1991.
34. Rahul Rachuri and Peter Scholl. Le mans: Dynamic and fluid MPC for dishonest majority. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 719–749. Springer, Heidelberg, August 2022.
35. A. Ruiz and J. L. Villar. Publicly verifiable secret sharing from paillier's cryptosystem. In *WEWoRC 2005–Western European Workshop on Research in Cryptology*, 2005.
36. Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 148–164. Springer, Heidelberg, August 1999.
37. Markus Stadler. Publicly verifiable secret sharing. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 190–199. Springer, Heidelberg, May 1996.
38. Ida Tucker. *Functional encryption and distributed signatures based on projective hash functions, the benefit of class groups. (Chiffrement fonctionnel et signatures distribuées fondés sur des fonctions de hachage à projection, l'apport des groupes de classe)*. PhD thesis, University of Lyon, France, 2020.

Reference Material

A Proofs

A.1 Proof of Theorem 1

First consider the following claim

Theorem 13. *Let $\mathbb{F}$ be a field, and $\alpha_1, \dots, \alpha_n$ be pairwise distinct elements of $\mathbb{F}$. Then, for every polynomial $h \in \mathbb{F}[X]$ of degree at most $n-2$, then $\sum_{i=1}^{n} v_i \cdot h(\alpha_i) = 0$ where $v_i = \prod_{j \in [n] \setminus \{i\}} (\alpha_i - \alpha_j)^{-1}$*

Proof. For $i \in [n]$, let

$$L_i(X) = \mathrm{Lag}_{i,[n],A} = \prod_{j \in [n] \setminus \{i\}} \frac{X - \alpha_j}{\alpha_i - \alpha_j}$$

be the Lagrange interpolation polynomial. L_i is a polynomial in $\mathbb{F}[X]$ of degree at most $n-1$ such that $L_i(\alpha_k) = 1$ if $k = i$ and $L_i(\alpha_k) = 0$ if $k \in [n] \setminus \{i\}$. Moreover, the coefficient of X^{n-1} in $L_i(X)$ is

$$\prod_{j \in [n] \setminus \{i\}} \frac{1}{\alpha_i - \alpha_j} = v_i$$

Then $h(X)$ and $\widehat{h}(X) = \sum_{i=1}^{n} h(\alpha_i) L_i(X)$ are polynomials of degree $< n$ which coincide in their evaluations on $\{\alpha_1, \dots, \alpha_n\}$, hence by uniqueness of the interpolation polynomial, $h(X) = \widehat{h}(X)$. Compare now the coefficient of X^{n-1} on both sides. On the left, this is 0 because $\deg h \leq n-2$; on the right, this is $\sum_{i=1}^{n} v_i h(\alpha_i)$. Since they need to be equal $\sum_{i=1}^{n} v_i h(\alpha_i) = 0$ and we get the claim.

Now we can prove Theorem 1

Proof (of Theorem 1).
Consider the quantity $T := \sum_{i=1}^{n} v_i m^*(\alpha_i) y_i$ where m^* is sampled uniformly in $\mathbb{F}[X]_{\leq n-d-2}$.

If there exists a polynomial $p \in \mathbb{F}[X]$ of degree $\leq d$ such that $y_i = p(\alpha_i)$ for all $i \in [n]$, let $h(X) = p(X) \cdot m^*(X)$. This is a polynomial of degree at most $n-2$ with $T = \sum_{i=1}^{n} v_i h(\alpha_i)$. By Theorem 13, $T = 0$.

Now consider the set $V = \{(v_1 m^*(\alpha_1), v_2 m^*(\alpha_2), \dots, v_n m^*(\alpha_n)) : m^*(X) \in \mathbb{Z}_q[X]_{\leq n-d-2}\}$. This is a vector space of dimension $n-d-1$ inside $\mathbb{F}^n$. By linear algebra, its orthogonal space [11] $V^\perp$ has dimension $d+1$, i.e. it is exactly $\{(p(\alpha_1), p(\alpha_2), \dots, p(\alpha_n)) : \deg p \leq d\}$. Therefore any $\mathbf{y} = (y_1, y_2, \dots, y_n)$ that is *not* of the form $(p(\alpha_1), p(\alpha_2), \dots, p(\alpha_n))$ with $\deg p \leq d$ cannot be in $V^\perp$.

For such a $\mathbf{y}$, consider the linear map $L_\mathbf{y} : V \to \mathbb{F}$ where each element $(v_1 m^*(\alpha_1), v_2 m^*(\alpha_2), \dots, v_n m^*(\alpha_n))$ of V is taken into $\sum_{i=1}^{n} v_i m^*(\alpha_i) y_i$. $L_\mathbf{y}$ cannot be identically zero by the above, so its kernel must be a strict subspace of V, of dimension one unit less, i.e. $n-d-2$. This implies that $|\mathrm{Ker}\ L_\mathbf{y}|/|V| = 1/|\mathbb{F}|$, so $\Pr[T = 0] = 1/|\mathbb{F}|$ in this case.

A.2 Proof of PVSS correctness (Theorem 4)

Proof. If all parties in $[n]$ honestly create keys, then for all i we have $\mathsf{pk}_i = g_q^{\mathsf{sk}_i}$ for some sk_i. If the dealer is honest the values R and B_i are of the form $R = g_q^r$, $B_i = \mathsf{pk}_i^r f^{p(\alpha_i)}$ where $p(X)$ is of degree $t+k-1$ and $p(\beta_j) = s_j$ are the coordinates of the secret. Then clearly $\mathsf{DecShare}$, when honestly applied to (R, B_i), first creates $f_i = B_i \cdot R^{-\mathsf{sk}_i} = \mathsf{pk}_i^r f^{p(\alpha_i)} g_q^{-r\mathsf{sk}_i} = \mathsf{pk}_i^r f^{p(\alpha_i)} \mathsf{pk}_i^{-r} = f^{p(\alpha_i)}$ and then $A_i = \mathsf{CLSolve}(f_i) = p(\alpha_i)$. Therefore given a set $\mathcal{T}$ of parties of size at least $t+k$ who correctly decrypted their shares, and a subset $\mathcal{T}'$ of exactly $t+k$ parties, we have that the reconstructed values are $\prod_{i \in \mathcal{T}'} p(\alpha_i) L_i(\beta_j) = p(\beta_j) = s_j$, for each $j \in [k]$, by Lagrange interpolation.

[11] the space of all $\mathbf{w} \in \mathbb{F}^n$ such that $\sum v_i w_i = 0$ for all $\mathbf{v} \in V$

A.3 Proof of PVSS IND2-secrecy (Theorem 6)

The proof is similar to some extent to the one for the DHPVSS in [10], although with the difference that here we only need to assume `DDH-f` instead of `DDH` in the whole group. Another difference is that the proof of [10] only showed the weaker notion of IND1-security (where the challenger instead of the adversary choose the secrets) although it can be easily adapted to show IND2.

By the previous lemma, it is enough to prove the theorem under the `DDH-qf` assumption. Let $\mathcal{A}$ be an adversary corrupting a set of up to t parties. Without loss of generality, we assume $\mathcal{A}$ corrupts $[n-t+1,n]$. We prove that if this adversary is such that

$$\Pr\Big[\mathrm{Game}^{\mathsf{ind\text{-}secrecy},1}_{\mathcal{A},\mathsf{PVSS}}(\lambda)=1\Big]-\Pr\Big[\mathrm{Game}^{\mathsf{ind\text{-}secrecy},0}_{\mathcal{A},\mathsf{PVSS}}(\lambda)=1\Big]=\epsilon$$

where $\mathrm{Game}^{\mathsf{ind\text{-}secrecy},b}_{\mathcal{A},\mathsf{PVSS}}$ is as in Definition 6, then we can construct $\mathcal{B}$ such that $\mathsf{Adv}^{\mathtt{DDH\text{-}qf}}_{\mathcal{B}}(\lambda)=\frac{|\epsilon|}{2(n-t)}$. This is enough as if $\frac{|\epsilon|}{2(n-t)}$ is negligible then so is $|\epsilon|$.

On input a tuple (g_q, X, Y, Z), where $X=g_q^x$, $Y=g_q^y$ where $x,y \leftarrow_{\$} \mathcal{D}_q$ and $Z=g^{xy}$ or $Z=g^{xy}f^u$ for uniformly random $u\in\mathbb{Z}_q$, $\mathcal{B}$ proceeds as follows:

- $\mathcal{B}$ chooses b at random in $\{0,1\}$
- $\mathcal{B}$ chooses a non-corrupted party $i_*\in[n-t]$ at random and sets $\mathsf{pk}_{i_*}=Y$. $\mathcal{B}$ simulates the proof $\mathsf{Pf}_{\mathsf{pk}_{i^*}}$ using the zero-knowledge simulator of $\mathtt{NIZKPoK}_{\mathsf{DL}}$
- For all $i\in[n-t]\setminus\{i_*\}$, $\mathcal{B}$ runs $(\mathsf{sk}_i,\mathsf{pk}_i,\mathsf{Pf}_{\mathsf{pk}_i})\leftarrow\mathsf{KeyGen}(\lambda)$. $\mathcal{B}$ sends $\{(\mathsf{pk}_i,\mathsf{Pf}_{\mathsf{pk}_i}):i\in[n-t]\}$ to $\mathcal{A}$.
- $\mathcal{B}$ waits for $\mathcal{A}$ to reply with pk_i and proofs $\mathsf{Pf}_{\mathsf{pk}_i}$ for each $i\in[n-t+1,n]$, and $\mathbf{s}^0,\mathbf{s}^1$ in the space of secrets $\mathbb{Z}_q^k$.
- $\mathcal{B}$ now checks $\mathsf{Pf}_{\mathsf{pk}_i}$, sending $\perp$ and aborting if these do not pass. Otherwise it extracts sk_i from $\mathsf{Pf}_{\mathsf{pk}_i}$ for $i\in[n-t+1,n]$ using the fact that $\mathsf{Pf}_{\mathsf{pk}_i}$ is a proof of knowledge and therefore it has a witness extractor.
- $\mathcal{B}$ creates Shamir sharings of $\mathbf{s}^0,\mathbf{s}^1$ with polynomials $p^0(X)$, $p^1(X)$ such that corrupted shares coincide, i.e. $p^c(\beta_j)=s^c_j$, for all $j\in[k]$ and $c\in\{0,1\}$ and $p^0(\alpha_i)=p^1(\alpha_i)$ for $i\in[n-t+1,n]$. For $i\in[n]$, let $\sigma_i^c=p^c(\alpha_i)$. Moreover, since $\sigma_i^0=\sigma_i^1$ for $i\in[n-t+1,n]$, we simply denote those values σ_i for $i\in[n-t+1,n]$.
- $\mathcal{B}$ defines $R=X$ and:
$$B_i=\begin{cases}R^{\mathsf{sk}_i}\cdot f^{\sigma_i^0} & i\in[i_*-1]\\ R^{\mathsf{sk}_i}\cdot f^{\sigma_i^1} & i\in[i_*+1,n-t]\\ R^{\mathsf{sk}_i}\cdot f^{\sigma_i} & i\in[n-t+1,n]\\ Z\cdot f^{\sigma_i^b} & i=i_*\end{cases}$$
- $\mathcal{B}$ simulates $\mathsf{Pf}_{\mathsf{Sh}}$ using the zero-knowledge simulator of $\mathtt{NIZK}_{\mathsf{Sh}}$ and sends $(R,B_1,\dots,B_n,\mathsf{Pf}_{\mathsf{Sh}})$ to $\mathcal{A}$
- $\mathcal{A}$ makes a guess b^* and $\mathcal{B}$ outputs 1 if $b=b^*$ and 0 otherwise. Let W be the event that $\mathcal{B}$ outputs 1, i.e. $b=b^*$.

Let W be the event that $\mathcal{B}$ outputs 1, i.e. that $b^*=b$.

If $Z=Z_1=g_q^{xy}f^u$ for uniform $u\in\mathbb{Z}_q$, then $(R,B_1,\dots,B_n,\mathsf{Pf}_{\mathsf{Sh}})$ is independent from b; indeed, the only computation involving b is that of B_{i_*}, but B_{i_*} is of the form $g_q^{xy}f^{u+\sigma_i^b}$ and u is uniform in $\mathbb{Z}_q$. Therefore clearly the guess b^* of $\mathcal{A}$ is independent from b, and hence $\Pr[W]=1/2$ in this case.

On the other hand if $Z=Z_0=g_q^{xy}=X^y$, then since $\mathsf{pk}_{i_*}=Y$, implicitely we have $y=\mathsf{sk}_{i_*}$ and since in addition $X=R$ we have $B_{i_*}=Z\cdot f^{\sigma^b_{i_*}}=R^{\mathsf{sk}_{i_*}}f^{\sigma_i^b}$. Therefore, condition to being in this case, we have the following facts:

- For every $j\in[n-t-1]$, the views of $\mathcal{A}$ when $i_*=j, b=0$ and $i_*=j+1, b=1$ are identical.
- If $i_*=n-t$ and $b=0$, $(R,B_1,\dots,B_n,\mathsf{Pf}_{\mathsf{Sh}})$ is distributed as a PVSS sharing of $\mathbf{s}^0$ (all $B_i=R^{\mathsf{sk}_i}f^{\sigma_i^0}$ for all i) and $\mathcal{A}$ is playing $\mathrm{Game}^{\mathsf{ind\text{-}secrecy},0}_{\mathcal{A},\mathsf{PVSS}}$.
- If $i_*=1$ and $b=1$, then $(R,C_1,\dots,C_n,\mathsf{Pf}_{\mathsf{Sh}})$ is distributed as a PVSS sharing of $\mathbf{s}^1$ and $\mathcal{A}$ is playing $\mathrm{Game}^{\mathsf{ind\text{-}secrecy},1}_{\mathcal{A},\mathsf{PVSS}}$.

Let $P_{j,c} = \Pr[W|i_* = j, b = c]$. Then the first item above implies

$$P_{j,0} = Pr[b^* = 0|i_* = j, b = 0] = \Pr[b^* = 0|i_* = j+1, b = 1] = 1 - P_{j+1,1}$$

and therefore $P_{j,0} + P_{j+1,1} = 1$ for all $j \in [n-t-1]$, while the second and third items imply $P_{n-t,0} = 1 - \Pr[\mathrm{Game}^{\text{ind-secrecy},0}_{\mathcal{A},\mathsf{PVSS}} = 1]$ and $P_{1,1} = \Pr[\mathrm{Game}^{\text{ind-secrecy},1}_{\mathcal{A},\mathsf{PVSS}} = 1]$ respectively, which by assumption implies $P_{n-t,0} + P_{1,1} = 1 + \epsilon$.

Therefore in this case

$$\Pr[W] = \frac{1}{2(n-t)} \sum_{j=1}^{n-t} \sum_{j=0}^{1} P_{j,c} = \frac{1}{2(n-t)}(n-t+\epsilon) = \frac{1}{2} + \frac{\epsilon}{2(n-t)}.$$

Thus $\mathsf{Adv}^{\mathtt{DDH\text{-}qf}}_{\mathcal{B}}(\lambda) = \frac{|\epsilon|}{2(n-t)}$ which concludes the proof.

A.4 Proof of Lemma 2

Proof. Let $\mathcal{B}$ be an adversary for DDH-f. We construct $\mathcal{A}$ an adversary for DDH-qf that uses $\mathcal{B}$ as follows: on challenge $(\mathsf{pp}_{CL}, X, Y, Z_b)$, $\mathcal{A}$ samples $c, d \leftarrow_\$ \mathbb{Z}_q$, constructs $X' = X \cdot f^c$, $Y' = Y \cdot f^d$ and $Z'_b = Z_b \cdot f^{cd}$ and outputs $\mathcal{B}(\mathsf{pp}_{CL}, X', Y', Z'_b)$.

We show $\mathsf{Adv}^{\mathtt{DDH\text{-}qf}}_{\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathtt{DDH\text{-}f}}_{\mathcal{B}}(\lambda) - \mathsf{negl}(\lambda)$, by showing that the challenge (X', Y', Z'_b) for $\mathcal{B}$ is distributed statistically close to that in the DDH-f experiment. Indeed, since $X = g_q^x$, $Y = g_q^y$ then clearly $X' = g^{x'}$, $Y' = g^{y'}$ for x', y' distributed statistically close to $\mathcal{D}$. More precisely (recall s is the order of g_q), $x' = x \mod s$, $x' = c \mod q$, and $y' = y \mod s$, $y' = d \mod q$. Now note that $x'y' = xy \mod s$ and $x'y' = cd \mod q$. It is now clear that $Z'_b = Z_b f^{cd} = g^{x'y'} f^u$ where $u = 0$ if $b = 0$ and uniformly random in $\mathbb{Z}_q$ if $u = 1$. We have indeed shown that the distribution received by $\mathcal{B}$ is statistically close to the one in the DDH-f experiment and hence we obtain the result

A.5 Proof of $\mathtt{NIZK}_{Sh}$ of Sharing Correctness (Theorem 7)

Completeness. In order to check completeness clearly we need to argue that if the statement is correct then $g_q^r = R$ and $U^r = V$ for the U, V constructed (deterministically) from the statement. Note that $V = \prod_{i=1}^n B_i^{w'_i} = \prod_{i=1}^n \mathsf{pk}_i^{r \cdot w'_i} f^{p(\alpha_i) \cdot w'_i} = U^r \cdot \prod_{i=1}^n f^{p(\alpha_i) \cdot w'_i}$ Since f generates a group of order q, $w'_i = w_i + c_i q$ and $w_i = m^*(\alpha_i) \cdot v_i$, $\prod_{i=1}^n f^{p(\alpha_i) \cdot w'_i} = f^{\sum_{i=1}^n p(\alpha_i) m^*(\alpha_i) \cdot v_i}$ Now we apply Theorem 1 that ensures $\sum_{i=1}^n p(\alpha_i) m^*(\alpha_i) \cdot v_i = 0 \mod q$. Therefore indeed $V = U^r$ and completeness follows from completeness of $\mathtt{NIZK}_{\mathsf{DLEQ}}$

Soundness. If $\Pi_{\mathsf{Sh}}.\mathsf{Verify}((f, g_q, (\mathsf{pk}_i)_{i=1}^n, R, (B_i)_{i=1}^n), \mathsf{Pf}_{\mathsf{Sh}})$ accepts then, except with probability ϵ_{DLEQ}, we have $g_q^r = R$, $U^r = V$ for some r where $U = \prod_{i=1}^n \mathsf{pk}_i^{w'_i}$ and $V = \prod_{i=1}^n B_i^{w'_i}$.

Call $J_i = \mathsf{pk}_i^{-r} B_i$ for all i. Since $U^r = V$, we have $\prod_{i=1}^n J_i^{w'_i} = 1$. Since $\hat{G} = \hat{G}^q \times F$, we can write $J_i = H_i f^{a_i}$ for some $a_i \in \mathbb{Z}_q$ and some $H_i \in \hat{G}^q$.

Moreover $\prod_{i=1}^n J_i^{w'_i} = \prod_{i=1}^n H_i^{w'_i} \cdot f^{\sum_{i=1}^n a_i w'_i}$ where the first factor is in the group $\hat{G}^q$ and the second is in F. Therefore $\prod_{i=1}^n H_i^{w'_i} = 1$, $f^{\sum_{i=1}^n a_i w'_i} = 1$.

The second equality implies $\sum_{i=1}^n a_i w'_i = 0 \mod q$, hence also $\sum_{i=1}^n a_i w_i = 0 \bmod q$ and since $w_i = v_i \cdot m^*(\alpha_i)$ in $\mathbb{Z}_q$, then by Theorem 1, except with probability $1/q$ we have that $a_i = p(\alpha_i)$ for some polynomial $p \in \mathbb{Z}_q[X]$ of degree at most $t + k - 1$, i.e. a_i are Shamir shares of some secret.

Now we consider the other equality $\prod_{i=1}^n H_i^{w'_i} = 1$ where $H_i \in \hat{G}^q$.

There are two cases:

- $H_i = 1 \; \forall i \in [n]$. Then we have $M_i = f^{a_i}$ and therefore $B_i = \mathsf{pk}_i^r f^{a_i}$ where a_i are correct Shamir shares and r is such that $R = g_q^r$, so the shares are correct.
- Some $H_i \neq 1$. Then by Lemma 3, except with probability $1/C$, H_i is of order smaller than C. But this implies that if the prover could create H_i, not all one, such that $\prod_{i=1}^n H_i^{w'_i} = 1$ with probability larger than $1/C$, then there has to be a prime $p < C$ such that p divides the order of H_i, hence

also the order of $\hat{G}$. In this case the prover knows that ρ, the randomness used by CLGen does *not* come from the distribution $\mathcal{D}_C^{rough}$ as defined in the $\mathtt{RO}_C$ assumption. Therefore this should only happen with negligible probability, or otherwise the prover would be a distinguisher that breaks this assumption.

Putting everything together we see that if $\Pi_{\mathsf{Sh}}.\mathsf{Verify}$ accepts $\mathsf{Pf}_{\mathsf{Sh}}$, then except with probability $\epsilon_{\mathsf{DLEQ}} + 1/q + 1/C + \mathsf{negl}(\lambda)$, $\mathsf{pk}_i^{-r} B_i = f^{a_i}$ with $g_q^r = R$, $a_i = p(\alpha_i)$ for some polynomial $p \in \mathbb{Z}_q[X]$ of degree at most $t + k - 1$. Hence the statement is in the language given by relation $\mathcal{R}_{\mathsf{Sh}}$.

Zero Knowledge. All that is sent by the prover is $\mathtt{NIZK}_{\mathsf{DLEQ}}.\mathsf{Prove}((g_q, U, R, V); r)$, where all arguments of the statement g_q, R, U, V can be deterministically computed from the statement of $\mathtt{NIZK}_{\mathsf{Sh}}$. Therefore, the proof is zero knowledge if $\mathtt{NIZK}_{\mathsf{DLEQ}}$ is.

A.6 Proof of Theorem 8

Proof. Let $\mathsf{Corr} \subseteq [n]$ be the set of the parties corrupted by the adversary, where $|\mathsf{Corr}| \leq t$ and let $\mathsf{Honest} = [n] \setminus \mathsf{Corr}$ the set of honest parties. We construct a simulator $\mathcal{S}$ that interacts with $\mathcal{F}_{\mathsf{DKG}}$ and the adversary $\mathcal{A}$, such that the view of the latter in the interaction with $\mathcal{S}$ and $\mathcal{F}_{\mathsf{DKG}}$ is indistinguishable from its view in the real execution of the protocol.

$\mathcal{S}$ acts as follows. It first extracts sk_i from the proofs $\mathsf{Pf}_{\mathsf{pk}_i}$ for $i \in \mathsf{Corr}$, using the fact that in $\mathsf{qCLPVSS}$, these are proofs of knowledge. Whenever it receives $(\textsc{Gen}, sid, ID_j)$ from $\mathcal{F}_{\mathsf{DKG}}$, such that party j is honest, then it runs step 1 of protocol Π_{DKG} honestly for party j as dealer, thereby sampling $s_j \in \mathbb{Z}_q$ uniformly at random, creating and publishing

$$(R_j, (B_{j,i})_{i\in[n]}, \mathsf{Pf}_{\mathsf{Sh}j}) \leftarrow \mathsf{qCLPVSS}.\mathsf{Share}(\mathsf{pp}, (\mathsf{pk}_i)_{i\in[n]}, s_j).$$

It also stores the Shamir shares $\sigma_{j,i}, i \in [n]$ of s_j created as part of running $\mathsf{qCLPVSS}.\mathsf{Share}$. Moreover, $\mathcal{S}$ adds j to $\mathcal{Q}$.

When $\mathcal{A}$ posts a message on the bulletin on behalf of party $j \in \mathsf{Corr}$, $\mathcal{S}$ tries to parse it as $(R_j, (B_{j,i})_{i\in[n]}, \mathsf{Pf}_{\mathsf{Sh}j})$ and runs the verification of $\mathsf{Pf}_{\mathsf{Sh}j}$. If this passes, it adds j to $\mathcal{Q}$.

Let $\mathcal{Q}_{\mathsf{Corr}} = \mathcal{Q} \cap \mathsf{Corr}$, i.e. the set of all corrupted parties for which the adversary has sent correct information in the previous step.

$\mathcal{S}$ uses the extracted sk_i to obtain $\sigma_{j,i}$ from $R_j, B_{j,i}$ for all pairs (i, j) such that both $i, j \in \mathcal{Q}_{\mathsf{Corr}}$. Concretely $\sigma_{j,i} = \mathsf{CLSolve}(B_{j,i} R_j^{-\mathsf{sk}_i})$. For every $i \in C \cap \mathcal{Q}$, $\mathcal{S}$ now computes $\mathsf{tsk}_i = \sum_{j\in\mathcal{Q}} \sigma_{j,i}$ and sends $(\textsc{Gen}, sid, ID_i)$, $(\textsc{SetShare}, sid, ID_i, \mathsf{tsk}_i)$ to $\mathcal{F}_{\mathsf{DKG}}$. This can be done because $\mathcal{S}$ knows all $\sigma_{j,i}$ for honest j (which it has simulated) and for adversarial j in $\mathcal{Q}$ (which it has obtained).

For every corrupted party, $\mathcal{S}$ now computes $\mathsf{tsk}_i = \sum_{j\in\mathcal{Q}} \sigma_{j,i}$ and sends $(\textsc{Gen}, sid, ID_i)$, $(\textsc{SetShare}, sid, ID_i, \mathsf{tsk}_i)$ to $\mathcal{F}_{\mathsf{DKG}}$.

The functionality sends the messages $(\textsc{Keys}, sid, \mathsf{tsk}_i, \{\mathsf{tpk}_j\}_{j\in[n]}, \mathsf{tpk})$ for corrupted parties i to $\mathcal{S}$. Now $\mathcal{S}$ executes step 2 for each honest party i by publishing instead the tpk_i received from the functionality and using the simulator of the NIZK to create a simulated proof $\mathsf{Pf}_{\mathsf{tpk}_i}$ for $(f, R_{\mathcal{Q}}, B_{\mathcal{Q},i}, h, \mathsf{tpk}_i, \mathsf{pk}_i)$ where all the other parts of the statement are as in the protocol.

Finally $\mathcal{S}$ waits for $\mathcal{A}$ to post $(\mathsf{tpk}_i, \mathsf{Pf}_{\mathsf{tpk}_i})$ for $i \in \mathcal{Q}_{\mathsf{Corr}}$. $\mathcal{S}$ defines C to be the indices $i \in \mathcal{Q}_{\mathsf{Corr}}$ for which $\mathcal{A}$ fails to post an accepting proof. $\mathcal{S}$ sends $(\textsc{Abort}, sid, C)$ to the functionality.

We show that the execution with $\mathcal{S}$ and $\mathcal{F}_{\mathsf{DKG}}$ is indistinguishable of an execution of the real protocol Π_{DKG}. First, notice that all messages produced by $\mathcal{S}$ in the first step are exactly as in the protocol, and $\mathcal{Q}$ is exactly as it would be in Π_{DKG}. The values tsk_i, $i \in \mathcal{Q}_{\mathsf{Corr}}$ computed by $\mathcal{S}$ are exactly the same as the adversary would obtain in the protocol, and hence so are the tpk_i, $i \in \mathcal{Q}_{\mathsf{Corr}}$ computed by the functionality. Finally, for every pair honest party j, note the information about the vector $(\sigma_{j,i})_{i\in\mathsf{Honest}}$ known by the adversary is exactly the fact that $\sigma_{j,i} = p_j(\alpha_i)$ for some polynomial of degree t such that $p(\alpha_i) = \sigma_{j,i}$ for $i \in \mathsf{Corr}$ and $(R_j, B_{j,i}, \mathsf{Pf}_{\mathsf{Sh}j})$ does not reveal more information beyond that. In order to see this assume without loss of generality that $|\mathsf{Corr}| = t$. Then by properties of interpolation the set of polynomials p_j with $p_j(\alpha_i) = \sigma_{j,i}$ for all $i \in \mathsf{Corr}$ has exactly q elements, and each gives a different $p_j(\beta)$. So any additional information that $(R_j, (B_{j,i})_{i\in[n]}, \mathsf{Pf}_{\mathsf{Sh}j})$ gives about p_j would translate in additional information about $p_j(\beta)$, contradicting the privacy property of $\mathsf{qCLPVSS}$. Moreover, by the simulation soundness of the proof and the IND-CPA security of the share encryption, the adversary cannot use the

tpk_i, encrypted shares and NIZK of sharing correctness from simulated honest parties to create a tpk_i, encrypted shares and a valid NIZK of sharing correctness for a corrupted party that are correlated to a simulated honest party's tpk_i and shares.

Therefore, the adversary has no additional information about $\sigma_i = (\sum_{j\in\mathcal{Q}} \sigma_{j,i})_{i\in\mathsf{Honest}}$ beyond the fact that $\sigma_i = p_j(\alpha_i)$ for a uniformly random polynomial conditioned to $p(\alpha_i) = \mathsf{tsk}_i$ for $i \in \mathsf{Corr}$, which is exactly the same distribution the functionality uses to choose tsk_i. In other words the tpk_i and hence tpk seen by the adversary in the simulation as distributed as in the protocol. Finally, by the zero knowledge property of $\mathsf{Pf}_{\mathsf{tpk}_i}$, this is still true about $(\mathsf{tpk}_i, \mathsf{Pf}_{\mathsf{tpk}_i})_{i\in\mathsf{Honest}}$.

A.7 Proof of Theorem 9

Proof. Correctness: It follows easily using the fact that $\langle f\rangle$ and $\mathbb{H}$ are groups of order q.

Soundness: Suppose that a malicious (PPT) prover can generate an instance $(g_q, U, M, f, h, R, V, B, D)$ which is not in the language but passes the proof with probability more than $1/C$. Then there must be two different challenges c and c' such that the prover can compute respective responses (u_r, u_d) and (u'_r, u'_d) that make the proof be accepted. This means $g_q^{u_r-u'_r} = R^{c-c'}$, $U^{u_r-u'_r} = V^{c-c'}$, $M^{u_r-u'r} f^{u_d-u'_d} = B^{c-c'}$, $h^{u_d-u'_d} = D^{c-c'}$. Now if $c-c'$ is invertible modulo the order of $\hat{G}$, let a be this inverse, i.e. $a(c-c') = 1 \bmod \mathrm{ord}(\hat{G})$. Note a is also an inverse of $c-c'$ modulo q, because q divides $ord(\hat{G})$. Then clearly clearly $d = (u_r - u'_r)a$, $r = (u_r - u'_r)a \bmod q$ makes (r, d) a witness of the relation, which is a contradiction.

If $c-c'$ is not invertible modulo $\mathrm{ord}(\hat{G})$ it must mean that $\gcd(c-c', ord(\hat{G})) \neq 1$ and hence the order of $\hat{G}$ is divisible by a prime smaller than C and the prover can distinguish this fact. This should not be possible with probability larger than $\mathsf{negl}(\lambda)$ or this would contradict the $\mathtt{RO}_C$ assumption.

Zero Knowledge: For a given instance in the language and a challenge c, we can simulate a conversation that is distributed statistically close to a real one as follows. Sample u_d uniformly at random in $\mathbb{Z}_q$ and sample $u_r \leftarrow_{\$} [-SC, SC + A]$.

Compute $R_* = R^{-c} g_q^{u_r}$, $V_* = V^{-c} U^{u_r}$, $B_* = B^{-c} M^{u_r} f^{u_d}$, $D_* = D^{-c} h^{u_d}$. Clearly $D_* = h^{-dc+u_d}$ is distributed uniformly in $\mathbb{H}$. On the other hand we have $R_* = g_q^{-rc+u_r}$, $V_* = U^{-rc+u_r}$, $B_* = M^{-rc+u_r} f^{-dc+u_d}$. Here f^{-dc+u_d} is distributed as f^{d_*} in the protocol, and independently from the integer $-rc + u_r$. On the other hand $-rc + u_r$ is distributed uniformly in $-rc + [-SC, SC + A]$ for a fixed value $-rc$ which is in $[-SC, SC+A]$. Then the -$rc + u_r$ is distributed uniformly in an interval of size $-2SC + A$ that contains A. Under the assumption that SC/A is negligible, the distributions are statistically close.

A.8 Proof of Theorem 10

Proof. Correctness. It follows from the explanation in Section 4.2.

Soundness. If the proof passes, then with probability at most $\epsilon_{\mathsf{MDLEQ}}$, we have $g_q^r = R, U^r = V, M^r f^d = B, h^d = D$ for some $r \in \mathbb{Z}$ and $d \in \mathbb{Z}_q$. Reasoning exactly the same as in the proof of Theorem 7 we have that $g_q^r = R, U^r = V$ imply that in that case, $R = g_q^r$ and $B_i = \mathsf{pk}_i^r f^{p(\alpha_i)}$ for $i \in [n]$ for some $p(X) \in \mathbb{Z}_q[X]_{\le t}$, except with probability $1/C + 1/q + \mathsf{negl}(\lambda)$. Moreover, if $\prod_{i=1}^n D_i^{w_i} = 1_{\mathbb{H}}$ passes then by Lemma 3, except with probability $1/q$ we have $D_i = h^{\hat{p}(\alpha_i)}$ for some $\hat{p}$ of degree at most t. Therefore all of the above occurs except with probability at most $1/C + 2/q + \mathsf{negl}(\lambda)$. We assume this is the case in the following.

Now the statement $M^r f^d = B$ ensures that $(\prod_{i=1}^{t+1} \mathsf{pk}_i^{re_i}) \cdot f^d = \prod_{i=1}^{t+1} B_i^{e_i} = \prod_{i=1}^{t+1} \mathsf{pk}_i^{re_i} f^{e_i p(\alpha_i)}$. Hence clearly $f^d = \prod_{i=1}^{t+1} f^{e_i p(\alpha_i)}$ so $d = \sum_{i=1}^{t+1} e_i p(\alpha_i) \bmod q$. On the other hand $h^d = D$ implies $h^d = \prod_{i=1}^{t+1} D_i^{e_i} = h^{\hat{p}(\alpha_i)}$, therefore $d = \sum_{i=1}^{t+1} e_i \hat{p}(\alpha_i) \bmod q$. Then $\sum_{i=1}^{t+1} e_i(p(\alpha_i) - \hat{p}(\alpha_i)) = 0 \bmod q$. Under the uniform random choice of e_i in $\mathbb{Z}_q$, we have that if $p(\alpha_i) \neq \hat{p}(\alpha_i)$ for some i, the above would only happen with probability at most $1/q$. Therefore except with probability $1/q$ we have $p(\alpha_i) = \hat{p}(\alpha_i)$ for all $i \in [t + 1]$. Since they are polynomials of degree t, then this implies $p(X) = \hat{p}(X)$.

For here we conclude that if the statement is incorrect the proof will pass with probability at most $\epsilon_{\mathsf{MDLEQ}} + 1/C + 3/q + \mathsf{negl}(\lambda)$.

Moreover, being based on the Fiat-Shamir transform in the random oracle model, we observe that by the result of [21] this scheme is simulation sound.

Zero Knowledge. It follows from the fact that $\mathsf{Pf}_{\mathsf{MDLEQ}}$ is the only thing sent by the prover and all the rest can be simulated (in fact it is computed by the verifier).

A.9 Proof of Theorem 11

Proof. Let $\mathsf{Corr} \subseteq [n]$ be the set of the parties corrupted by the adversary, where $|\mathsf{Corr}| \leq t$ and let $\mathsf{Honest} = [n] \setminus \mathsf{Corr}$ the set of honest parties. We construct a simulator $\mathcal{S}$ that interacts with $\mathcal{F}_{\mathsf{BDKG}}$ and the adversary $\mathcal{A}$, such that the view of the latter in the interaction with $\mathcal{S}$ and $\mathcal{F}_{\mathsf{BDKG}}$ is indistinguishable from its view in the real execution of the protocol.

$\mathcal{S}$ acts as follows. It generates public keys and secret keys for the honest parties. It extracts sk_i from the proofs $\mathsf{Pf}_{\mathsf{pk}_i}$ for $i \in \mathsf{Corr}$, using the fact that in qCLPVSS, these are proofs of knowledge. Whenever it receives $(\textsc{Gen}, sid, ID_j)$ from $\mathcal{F}_{\mathsf{BDKG}}$, such that party j is honest, then *except for the last honest party*, it runs the only round of Π_{BDKG} honestly for party j as dealer, thereby sampling $s_j \in \mathbb{Z}_q$ uniformly at random, creating $(R_j, (B_{j,i})_{i\in[n]}, \cdot) \leftarrow \mathsf{qCLPVSS.Share}(\mathsf{pp}, (\mathsf{pk}_i)_{i\in[n]}, s_j)$, $D_{i,j} = h^{\sigma_{i,j}}$ and a proof $\mathsf{Pf}_{\mathsf{ExtSh}j}$. It publishes $(R_j, (B_{j,i})_{i\in[n]}, (D_{i,j})_{i\in[n]}, \mathsf{Pf}_{\mathsf{ExtSh}j})$. It also stores the Shamir shares $\sigma_{j,i}, i \in [n]$ created as part of running qCLPVSS.Share. Moreover, $\mathcal{S}$ adds j to $\mathcal{Q}_*$.

When $\mathcal{A}$ posts a message on the bulletin on behalf of party $j \in \mathsf{Corr}$ before $\mathcal{S}$ has posted the message for the last honest party, $\mathcal{S}$ tries to parse it as $(R_j, (B_{j,i})_{i\in[n]}, (D_{j,i})_{i\in[n]}, \mathsf{Pf}_{\mathsf{ExtSh}j})$ and runs the verification of $\mathsf{Pf}_{\mathsf{Sh}j}$. If this passes then $\mathcal{S}$ adds j to $\mathcal{Q}_*$. $\mathcal{S}$ computes $\widetilde{\mathsf{tpk}}_i = \prod_{j\in\mathcal{Q}_*} D_{j,i}$ for all i, uses its knowledge of all sk_i to obtain the $\sigma_{j,i}$ sent by corrupted parties so far and defines $\widetilde{\mathsf{tsk}}_i = \sum_{j\in\mathcal{Q}_*} \sigma_{j,i}$.

Now $\mathcal{S}$ waits until the functionality sends $(\textsc{keys}, sid, \{\mathsf{tpk}_i\}_{i\in[n]}, \mathsf{tpk}, \mathsf{tsk}_{i\in\mathsf{Corr}})$, and computes the message for the last honest party j as follows: it defines $D_{j,i} = \mathsf{tpk}_i \cdot \widetilde{\mathsf{tpk}}_i^{-1}$ for all i and $\sigma_{j,i} = \mathsf{tsk}_i - \widetilde{\mathsf{tsk}}_i$ for $i \in \mathsf{Corr}$. If $h^{\sigma_{j,i}} \neq D_{j,i}$, it aborts. Otherwise sample $r_j \leftarrow \mathcal{D}_q$, compute $R_j = g_q^r$, $B_{j,i} = \mathsf{pk}_i^{r_j} f^{\sigma_{j,i}}$ for all $i \in \mathsf{Corr}$, and $B_{j,i} = \mathsf{pk}_i^{r_j} f^{\sigma'_{j,i}}$ for honest i, where $\sigma'_{j,i}$ are such that there exists a polynomial p_j of degree at most t with $p_j(\alpha_i) = \sigma_{j,i}$ for i corrupt, $p_j(\alpha_i) = \sigma'_{j,i}$ for i honest, and compute the proof $\mathsf{Pf}_{\mathsf{ExtSh}j}$ using the zero knowledge simulator. Post all values as honest party j would do.

Now for all corrupt parties that have still not posted anything, whenever $\mathcal{A}$ posts a message on behalf of them, $\mathcal{S}$ runs the verification of $\mathsf{Pf}_{\mathsf{Sh}j}$ and if it passes, it uses the knowledge of all sk_i to obtain the polynomial p_j used for sharing (if this does not exist, it aborts). Let $\mathcal{B}$ all corrupt parties that have been posted proofs that pass the verification after $\mathcal{S}$ published the message of the last honest party. Then $\mathcal{S}$ defines $p' = \sum_{j\in\mathcal{B}} p_j$ and sends $(\textsc{bias}, sid, p')$ to the functionality.

We show that the execution with $\mathcal{S}$ and $\mathcal{F}_{\mathsf{BDKG}}$ is indistinguishable of an execution of the real protocol Π_{DKG}. First, notice that all messages produced by $\mathcal{S}$ on behalf of honest parties are exactly as in the protocol, except for the last one. The message for the last honest party guarantees that the current public keys and the current secret keys corresponding to corrupted parties are the ones that the functionality sent. By the zero knowledge property of the proof, $\mathsf{Pf}_{\mathsf{ExtSh}j}$ is distributed as an honest proof. At this point, the adversary can obtain from the published information the current $\mathsf{tpk}, \mathsf{tpk}_i$ for all parties and tsk_i for corrupted parties i that the functionality has sent to $\mathcal{S}$. Now the adversary posts additional PVSS for parties that have not yet spoken and the simulator translates this into a polynomial p' that corresponds to the sum of the sharing polynomials of all corrupted parties that have sent after $\mathcal{S}$ published a message on behalf the last honest party. By the simulation soundness of the proof and the IND-CPA security of the share encryption, the adversary cannot use the tpk_i, encrypted shares and NIZK of sharing correctness from simulated honest parties to create a tpk_i, encrypted shares and a valid NIZK of sharing correctness for a corrupted party that are correlated to a simulated honest party's tpk_i and shares. Therefore the functionality will update the public and private keys exactly as the adversary does with these last messages and the output will be the same as in a real protocol. Finally, throughout the simulation, we note that the simulator only aborts when a proof of a false statement by the adversary has passed, which we know only happens with negligible probability.

B The public verifiable secret sharing scheme DHPVSS from [10]

For comparison, and since we are following its blueprint, we recall the DDHPVSS scheme in YOLO YOSO [10] for a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q where DDH is assumed to be hard. As we have already mentioned, the main qualitative difference with our scheme is that here parties can in principle only reconstruct $g^s \in \mathbb{G}$ rather than elements $s \in \mathbb{Z}_q$. Since the sharing can actually also be done knowing only g^s and not s, we can see this as a PVSS for secrets in $\mathbb{G}$. We describe the case $k = 1$ (one element in $\mathbb{G}$ as secret) only, as this is the case described in [10], but we observe it is trivial to adapt this to $k > 1$

with the same modifications as in our scheme. The PVSS assumes non-interactive zero knowledge proofs of discrete logarithm $\texttt{NIZKPoK}_{\mathsf{DL}}$ and proof of discrete logarithm equality $\texttt{NIZK}_{\mathsf{DLEQ}}$ which in this case of known order groups can easily be constructed as Schnorr proofs.

$\mathsf{DHPVSS.Setup}(q, t, n, \lambda)$:

1. Specify a set of pairwise distinct points $\{\beta, \alpha_1, \ldots, \alpha_n\} \subset \mathbb{Z}_q$. These points determine also $v_i = \prod_{j\in[n]\setminus\{i\}}(\alpha_i - \alpha_j)^{-1}$ for every $i \in [n]$. Let $\mathsf{pp_{Sh}} = (q, t, \beta, (\alpha_i)_{i\in[n]}, (v_i)_{i\in[n]})$
2. Specify a description of a random oracle $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q[X]_{\le n-t-2}$
3. The output is then $\mathsf{pp_{Sh}}$.

$\mathsf{DHPVSS.KeyGen}(\mathsf{pp}, i)$:

1. Sample $\mathsf{sk}_i \leftarrow\in \mathbb{Z}_q$ and compute $\mathsf{pk}_i = g^{\mathsf{sk}_i}$.
2. Create proof $\mathsf{Pf}_{\mathsf{pk}_i} = \texttt{NIZKPoK}_{DL}.\mathsf{Prove}(\{((g, \mathsf{pk}); \mathsf{sk}_i) : \mathsf{pk}_i = g^{\mathsf{sk}_i}\})$
3. $\mathrm{Output}(\mathsf{sk}_i, \mathsf{pk}_i, \mathsf{Pf}_{\mathsf{pk}_i})$.

$\mathsf{DHPVSS.VerifyKey}(\mathsf{pp}, i, \mathsf{pk}_i, \mathsf{Pf}_{\mathsf{pk}_i})$: Run verification of $\mathsf{Pf}_{\mathsf{pk}_i}$ and output its result.

$\mathsf{DHPVSS.Dist}(\mathsf{pp}, (\mathsf{pk}_i)_{i\in[n]}, g^s)$, where $s \in \mathbb{Z}_q$:

1. Create a Shamir sharing of s: compute g^{σ_i} where $\sigma_i = p(\alpha_i)$ for a uniform polynomial in the set of polynomials in $\mathbb{Z}_q[X]_{\le t}$ with $p(\beta) = s$. This can be done even without explicitly knowing s, by sampling $p'(X) \leftarrow_\$ \mathbb{Z}_q[X]_{\le t-1}$ and computing $g^{\sigma_i} = g^s \cdot g^{(\alpha_i-\beta)\cdot p'(\alpha_i)}$ (since this induces a correctly distributed $p(X) = s + (X - \beta) \cdot p'(X)$).
2. Sample $r \leftarrow_\$ \mathbb{Z}_q$ and compute $R = g^r$.
3. Create $B_i = \mathsf{pk}_i^r \cdot g^{\sigma_i}$.
4. Create the sharing proof $\mathsf{Pf_{Sh}} = \texttt{NIZK}_{Sh}((g, (\mathsf{pk}_i)_{i=1}^n, R, (B_i)_{i=1}^n); (p, r) : \deg p \le t, R = g^r, B_i = \mathsf{pk}_i^r g^{p(\alpha_i)}\ \forall i \in [n])$ as follows:
   - Sample $m^*(X) = \mathcal{H}(\mathsf{pk}_1, \ldots, \mathsf{pk}_n, B_1, \ldots, B_n)$ where $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q[X]_{\le n-t-2}$.
   - Compute $V = \prod_{i=1}^n \mathsf{pk}_i^{v_i\cdot m^*(\alpha_i)}$ and $U = \prod_{i=1}^n B_i^{v_i\cdot m^*(\alpha_i)}$.
   - Compute $\mathsf{Pf_{Sh}} = \texttt{NIZK}_{DLEQ}.\mathsf{Prove}((g, U, R, V); r) : g^r = R \wedge U^r = V)$
5. Output $(R, B_1, \ldots, B_n, \mathsf{Pf_{Sh}})$. We define $C_i := (R, B_i)$ for all $i \in [n]$.

$\mathsf{DHPVSS.VerifySharing}(\mathsf{pp}, (\mathsf{pk}_i)_{i\in[n]}, (R, B_1, \ldots, B_n, \mathsf{Pf_{Sh}}))$: Run the verification of $\texttt{NIZK}_{Sh}$ by constructing m^*, U, V from the arguments as in Dist and then using $\texttt{NIZK}_{DLEQ}.\mathsf{Verify}((g, U, R, V); r) : g^r = R \wedge U^r = V)$.

$\mathsf{DHPVSS.DecShare}(\mathsf{pp}, \mathsf{pk}_i, \mathsf{sk}_i, C_i)$, where $C_i = (R, B_i)$:

1. Compute $S_i = B_i \cdot R^{-\mathsf{sk}_i}$. Let $M_i = B_i S_i^{-1} = R^{\mathsf{sk}_i}$
2. Compute $\mathsf{Pf}_{\mathsf{Dec}i} = \texttt{NIZK}_{DLEQ}.\mathsf{Prove}((g, R, \mathsf{pk}_i, M_i); \mathsf{sk}_i) : g^{\mathsf{sk}_i} = \mathsf{pk}_i, R^{\mathsf{sk}_i} = M_i)$.
3. Output $(S_i, \mathsf{Pf}_{\mathsf{Dec}i})$.

$\mathsf{DHPVSS.Rec}(\mathsf{pp}, \{S_i : i \in \mathcal{T}\})$:

1. If $|\mathcal{T}| < t + 1$, output $\perp$.
2. Otherwise select $\mathcal{T}' \subseteq \mathcal{T}$, with $|\mathcal{T}'| = t + 1$ (e.g. the first $t + k$ indices in $\mathcal{T}$).
3. Define $S' = \sum_{i\in\mathcal{T}'} S_i^{L_i(\beta)}$ where $L_i(X) = \mathrm{Lag}_{i,\mathcal{T}',\{\alpha_i : i\in\mathcal{T}'\}}$.
4. Output S'.

$\mathsf{DHPVSS.VerifyDec}(\mathsf{pp}, C_i, S_i, \mathsf{Pf}_{\mathsf{Dec}i})$: Parse $C_i = (R, B_i)$. Compute $M_i = B_i S_i^{-1}$, verify $\mathsf{Pf}_{\mathsf{Dec}i}$ is a valid proof of discrete log equality for the statement by running $\texttt{NIZK}_{DLEQ}.\mathsf{Verify}((g, R, \mathsf{pk}_i, M_i), \mathsf{Pf}_{\mathsf{Dec}i})$, outputting the result of the verification.