Non-interactive VSS using Class Groups and Application to DKG

Aniket Kate*†, Easwar Vivek Mangipudi†, Pratyay Mukherjee†,
Hamza Saleem†, Sri Aravinda Krishnan Thyagarajan‡,

* *Purdue University* † *Supra Research* ‡ *NTT Research*

***Abstract*—We put forward the first non-interactive verifiable secret sharing scheme (NI-VSS) using classgroups – we call it cgVSS. Our construction follows the standard framework of encrypting the shares to a set of recipients and generating a non-interactive *proof of correct sharing*. However, as opposed to prior works, such as Groth's [Eprint 2021], or Gentry et al.'s [Eurocrypt 2022], we do not require any range proof - this is possible due to the unique structure of class groups, that enables efficient encryption/decryption of large field elements in the exponent of an ElGamal-style encryption scheme. Importantly, this is possible without destroying the additive homomorphic structures, which is required to make the proof-of-correctness highly efficient. This approach not only substantially simplifies the scheme, but also outperforms the state-of-art schemes significantly. Our implementation shows that cgVSS outperforms (a simplified implementation of) Groth's protocol in overall communication complexity by 5.6x and about $2.4 - 2.7$x in computation time per node (for a 150 node system).**

Additionally, we formalize the notion of public verifiability, which enables anyone, possibly outside the participants, to verify the correctness of the dealing. In fact, we re-interpret the notion of public verifiability and extend it to the setting when all recipients may be corrupt and yet can not defy public verifiability – to distinguish with state-of-art we call this *strong* public verifiability. Our formalization uses the universal composability framework.

Finally, through a generic transformation, similar to Groth's [Eprint 2021], we obtain a NI-DKG scheme for threshold systems, where the secret key is the discrete log of the public key. Our security analysis in the VSS-hybrid model uses a formalization that also considers a (strong) public verifiability notion for DKG, even when more than threshold parties are corrupt. Instantiating with cgVSS we obtain the first NI-DKG scheme from class groups – we call it cgDKG.

1. Introduction

In a threshold secret sharing scheme [1], [2], a dealer distributes a secret among a set of n parties in such a way that the secret can only be reconstructed if a subset of $t + 1$ or more parties contribute their shares. A potential concern arises when a malicious dealer distributes shares in a manner that enables two different subsets of $t + 1$ or more parties to reconstruct two different secret values. A verifiable secret sharing (VSS) scheme [3] addresses this concern and enhances security by ensuring that each party receives a share and proof that their share is a valid part of the secret. This crucial feature allows parties to confirm the validity of their shares without needing to reconstruct the actual secret, rendering VSS highly valuable for secure distributed computing (SDC) applications such as randomness beacon [4], [5], [6], distributed key generation (DKG) [7], [8], [9], [10] for threshold cryptography [11], [12], [13], [14], and multiparty computation [15], [16], [17], [18].

Over the past decade, the increasing prominence of blockchains, cryptocurrencies, and the emergence of decentralized finance (DeFi) has sparked substantial practical interest in VSS and its various SDC applications. These applications encompass but are not limited to threshold signatures for wallet security [19], blockchain consensus certifcates [20], distributed randomness services [21], [22], [23], as well as generic secure multi-party computation [24]. Like blockchain ledgers, blockchain-based applications rely heavily on demonstrating the system's correctness to *any interested party*, possibly outside the system. Consequently, these applications require the employed SDC solutions to be publicly verifiable [25], [26]. In particular, the protocol execution transcript should be convincing evidence to anyone that the system output is correct, even when all parties are malicious.[1] Considering that an interested verifier might arise after the protocol execution concludes, the verification procedure should be non-interactive and transferable, allowing it to convince an unlimited number of verifiers.

Until recently, the SDC literature has mostly focused on unconditionally hiding VSS protocols [7], [27], [28], [29], [30], [31], [32] as they can offer the best possible secrecy guarantee using secure and authenticated channels between the dealer and each party while being efficient as compared to perfectly secure (or unconditional) VSS schemes [33] due to their otherwise computational nature. However, any communication over secure and authenticated channels is not publicly verifiable [25] so are these VSS

1. Note that when all parties are malicious, no privacy or robustness (a.k.a. guaranteed termination) properties can be guaranteed. Public verifiability solely focuses on the correctness of the protocol output, if the protocol terminates.

schemes.[2] Moreover, to the best of our knowledge, replacing secure and authenticated channels with public-key encryption does not solve the problem as these schemes continue to be incorrect when the number of malicious parties exceeds the threshold t. Nevertheless the notion of publicly verifiable secret sharing (PVSS) already exists [3], [5], [8], [25], [36], [37], [38], [39], [40] assuming a public-key infrastructure (PKI) is in place and a non-interactive zero-knowledge (NIZK) proof of correct sharing is feasible. Furthermore, these protocols require participants to speak only once, if a PKI is in place (the PKI can be used unlimited times) and are also termed as non-interactive VSS (NI-VSS) – this is an important feature that often comes handy in permissioned blockchain ecosystems, especially when synchronization among the participants is a problem. Henceforth, whenever we refer to NI-VSS we mean non-interactive VSS (in the PKI setup) with public verifiability.

Recent works on NI-VSS [8], [40], developed in the blockchain context, follow a standard template to construct a NI-VSS: the dealer, with a secret s creates a share s_i for party P_i and then broadcasts a multi-receiver encryption vector of all s_i's encrypted with corresponding public keys, in that each s_i is encrypted under pk_i and so on. Every party P_i can decrypt their own s_i with sk_i, but nothing else. To enable (public) verifiability, the dealer additionally provides proof of correct encryption with respect to the commitments of shares. Notably, if the encryption scheme is additively homomorphic, then the proof of correctness can be made specialized and thus more efficient by exploiting the homomorphic structure. In particular, Gentry, Halevi, and Lyubashevsky [40] use a variant of Regev's lattice-based encryption, whereas Groth [8] uses exponentiated ElGamal encryption. While the lattice-based approach is asymptotically beneficial in terms of computation complexity, they additionally needed to employ range proof systems such as Bulletproofs [41], thereby incurring significant performance overhead and design complexity. Instead, Groth [8] uses exponentiated ElGamal encryption over cyclic groups where discrete log is hard; however, since the plaintexts have to be small to facilitate efficient decryption (because discrete log is hard), a so-called "chunking technique", in that a standard-sized (say, 256 bit) plaintext is split into small chunks (say, 16 bits each), is used.[3] However, this also requires the dealer to prove that the chunking is done correctly. To resolve this, a novel Schnorr-style Fiat-Shamir based NIZK proof technique, called *proof-of-correct-chunking* was employed by Groth [8]. While this makes their protocol more efficient compared to [40], it comes at the cost of rendering the final design more communication heavy and substantially more complex. For example, if we use a simplified variant (that is witout forward secrecy) of Groth's protocol [8] for a publicly verfiable DKG with 200 parties, as much as 441MB data (cf. Fig 7) needs to be communicated over the broadcast channel (or posted on the bulletin board/ledger) in total.

1.1. Our Contribution

A New and Simple NI-VSS. We propose a new NI-VSS scheme (in Section 5), which follows the same "encrypt-and-prove" paradigm as above, but completely avoids any range-proof and "chunking" of the secret key. In particular, we use a class group-based additively homomorphic encryption scheme [42], which is structurally similar to ElGamal, but supports encryptions of large plaintexts in the exponent. Specifically, the class group-based encryption puts the plaintext in the exponent of a sub-group where the discrete log is easy, thus enabling efficient decryption – security is based on *existing class group-based assumptions*. Usage of a class group in the above template not only *significantly simplifies* the design, but also makes *considerable gain in the performance* compared to the state-of-art (a simplified version Groth's NI-VSS [8]) as evident by our implementations (cf. Section 7). Also, since deploying our NI-VSS scheme requires a PKI setup for class-group encryptions, we show in Section 4.1 how to realize that using NIZK proofs of the argument of knowledge over class groups. Our NIZK proofs are adapted from prior works such as [43], but supports a stronger knowledge extraction requirement – for this we provided a modified analysis in Appendix A.

Generic Transformation to NI-DKG. We then propose a generic efficiency-preserving transformation of our NI-VSS scheme to a NI-DKG protocol (in Section 6). Instantiating with cgVSS we obtain a simple and efficient class group-based DKG protocol for key generation in the discrete log (DLog)-based threshold settings (that supports popular threshold signatures such as BLS, Schnorr etc.). The resulting DKG protocol is non-interactive (NI-DKG) as well as publicly verifiable. However, we remark that due to the usage of DLog-based commitments our new NI-DKG protocol is susceptible to the so-called biasing public-key attack [7]. Nevertheless, as argued in [7], [18], [44] this suffices for many DLog based applications such as threshold Schnorr signature, BLS etc. We also note that it appears that this can be fixed by using the same technique proposed by Gennaro et al. [7], where they use a perfectly hiding commitment (such as Pederson's) instead of the DLog-based commitments. But this comes at the cost of more rounds of interactions. The (im)possibility of NI-DKG without the biasing public-key attack is an interesting and major open question.

New Definitions with Public Verifiability. Additionally we propose a new UC-based formal definition for NI-VSS (cf. Section 5) that takes our stronger notion of public verifiability into account. We formally prove that our construction cgVSS securely realizes our ideal functionality

2. We consider public verifiability against up to n corruptions here. As we discuss later, a weaker version of public verifiability has been considered in the recent literature [34], [35] that holds only when the adversary can compromise up to t parties. To distinguish, we call our notion *strong* public verifiability. Unless otherwise mentioned, by public verifiability we will be referring to this stronger notion throughout this paper.

3. We note that other ways to encrypt large messages, such as hashed ElGamal, do not work as they lack the additive homomorphism structure for the specialized proof of correct sharing to work.

$\mathcal{F}_{\mathsf{VSS}}$ (cf. Theorem 4). Our generic transformation from NI-VSS to NI-DKG (cf. Section 6) is provided in $\mathcal{F}_{\mathsf{VSS}}$-hybrid, and is shown to securely realize our DKG functionality $\mathcal{F}_{\mathsf{DKG}}$, which also is equipped with (strong) public verifiability.[4] Our NI-DKG definition differs substantially from existing ones [47], [48], [49] because of two reasons. Firstly, it is specially equipped to handle public verifiability as mentioned above. Secondly, our definition is weakened to account for public-key biasing, whereas prior definitions do not allow that.[5] We also remark that, (variants of) the prior NI-VSS schemes, namely the works such as Gentry et al. [40] and Groth [8], plausibly satisfy our definitions too. We do not investigate that formally.

Benchmarking. Finally, we implement our NI-VSS protocol cgVSS and compare that with the closest existing scheme by Groth's [8] in terms of dealer/receiver times, and the total bit-length of the message broadcast by the dealer (in Section 7). For comparison, we implement a simplified version of the VSS mechanism (referred to as GrothVSS henceforth) proposed by Groth [8] without forward secrecy. Our implementation shows that the bit-length of the total broadcast message for a single execution for 150 users is 296.51 Kb for the cgVSS compared to 1.66 Mb in GrothVSS which is a 5.6x improvement. Also, in the same setting the gain in dealer's/receiver's computation time is about $2.4 - 2.7$x. In summary, our protocol cgVSS outperforms the state-of-art GrothVSS both in communication and computation. This is despite the class-group operations being in the regime of other similar composite order groups, such as RSA. Essentially the performance gain can be attributed for the design simplification, in that any range proof (or proof-of-chunking ala Groth [8]) is totally dispensed with.[6] Importantly, this means that our scheme cgVSS scales much better with the increasing number of parties compared to GrothVSS. We also benchmark the DKG protocols (cf. Sec. 7) end-to-end and compare GrothDKG with cgDKG; for a 50 node network, GrothDKG takes 69.7sec and cgDKGtakes 47.9sec.

1.2. Related Work and Discussion

PVSS. In their seminal paper, Chor et al. [3] introduced the notion of verifiable secret sharing (VSS). Stadler [36] first proposed publicly verifiable VSS (PVSS) and two constructions using verifiable ElGamal encryption. A long line of works [25], [51], [52] , [5], [8], [36], [38], [39], [53], [54], [55], [56], [57] realized publicly verifiable and non-interactive VSS schemes. They typically employ encryption mechanisms, including Paillier [38], [39], [57], ElGamal-in-the-exponent [8], pairing [58], [59] and lattice-based encryptions [40]. The schemes, that use Paillier encryption, suffer from long exponentiations and proof size, and one that uses ElGamal in the exponent [8] requires small exponents due to the hardness of DLog. The schemes involving pairing generate shares are group elements (not scalars) and are not suitable for settings such as threshold signatures. PVSS schemes based on lattice-encryption schemes [40] are indeed asymptotically efficient, albeit require large public keys and ciphertext sizes.

Different notions of public verifiability. While the concept of public verifiability has been around for long time, we observe that there is a lack of formalization, which resulted into different interpretations. In particular, for all non-interactive protocols the public verifiability holds even if more than t recipients (possibly everyone) are corrupt. A motivation for this strong notion is the electronic voting scenario, for which the concept of PVSS was originally developed. In particular, a correct ballot cast by a voter via PVSS must be *self-verifiable*, that is the verifiability must not depend on any other participants including the voting servers. Strong public verifiability provides the exact guarantee that a voter's ballot cannot be falsely discarded or manipulated to a different vote even when all voting servers (i.e., VSS recipients) are compromised. For VSS schemes that are not publicly verifiable such as [28], it is possible for a majority of servers to force the voter to reveal her ballot on the broadcast channel with false complaints, and the voter would have no way to prove the legitimacy of her vote in that case.

Very recently a weaker version of public verifiability has been considered [34], [35] that holds only against an adversary which can compromise up to t recipients. This notion, while falling short of providing guarantee in settings similar to above such as voting, can be interesting in the blockchain settings over the internet. Nevertheless, in this paper we focus on the traditional notion of public verifiability, re-interpret and formally capture through our UC-based definitions. To distinguish with the weaker variant (a la [34], [35]) we call it strong public verifiability in this work.

DKG. Several DKG protocols to support DLog-based threshold systems have been studied [7], [8], [9], [50], [50], [60], [61], [62], [63] in the literature in the synchronous and asynchronous settings. However, to achieve public verifiablity, the nodes need to perform PVSS (instead of VSS or asynchronous VSS). Any aggregatable PVSS scheme [9] which supports homomorphic operations on the secret shares [8], [38], [39] may be employed to realize a publicly verifiable DKG mechanism. To achieve non-interactive DKG, one should employ a PVSS for the secret sharing. Groth [8] proposed a non-interactive distributed key generation mechanism using ElGamal encryptions of shares that can be publicly verified by all the parties from the commitments of the polynomial coefficients. We use a simplified variant of this scheme as our baseline.

Biasing the public key. Recently, Katz [50] proposed two round-optimal constructions for a 'robust' DKG mechanism, where they define *robustness* as guaranteed-output-delivery.

4. Note that, this is, in spirit, somewhat similar to the concept of publicly auditable MPC [45], [46], that allows public verification of transcripts of an MPC protocol even when all parties are corrupt.

5. A recent simulation-based definition put forward by Katz [50] also formalizes this, but in a different manner.

6. Moreover, the simplified design itself is a substantial advantage from engineering/deployment perspective as well.

Their definition requires that the DKG mechanism outputs an *unbiased* public key. However, unbiased public keys are not an absolute requirement for DKG mechanisms, since it has been shown that biased public keys can be securely employed for certain systems as long as the secret key is secure. Gennaro et al. [7] show that biased public keys can be securely employed for any cryptographic system relying on the DLog assumption, like the threshold version of the Schnorr signature scheme. Bacho and Loss [44] show that DKG mechanisms that output biased public keys can be employed for generating key shares of adaptively secure BLS scheme as long as they can be shown to be *oracle-aided-algebraic-simulatable* (see [44, Sections 3,4.3]. Braun, Damgård, and Orlandi [18] propose an encryption scheme based on class groups that is secure even with biased public keys. In this work, we explicitly define the functionality (see Figure 5) to allow the adversary to bias the public key. This allows us to achieve an efficient non-interactive DKG protocol. We show (see Appendix E) that our definition is oracle-aided-simulatable as defined in [44] and hence can be employed for BLS [64], making it suitable for DLog-based systems and signature schemes like BLS.

Strong Public Verifiability of DKG. Our NI-VSS to NI-DKG transformation carries over the strong public verifiability. However, we need to interpret what it actually means in the context of DKG. First we note that if more than t parties in a DKG protocol are compromised, we cannot guarantee the confidentiality of the shared secret/private key: the adversary can simply interpolate its shares to compute the secret. As a result for applications such as threshold signing, the adversary can easily sign any message in this case. However, we notice that strong public verifiability is still meaningful for applications such as distributed verifiable randomness services [21], [65], [66] (DVRF).[7] In this case, even if the secret-key is known to the adversary, and as a consequence the DVRF output is predictable, yet the adversary can not deviate from computing a correct value – this is guaranteed by the VRF definition itself. Intuitively, this means that the output of the VRF can not biased to a specific value, desired by the attacker. So, in other words, loss of unpredictability does not immediately means a loss of so-called "unbiasability". Now, in the strong public verifiability setting, we are pretty much in the same scenario, where more than t parties are compromised, and the adversary knows the secret. It turns out, public verifiability can indeed guarantee an unbiasability even in this case, as long as there is at least one honest party. We capture this in our UC functionality $\mathcal{F}_{\mathsf{DKG}}$ in Section 6. However, to prove that our generic NI-DKG construction (in $\mathcal{F}_{\mathsf{VSS}}$-hybrid) achieves this, we need to make another assumption, that is the adversary is non-rusing. Othewrise a rushing adversary would know the dealing of the honest parties before committing its own dealings, and can actually set the final secret to an arbitrary value of her choice (for example 0) – rendering the guarantee useless in practice. However, a non-rushing adversary has to commit the corrupt party's dealing ahead of time, and thereby can not execute this attack. It is worth noting that, as long as up to t parties are corrupt, our protocol continues to securely realizes the $\mathcal{F}_{\mathsf{DKG}}$ functionality against rushing adversary. In a nutshell, we obtain a degraded, yet meaningful security guarantees (perhaps the best possible) beyond t corruption, while keeping the *full* security guarantee up to t corruption. This is formalized in Theorem 6.

7. Specifically, when a DKG protocol is deployed to support a DVRF protocol.

2. Preliminaries

2.1. Notation

We use $\mathbb{Z}$ for all integers, and $\mathbb{N}$ for all natural numbers $\{1, 2 \ldots\}$. Vectors are denotes as $\vec{v}$, and it's i-th element by v_i. For a vector (or ordered set) $\vec{v}_i$, its j-th element is denoted $\vec{v}_{i,j}$. For a (possibly ordered) set of values $v_1, \ldots, v_n$ and a subset $S \subseteq [n]$, we write $\{v_i\}_{i \in S}$ to denote the values $\{v_i\}_{i \in S}$. We use the notation $x \xleftarrow{\$} \mathcal{D}$ to indicate that x is randomly sampled from the distribution $\mathcal{D}$ and the notation $h \leftarrow y$ to indicate that the h has been assigned the value y. Also, for any (possibly randomized) algorithm A we denote $y \leftarrow A(x)$ to express that A on input x yields the output y. Sometimes we denote $A^B(\cdot)$ to denote that the algorithm A has oracle access to another algorithm B. Unless explicitly mentioned otherwise, all algorithms (including the adversary) considered in this paper are probabilistic polynomial time (PPT). Sometimes, we explicitly use the notation $A(x; r)$ to determinize A when run on input x and fixed randomness r. In a multiparty system, we say an adversary is k-bounded if it may corrupt upto k parties. We indicate the set $\{1, 2, \cdots, n\}$ by $[n]$. We use the symbol $\stackrel{?}{=}$ to indicate a check of equality of the left and right-hand side entities of the symbol. $(a \stackrel{?}{=} b)$ returns a boolean value denoting whether the equality holds or not. The computational security parameter is denoted by λ (a typical value 256), and the statistical security parameter is denoted by λ_{st} (typical value 40). To denote that a value x is polynomial in λ, we write $x \in \mathsf{poly}(\lambda)$; similarly for exponential values, we write $x \in 2^{O(\mathsf{poly}(\lambda))}$. We say that a function is negligible in λ, if it vanishes faster than $1/\mathsf{poly}(\lambda)$ for any polynomial poly.

Also, note that for our chosen values, we have $2^{-\lambda_{\mathsf{st}}} = O(\mathsf{negl}(\lambda))$.

2.2. Shamir Secret Sharing

We use Shamir's secret sharing [1]. In a typical Shamir's secret sharing, a field element $s \in \mathbb{Z}_q$ can be shared in a t out of n fashion by choosing a t-degree uniform random polynomial $P(x) \xleftarrow{\$} \mathbb{Z}_q[x]^t$ with constraint $P(0) = s$. The i-th share is computed as $s_i \leftarrow P(i)$. To reconstruct one may use Lagrange coefficients L_is as $s = \sum_{i=1}^{t+1} L_i s_i$. Due to linearity, this can be performed in the exponent without computing s. We denote this by $\mathsf{Shamir}_{n,t,q}(s) = (s_1, \ldots, s_n)$.

2.3. DLog Commitments

We will be using discrete log (DLog) commitments, that are defined over any cyclic group $\bar{G}$ of prime order q. A commitment of a value $x \in \mathbb{Z}_q$ is simply defined to be $\bar{g}^x$, where $\bar{g}$ is a generator of $\bar{G}$. Note that, the commitment scheme does not guarantee hiding, but provides computational binding, as long as the discrete log is hard over $\bar{G}$. A commitment of s is generally denoted by $\mathsf{cmt}(s)$.

2.4. Definition of NIZK

Let $\mathfrak{R}$ be an efficiently computable binary NP relation. For any pair $(\mathsf{inst}, \mathsf{wit}) \in \mathfrak{R}$, we refer to inst as the instance and wit as the witness. If it is computationally hard (in the average case) to determine a witness from a statement, then the relation is called a *hard relation*. For any hard relation $\mathfrak{R}$ we define *NIZK arguments of knowledge (resp. NIZK proof)* in the random oracle model.

Definition 1 (Non-interactive Zero-knowledge Argument of knowledge (resp. Proof) in ROM)**.** Let pp be some public parameters that include a computational security parameter λ, and a statistical security parameter λ_{st}, generated in a setup, and available to all algorithms. Let H be a hash function with an appropriate domain/range, modeled as a random oracle. A secure NIZK for a binary hard relation $\mathfrak{R}$ consists of two PPT algorithms Prove and Verify with oracle access to H defined as follows:

- $\mathsf{Prove}^H(\mathsf{inst}, \mathsf{wit})$. The algorithm takes as input an instance-witness pair and outputs a proof π if $(\mathsf{inst}, \mathsf{wit}) \in \mathfrak{R}$ and $\perp$ otherwise.
- $\mathsf{Verify}^H(\mathsf{inst}, \pi)$. The algorithm takes as input an instance inst and a candidate proof π, and outputs a bit $b \in \{0,1\}$ denoting acceptance or rejection.

We call a ROM-based NIZK scheme a *secure argument of knowledge (resp. proof)* if the algorithms satisfy *perfect completeness*, *statistical zero-knowledge in ROM* and *argument of knowledge (resp. statistical soundness in ROM)*, defined as follows:

- **Perfect completeness**: For any $(\mathsf{inst}, \mathsf{wit}) \in \mathfrak{R}$,
$$\Pr\left[\mathsf{Verify}^H(\mathsf{inst}, \pi) = 1 \mid \pi \leftarrow \mathsf{Prove}^H(\mathsf{inst}, \mathsf{wit})\right] = 1.$$
- **Statistical Zero-knowledge (in ROM)**: There must exist a PPT simulator $\mathcal{S}$ such that for any $(\mathsf{inst}, \mathsf{wit}) \in \mathfrak{R}$ the statistical distance between the following two probability distribution is bounded by a negligible function of λ_{st} as long as an unbounded verifier may ask a bounded (depends on $\lambda, \lambda_{\mathsf{st}}$) number of queries to the random oracle (simulated by $\mathcal{S}$):
  - Output $(\mathsf{inst}, \pi, Q_H)$ where $\pi \leftarrow \mathsf{Prove}^H(\mathsf{inst}, \mathsf{wit})$;
  - Output $(\mathsf{inst}, \pi, Q_H)$ where $\pi \leftarrow \mathcal{S}'(\mathsf{inst})$

  where $\mathcal{S}'$ returns a simulated proof $\pi \leftarrow \mathcal{S}(\mathsf{inst})$ on input $(\mathsf{inst}, \mathsf{wit})$ if $(\mathsf{inst}, \mathsf{wit}) \in \mathfrak{R}$ and $\perp$ otherwise and Q_H denotes the random oracle query-answer pairs made by the verifier;
- **Argument of knowledge**: For all PPT adversary $\mathcal{A}^H$, there exists a PPT extractor $\mathcal{E}^{\mathcal{A}}$ such that
$$\Pr\Big[(\mathsf{inst}, \mathsf{wit}) \notin \mathfrak{R} \text{ and } \mathsf{Verify}^H(\mathsf{inst}, \pi) = 1 \mid (\mathsf{inst}, \pi) \leftarrow \mathcal{A}^H(1^\lambda); \mathsf{wit} \leftarrow \mathcal{E}^{\mathcal{A}}(\mathsf{inst}, \pi)\Big] \leq \mathsf{negl}(\lambda)$$
  for some negligible function negl, where $\mathcal{A}$'s RO queries to H are simulated by the extractor.
- **Statistical Soundness (in ROM).** For any unbounded adversary $\mathcal{A}^H$, that may ask a bounded numner of RO queries to H we have that:
$$\Pr[1 \leftarrow \mathsf{Verify}^H(\mathsf{inst}, \pi) \wedge \mathsf{inst} \notin \mathfrak{R} \mid (\mathsf{inst}, \pi) \leftarrow \mathcal{A}^H(pp_{\mathsf{PoC}})] \leq \mathsf{negl}(\lambda_{\mathsf{st}})$$

Note that, we necessarily rely on unbounded adversaries making a bounded number of RO queries. This number, however, may be sub-exponential in $\lambda, \lambda_{\mathsf{st}}$. .

2.5. Class Groups

Castagnos and Laguillaumie [42] propose an ElGamal-like encryption scheme using class groups. The main idea is to use a composite order group of unknown order with an underlying subgroup of known order where the discrete logarithm is easy. Since then, a number of works showed the feasibility of several cryptographic tasks [18], [43], [67], [68] including two-party ECDSA [68], multi-party computation [18] etc.

In this paper, we follow a presentation similar to [18]. We consider a finite abelian group $\widehat{G}$ of *unknown* order $q \cdot \hat{s}$ with an unknown (and hard to compute) $\hat{s}$, and known q such that q and $\hat{s}$ are co-prime; $\widehat{G}$ is factored as $\widehat{G} \simeq \widehat{G}^q \times F$, where $F = \langle f \rangle$ is the unique subgroup of order q. An upper bound $\bar{s}$ is known for $\hat{s}$. We also consider a cyclic subgroup $G = \langle g \rangle$ of $\widehat{G}$, such that G has order $q \cdot s$ and s divides $\widehat{s}$ – hence q and s are also co-prime. Both s, s' are odd and all s, s', q are exponential in λ. Unlike $\widehat{G}$, the elements of G are not efficiently recognizable. $G^q = \langle g_q \rangle$ denotes the cyclic subgroup of G of the q-th power. So, G can be factored as $G \simeq G^q \times F$ and $g = g_q \cdot f$. We also consider two distributions $\mathcal{D}$ and $\mathcal{D}_q$ over $\mathbb{Z}$ $\{g^x \mid x \leftarrow \mathcal{D}\}$ and $\{g_q^x \mid x \leftarrow \mathcal{D}_q\}$, such that they induce distributions over G and G^q respectively, that are statistically close (within distance $2^{-\lambda_{\mathsf{st}}}$) to uniform distributions over respective domains.

The framework specifies polynomial time algorithms $(\mathsf{CG.ParamGen}, \mathsf{CG.Solve})$ with the following description:

- $(q, \lambda, \lambda_{\mathsf{st}}, \bar{s}, f, g_q, \widehat{G}, F, \mathcal{D}, \mathcal{D}_q; \rho) \leftarrow \mathsf{CG.ParamGen}(1^\lambda, 1^{\lambda_{\mathsf{st}}}, q)$. This algorithm, on input the computational security parameter λ, the statistical security parameter λ_{st} and a modulus q, outputs the group parameters and the randomness ρ used to generate them. For convenience, we include the descriptions of the distributions $\mathcal{D}$ and $\mathcal{D}_q$ as well.
- $x \leftarrow \mathsf{CG.Solve}(f^x, (q, \lambda, \lambda_{\mathsf{st}}, \bar{s}, f, g_q, \widehat{G}, F, \mathcal{D}, \mathcal{D}_q))$. This algorithm deterministically solves the discrete log in group F.

Hardness assumptions on class groups. We formally recall some of the computational hardness assumptions we require for proving the security of our scheme. All assumptions below use a common setup: for the security parameters $\lambda, \lambda_{\mathsf{st}} \in \mathbb{N}$, modulus $q \in \mathbb{Z}$ consider a set of public parameters $pp_{\mathsf{CG}} := (q, \lambda, \lambda_{\mathsf{st}}, \bar{s}, f, g_q, \widehat{G}, F, \mathcal{D}, \mathcal{D}_q; \rho) \leftarrow \mathsf{CG.ParamGen}(1^\lambda, 1^{\lambda_{\mathsf{st}}}, q)$ generated using a uniformly random ρ.

Definition 2 (q-Hard subgroup membership assumption [69]). Sample $x \xleftarrow{\$} \mathcal{D}_q$ and $u \xleftarrow{\$} \mathbb{Z}_q$. Sample a bit $b \xleftarrow{\$} \{0,1\}$ uniformly at random. If $b = 0$, define $h^* \leftarrow g_q^x$, otherwise if $b = 1$ define $h^* \leftarrow f^u \cdot g_q^x$. Then we say that the hard subgroup membership assumption holds over the classgroup framework, if for any PPT adversary $\mathcal{A}$, the following probability is negligible in λ.

$$\left| \Pr\left[b = b^* \mid b^* \leftarrow \mathcal{A}(pp_{\mathsf{CG}}, h^*)^{\mathsf{CG.Solve}(\cdot)} \right] - \frac{1}{2} \right|$$

Definition 3 (Low order assumption [70]). *Let $\mathcal{B} \in \mathbb{N}$. Then we say that the low order assumption over $\widehat{G}$ holds if for any PPT algorithm $\mathcal{A}$, the following probability is negligible in λ:*

$$\Pr\left[\mu^d = 1 \wedge 1 \neq \mu \in \widehat{G} \wedge 1 < d < \mathcal{B} \mid (\mu, d) \leftarrow \mathcal{A}(pp_{\mathsf{CG}})^{\mathsf{CG.Solve}(\cdot)}\right]$$

Definition 4 (Strong root assumption [70]). *Sample $Y \xleftarrow{\$} \widehat{G}^q$. Then we say that the strong root assumption holds over $\widehat{G}$, if for any PPT algorithm $\mathcal{A}$ and any $k \in \mathbb{Z}$ the following probability is negligible in λ:*

$$\Pr\left[X^e = Y \wedge e \neq 2^k \wedge X \in \widehat{G} \mid (X, e) \leftarrow \mathcal{A}(pp_{\mathsf{CG}}, Y)^{\mathsf{CG.Solve}(\cdot)}\right]$$

2.6. Universal Composability

We follow the Universal Composability Framework [71], in that a real-world multi-party protocol realizes an ideal functionality. Similar to the simplified UC framework [72] we assume the existence of a *default authenticated channel* in the real world. This significantly simplifies our definitions and can easily be removed using an ideal authenticated channel functionality [73].

We consider a *fixed number of parties* in the system and a **static corruption** model, that is, neither the set of participants nor the set of corrupt parties can change during the execution. The corrupt parties can behave in a completely **malicious** manner and may collude with each other. When we say that a real world protocol *securely realizes* an ideal world functionality, then we mean that for all PPT adversary in the real world, there is an ideal world adversary (simulator) such that no PPT environment can distinguish between the ideal world and the real world's output that consists of the inputs to the parties, the outputs of the honest parties and the output of the adversary. For a formal presentation we refer to [71], [72]

3. Building Blocks

Our NI-VSS scheme is based on three building blocks over the class groups: (i) a NIZK proof for knowledge of exponent; (ii) a multi-receiver encryption scheme; (iii) and a non-interactive sigma protocol that ensures compact proof of correct secret-sharing. Next, we present them in order.

3.1. NIZK for Knowledge of exponent

Now we present our NIZK construction for knowledge of exponents over class groups. We use a simpler variant of different sigma protocols used in prior works [43], [70], [74]. Similarly to those, we show that NIZK proof system is a secure argument of knowledge (Def. 1) from two new assumptions, hardness of finding low-order elements (Def. 3), and hardness of finding a root (Def. 4) over group $\widehat{G}$. Below we describe the construction.

Consider the class group parameters $pp_{\mathsf{CG}} = (q, \lambda, \lambda_{\mathsf{st}}, \bar{s}, f, g_q, \widehat{G}, F, \mathcal{D}, \mathcal{D}_q; \rho)$ generated using $\mathsf{CG.ParamGen}(1^\lambda, 1^\lambda_{\mathsf{st}}, q)$, an instance $\mathsf{inst} = (g_q, h) \in G^q \times G^q$ and witness $\mathsf{wit} = k \xleftarrow{\$} \mathcal{D}_q$ such that $h \leftarrow g_q^k \in G^q$. Also, consider a hash function H (modeled as random oracle) which maps to a range $[\mathcal{B}]$ for an integer $\mathcal{B} = 2^\lambda$. The set of public parameters for the proof system is defined as $pp_{\mathsf{Kex}} \leftarrow (H, \mathcal{B}) \cup \{pp_{\mathsf{CG}}\}$. Then the proof system consists of the following two algorithms (for simplicity we keep the RO notation implicit):

- $\mathsf{Kex.Prove}(pp_{\mathsf{Kex}}, \mathsf{inst}, \mathsf{wit}) \rightarrow \pi$. This randomized algorithm takes an instance-witness pair $(\mathsf{inst}, \mathsf{wit}) = ((g_q, h), k)$ as input. Then it executes the following steps:
  - Samples an integer $r \xleftarrow{\$} [\mathcal{B} \cdot |\mathcal{D}_q| \cdot 2^{\lambda_{\mathsf{st}}}]$
  - $a \leftarrow g_q^r$;
  - $c \leftarrow H(g_q, h, a) \in \mathcal{B}$;
  - $s \leftarrow r + kc \in \mathbb{Z}$;
  - Output the NIZK proof $\pi = (c, s)$
- $\mathsf{Kex.Ver}(pp_{\mathsf{Kex}}, \mathsf{inst}, \pi) \rightarrow 1/0$. This deterministic algorithm takes an instance $\mathsf{inst} = (g_q, h)$ and a candidate proof $\pi = (c, s)$ as input. Then:
  - Check if $s \leq (2^{\lambda_{\mathsf{st}}} + 1) \cdot \mathcal{B} \cdot |\mathcal{D}_q|$;
  - If the above check fails output 0 and stop. Else compute $a \leftarrow g_q^s \cdot (h^c)^{-1}$;
  - Output $(c \stackrel{?}{=} H(g_q, h, a)) \in \{0, 1\}$.

Security. As detailed in Definition 1, a NIZK proof system is called *secure argument of knowledge* if it satisfies *completeness*, *statistical zero-knowledge* and *argument of knowledge*. Completeness follows immediately. The statistical zero-knowledge argument is analogous to Schnorr's proof over cyclic groups, except that now the simulator needs to sample s carefully to match the range. Since we compute it over an integer as the group order is unknown, we need to ensure that the value s can be simulated without the knowledge of k. For that, we rely on a statistical argument. In particular, we choose a "mask" r randomly from a range, which is larger than the range of kc by a factor of $2^{\lambda_{\mathsf{st}}}$. So, to simulate, it is possible to sample s from a range such that the simulated value is within statistical distance $2^{-\lambda_{\mathsf{st}}}$ to the actual value. The argument of knowledge is more intricate, and uses two more assumptions over class groups – this can be done by carefully adjusting analysis from prior works [43], [70], [74]. The main difference from Schnorr's proof is again that due to unknown order s is an integer. Nevertheless, using the class group structure we can ensure that unless the witness k is extracted, one of the low-order or strong root assumptions is broken. Formally we prove the following theorem.

Theorem 1. *For any $\lambda, \lambda_{\mathsf{st}} \in \mathbb{N}$ and any modulus $q \in \mathbb{Z}$, for a correctly generated class group parameters $pp_{\mathsf{CG}} \leftarrow \mathsf{CGE.KeyGen}(1^\lambda, 1^\lambda_{\mathsf{st}}, q)$ as long as the low order assumptions (Def. 3) and the strong root assumption (Def. 4) holds over the class group $\widehat{G}$, the NIZK proof system described above is secure argument of knowledge in the random oracle model.*

We defer the full proof to Appendix A.

3.2. Multi-receiver Encryption from Class groups

We first provide a definition of multi-receiver encryption by simply extending from prior notions [75], [76] where the adversary can corrupt t out of n parties and possibly know their secrets too.

Definition 5 (Multi-receiver Encryption)**.** Let $n, t, \in \mathbb{N}$ such that $n > t$. Let pp be a set of public parameters. A multi-receiver encryption scheme consists of three algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax:

- $\mathsf{KeyGen}(pp)$. The algorithm takes a set of system parameters and return a pair of keys (sk, pk).
- $\mathsf{mrEnc}(pp, \vec{pk}, \vec{m})$. The algorithm takes a vector of messages and a vector of public keys to generate a vector of ciphertext of the form $(R, \vec{E})$, with the common randomness-dependent part R and message-dependent (and key-dependent) individual parts $E_1, \ldots, E_n$.
- $\mathsf{Dec}(pp, sk, (R, E))$. The algorithm takes a specific secret key sk and the corresponding ciphertext (R, E) to output a message m.

Security. Before describing the security definition, first let us define an *admissible* adversary, which chooses a corrupt set $C \subseteq [n]$, for which it generates keys by *correctly* running $(sk_i, pk_i) \leftarrow \mathsf{KeyGen}_{pp}$ for all $i \in C$. Looking ahead, this assumption is removed in the PKI setup (cf. Section 4.1) by making every party producing a NIZK argument of knowledge (cf. Definition 1) of the sk_i for a public key pk_i (in the construction $pk_i = g_q^{sk_i}$, so a knowledge-of-exponent argument suffices).

We call a multi-receiver encryption scheme *secure*, if for any correctly generated pp any $n, t \in \mathbb{N}$ $(n > t)$ and for any *admissible* PPT adversary $\mathcal{A}$ the probability that the following experiment outputs 1 is bounded by at most $\mathsf{negl}(\lambda)$ away from $1/2$:

- Once generated, give pp to $\mathcal{A}$
- From $\mathcal{A}$ receive $C \subset [n]$. Define $t \leftarrow |C|$ and $H \leftarrow [n] \setminus C$.
- For all $i \in H$ run $\mathsf{KeyGen}(pp)$ (each time with fresh randomness) to obtain $\{(sk_i, pk_i)\}_{i \in H}$. Give all $\{pk_i\}_{i \in H}$ to $\mathcal{A}$. Receive $\{pk_i\}_{i \in C}$ from $\mathcal{A}$.
- Receive challenge vectors $(\vec{m}_0, \vec{m}_1)$ of length n from $\mathcal{A}$ such that for all $i \in C : m_{0,i} = m_{1,i}$; if this does not hold then output a random bit and abort.
- Choose a uniform random b and encrypt
$$(R, \{E_i\}_{i \in [n]}) \leftarrow \mathsf{mrEnc}(pp, \{pk_i, m_{b,i}\}_{i \in [n]})$$
.
- Receive b' from $\mathcal{A}$, output $(b \stackrel{?}{=} b')$.

3.2.1. Construction of the Encryption. We present multi-receiver encryption from class groups in this section. This is a simple adaptation of the base scheme from [42].

Let pp_{CG} be the public parameters generated by running $(q, \lambda, \lambda_{\mathsf{st}}, \bar{s}, f, g_q, \widehat{G}, F, \mathcal{D}, \mathcal{D}_q) \leftarrow \mathsf{CG.ParamGen}(1^\lambda, 1^\lambda_{\mathsf{st}}, q)$ for some appropriately chosen $\lambda, \lambda_{\mathsf{st}}, q$. Let $n, t \in \mathbb{N}$ be such that $n > t$. The multi-receiver encryption scheme is comprised of three algorithms $\mathsf{CGE.KeyGen}$, $\mathsf{CGE.mrEnc}$ and $\mathsf{CGE.Dec}$ for generating the keys, (multi-receiver) encryption and decryption, respectively:

- $\mathsf{CGE.KeyGen}(pp_{\mathsf{CG}}) \to (sk, h)$:
  - $sk \xleftarrow{\$} \mathcal{D}_q$
  - $h \leftarrow g_q^{sk}$
- $\mathsf{CGE.mrEnc}(pp_{\mathsf{CG}}, \{h_i, m_i\}_{i \in [n]}) \to (R, \{E_i\}_{i \in [n]})$
  - $r \xleftarrow{\$} \mathcal{D}_q$
  - $R \leftarrow g_q^r$
  - For all $i \in [n]$: $E_i \leftarrow f^{m_i} h_i^r$
- $\mathsf{CGE.Dec}(pp_{\mathsf{CG}}, sk, R, E) \to m$
  - $M \leftarrow \frac{E}{R^{sk}}$
  - $m \leftarrow \mathsf{CG.Solve}(pp_{\mathsf{CG}}, M)$

In this description, we use the notation h for the public key instead of pk. Throughout the paper, we use them interchangeably.

Security. The security argument of single-receiver scheme provided in [42] can be extended easily to the multi-receiver setting. Formally we can prove the following theorem:

Theorem 2. *For any $\lambda, \lambda_{\mathsf{st}}$ and any modulus $q \in \mathbb{Z}$ let $(q, \lambda, \lambda_{\mathsf{st}}, \bar{s}, f, g_q, \widehat{G}, F, \mathcal{D}, \mathcal{D}_q; \rho) \leftarrow \mathsf{CG.ParamGen}(1^\lambda, 1^\lambda_{\mathsf{st}}, q)$ be a set of correctly generated class group parameters. Then, for that set of parameters as long as hard subgroup assumptions (Def. 2) holds, the above multi-receiver encryption scheme is secure according to Definition 5 for any $n, t \in \mathbb{N}$ such that $n > t$.*

We defer the proof to Appendix B.

3.3. Proof of Correct Secret Sharing

Looking ahead, in our NI-VSS protocol we shall require the dealer to produce a NIZK proof of correct sharing, where shares are encrypted with the above multi-receiver encryption. We essentially use the Groth's [8] variant of sigma protocol, adapted to our class group setting. The overall idea, as we recall from [8], is to use Schnorr-like proof for knowledge of exponent in a compact fashion. Note that, the multi-ciphertext consists of a group element $R = g_q^r$ and another n group elements of the form $E_i = f^{s_i} h_i^r$. The dealer is required to prove that the encrypted messages vector forms a legitimate t out of n Shamir's secret sharing. The crux of the idea is to combine these different exponents in a way such that they are consistent with the evaluation of t-degree secret polynomial used for secret-sharing – to enable these DLog commitments of the secret polynomial are used. Let us now describe the scheme in detail.

Consider any cyclic group (typically an elliptic curve) $\langle \bar{g} \rangle = \bar{G}$ of prime order q. Note that $\bar{G}$ is isomorphic to F. Also, consider hash functions (modeled as random oracles in the proof) H, H' both mapping $\rightarrow \mathbb{Z}_q$. The public parameter of the proof system is defined as $pp_{\mathsf{PoC}} := \{\bar{g}, \bar{G}, H, H'\} \cup pp_{\mathsf{CG}}$, where $pp_{\mathsf{CG}} \leftarrow \mathsf{CG.ParamGen}(1^\lambda, 1^\lambda_{\mathsf{st}}, q)$.

Now consider a secret $s \in \mathbb{Z}_q$, and let $(s_1, \ldots, s_n)$ be a t out of n Shamir's secret-sharing of s, which is generated by randomly choosing a t-degree secret polynomial $P(x)$ over $\mathbb{Z}_q$ such that $P(i) = s_i$ for all $i \in [n]$. Also, denote the coefficients of P by $a_0, a_1, \ldots, a_t$ each in $\mathbb{Z}_q$ and corresponding DLog commitments over $\bar{G}$ as $A_0, A_1, \ldots, A_t$ where $A_i = \bar{g}^{a_i}$ for $i \in \{0, \ldots, t\}$. The shares $s_1, \ldots, s_n$ are then encrypted using the multi-receiver encryption scheme described above as $\mathsf{CGE.mrEnc}(pp_{\mathsf{CG}}, \{h_i, s_i\}_{i\in[n]}; r)$ using randomness $r \in \mathcal{D}_q$ (we determinize the encryption algorithm here) to produce a ciphertext tuple $(R, \{E_i\}_{i\in[n]})$. The NIZK proof we describe below proves a hard relation $\mathfrak{R}_{\mathsf{CS}}$ that consists of instances (inst, wit) where each inst and the corresponding witness wit are of the form:

- $\mathsf{inst} = \big(\{h_i\}_{i\in[n]}, (R, \{E_i\}_{i\in[n]}), (A_0, \ldots, A_t)\big)$;
- $\mathsf{wit} = ((s_1, \ldots, s_n), r)$

such that the following holds:

- there exists a t-degree polynomial $P(x) = a_0 + a_1 x + \ldots a_t x^t$ over $\mathbb{Z}_q$ such that for all $i \in [n]$: $s_i = P(i)$; and for all $j \in \{0, \ldots, t\}$: $A_j = \bar{g}^{a_j}$;[8]
- encrypting $s_1, \ldots, s_n$ with randomness r using public keys $h_1, \ldots, h_n$ yields the multi-receiver ciphertext $(R, \{E_i\}_{i\in[n]})$

Our proof of correct sharing consists of two algorithms PoCS.Prove and PoCS.Ver, which are described in Figure 1.

Next we show that our construction (cf. Fig. 1) is a NIZK proof in ROM (as Def. 1) by formally proving the following theorem.

Theorem 3. *For any security parameters $\lambda, \lambda_{\mathsf{st}} \in \mathbb{N}$ and any modulus $q \in \mathbb{N}$, our NIZK construction described in Fig. 1 is a secure proof system (as described in Def. 1) in the random oracle model.*

8. Note that, the coefficients $a_0, a_1, \ldots$ of polynomial P can be computed from the evaluations $s_1, \ldots, s_n$, therefore we do not include the coefficients within the witness separately.

Proof of Correct Sharing

- $\mathsf{PoCS.Prove}(pp_{\mathsf{PoC}}, \mathsf{inst}, \mathsf{wit}) \rightarrow \pi_{\mathsf{CS}}$:
  - Parse wit as $\{(s_1, \ldots, s_n), r\}$.
  - Sample $\alpha, \xleftarrow{\$} \mathbb{Z}_q, \rho \leftarrow [q \cdot |\mathcal{D}_q| \cdot 2^{\lambda_{\mathsf{st}}}]$.
  - $W \leftarrow g_q^\rho$ and $X \leftarrow \bar{g}^\alpha$
  - Compute:
    - $\gamma \leftarrow H(\mathsf{inst})$.
    - $Y \leftarrow f^\alpha \cdot \left(h_1^\gamma \cdot h_2^{\gamma^2} \ldots \cdot h_n^{\gamma^n}\right)^\rho \in G$.
    - $\gamma' \leftarrow H'(\gamma, W, X, Y)$.
    - $z_r \leftarrow r\gamma' + \rho \in \mathbb{Z}$.
    - $z_s \leftarrow \gamma' \sum_{i=1}^n s_i \gamma^i + \alpha \in \mathbb{Z}_q$.
  - Finally return $\pi_{\mathsf{CS}} \leftarrow (W, X, Y, z_r, z_s)$
- $\mathsf{PoCS.Ver}(pp_{\mathsf{PoC}}, \mathsf{inst}, \pi_{\mathsf{CS}}) \rightarrow 1/0$:
  - Parse π_{CS} as (W, X, Y, z_r, z_s).
  - Compute:
    - $\gamma \leftarrow H(\mathsf{inst})$.
    - $\gamma' \leftarrow H'(\gamma, W, X, Y)$.
  - Verify the following equality:
    - $W \cdot R^{\gamma'} \stackrel{?}{=} g_q^{z_r} \in G^q$;
    - $X \cdot (\prod_{j=0}^t A_j^{\sum_{i=1}^n i^k \gamma^j})^{\gamma'} \stackrel{?}{=} \bar{g}^{z_s} \in \bar{G}$;
    - $(\prod_{i=1}^n E_i^{\gamma^i})^{\gamma'} \cdot Y \stackrel{?}{=} f^{z_s} \cdot \prod_{i=1}^n (h_i^{\gamma^i})^{z_r} \in G$.
  - Return 1 if all of the above holds, and 0 otherwise.

Figure 1: Proof System of Correct Sharing.

We provide the proof sketch for completeness, soundness and zero-knowledge in Section C.

4. Our Model

Communication Model. For our non-interactive constructions, similar to all previous NI-VSS and NI-DKG construction schemes (such as [5], [8], [37]), we assume that every party has access to a *broadcast channel*. This is a common assumption for non-interactive publicly verifiable multiparty computation protocols [77], [78], where the adversary controls the communication channel and can delay the messages; however, it has to deliver those before the synchrony communication bound Δ. The adversary is also rushing and can delay the messages of the parties and inject its own messages after observing honest nodes' messages during the current round. Moreover, unlike interactive VSS/DKG constructions [7], [27], [28], we do not need any communication links between parties. Furthermore, we consider *static corruption*, in that the set of corrupt parties is fixed at the beginning of the execution and stays the same until the end.

4.1. PKI Setup for Class-groups

Though non-interactive in the online phase, our NI-VSS and NI-DKG protocols require a PKI setup for class-group encryptions. Once a PKI is successfully established, unbounded number of non-interactive executions can take place. Here we describe how a PKI is established for our multi-receiver encryption scheme (cf. Section 3.2).

Protocol for PKI setup. We realize the PKI for the multi-receiver encryption scheme $(\mathsf{CGE.KeyGen}, \mathsf{CGE.mrEnc}, \mathsf{CGE.Dec})$ with class-group parameters $pp_{\mathsf{CG}} \leftarrow \mathsf{CG.ParamGen}((q, \lambda, \lambda_{\mathsf{st}}, \bar{s}, f, g_q, \widehat{G}, F, \mathcal{D}, \mathcal{D}_q))$ along with the NIZK argument for knowledge of exponent (cf. Sec 3.1), that is for a pair (x, g_q^x), the NIZK argument would produce a proof of knowledge of x given g_q^x. An instantiation is provided in Section 3.1.

Suppose that the NIZK has algorithms $(\mathsf{Kex.Prove}, \mathsf{Kex.Ver})$ and public parameters pp_{Kex}, which is consistent with pp_{CG}. Then we describe a protocol Π^{pp}_{PKI} where $pp = \{pp_{\mathsf{CG}} \cup pp_{\mathsf{Kex}}\}$ as follows:

- Each party P_i executes:
  - $(sk_i, pk_i) \leftarrow \mathsf{CGE.KeyGen}(pp_{\mathsf{CG}})$.
  - $\pi_i \leftarrow \mathsf{Kex.Prove}(pp_{\mathsf{Kex}}, pk_i, sk_i)$.
  - Broadcast (pk_i, π_i).
- On receiving $\{(pk_j, \pi_j)\}_{j \neq i}$ each P_i runs for all $j \neq i$: $\mathsf{Kex.Ver}(pp_{\mathsf{Kex}}, pk_j, \pi_j)$. Create the list Q by including all j for which $\mathsf{Kex.Ver}$ returns 1 and also include i. Output $(sk_i, \{pk_i\}_{i \in Q})$.

Security Argument. Now we can argue that the protocol always terminates with a unique set of $\{pk_i\}_{i \in Q}$ and each honest party P_i receiving a corresponding sk_i, and with knowledge of no one else's secret key.

First, from the security of underlying NIZK argument of knowledge (cf. 1) we obtain that if $\mathsf{Kex.Ver}(pk_i, \pi_i)$ returns 1 for some i, then P_i indeed has a correct key pair (sk_i, pk_i) such that $pk_i = g_q^{sk_i} \in G^q$. The statistical zero-knowledge guarantees that everyone only knows their own key and *nothing else*. Finally, from the completeness of NIZK and the correctness of the encryption scheme it is straightforward to see that the protocol always terminates with a unique set of public keys output by the honest parties.

In the following sections, when we say that we **assume a PKI setup**, we imply that participants already executed the protocol Π^{pp}_{PKI}. Without loss of generality, we assume that $Q = \{1, \ldots, n\}$ – this will simplify the notations.

5. NI-VSS using Class Groups

We realize cgVSS, a non-interactive verifiable secret sharing mechanism from class groups. In Section 6 we provide a generic transformation to a NI-DKG, instantiating that with cgVSS we obtain a class-group based NI-DKG protocol cgDKG.

Our cgVSS scheme first establishes a PKI setup. Once a PKI is successfully established the online execution is non-interactive, in that a dealer just broadcasts a single message to the recipients in the online phase. The message contains the class-group based multi-receiver ciphertext (Section 3.2) and associated proofs of correct sharing (Section 3.3). Each recipient then locally decrypt and verify the shares.

5.1. Definition:NI-VSS

Ideal Functionality $\mathcal{F}_{\mathsf{VSS}}$. Following the UC paradigm [71] we provide a VSS ideal functionality, that captures all the properties we desire. The ideal functionality is described in Figure 2.

The functionality is parameterized with a set of public parameters pp, a cyclic group of prime order $\bar{G}$ with a uniform random generator $\bar{g}$; integers n, t such that $n \geq 2t+1$. It interacts with the following ideal (dummy) parties: the dealer P_D, n recipients $P_1, \ldots, P_n$ a public verifier P_V, and an ideal world adversary (a.k.a. simulator) $\mathcal{S}$. It initializes a list $T[\mathsf{sid}]$ for any sid with all entries set to $\top$ by default. Since we are in the static setting, the set of corrupt parties, C is known in the beginning. Based on that we mark sid either `honest` when $|C| \leq t$ or `corrupt` when $|C| > t$.

We discuss how the ideal functionality captures different properties, explicitly captured (and not captured) in existing definitions.

- **Privacy.** This only makes sense when P_D is honest and sid is marked `honest` as well. This is guaranteed by the fact that the simulator only obtains $\bar{g}^s$ in this case. This is captured by virtually all existing definitions in the literature, and also referred to as *secrecy* in some of them.
- **Uniqueness.** For a potentially corrupt dealer, uniqueness guarantees that, a dealing is always associated with a unique value (which maybe $\perp$ when dealing fails to verify). Now, when sid is marked `honest`, then this is captured as $T[\mathsf{sid}]$ is populated only once with either a valid pair $(s, \bar{g}^s)$ or $\perp$. Also, in this case a successful reconstruction is guaranteed, as long as the honest parties agree to participate in the reconstruction. When sid is marked `corrupt` there is a possibility that $T[\mathsf{sid}]$ has $(\star, \bar{h})$. In this case, nonetheless, any reconstruction effort fixes $T[\mathsf{sid}]$ to a specific $(s, \bar{h})$ such that $\bar{h} = \bar{g}^s$ (as checked in Step 3(b)iB). So, once the verifier is committed to a certain s in the exponent through $\bar{h}$, reconstruction becomes unique and consistent. However, when sid is `corrupt` then a successful reconstruction can not be guaranteed. In the literature, similar properties have been captures and are called uniqueness (in [47], [79]), or strong commitment (in [80]).
- **Strong Public Verifiability.** This is guaranteed by Step 4. In particular, whenever $T[\mathsf{sid}]$ is populated, immediately after that any party (including the public verifier) can verify whether the dealing succeeded or not. Note that, in this step, the ideal adversary does not engage, which implies if a dealer is honest, this holds even if all recipients are dishonest (when sid is marked `corrupt`). Due to this we call this property *strong* public verifiability, as it holds regardless of the corruption of the recipients. In contrast, the public verifiability as defined in Das et al. [35] only holds when $|C| \leq t$, which is equivalent to `honest` sid in our setting. Therefore, to differentiate we call our property *strong* public verifiabaility.

1) Upon $(\mathsf{sid}, \texttt{Dealing}, s)$ from P_D: only if P_D is honest, *and* $T[\mathsf{sid}] = \top$: compute a (n, t) secret sharing $(s_1, \ldots, s_n)$. Send (sid, s_i) to each $i \in H$ and $(\mathsf{sid}, \{\bar{g}^{s_i}\}_{i\in[n]})$ to everyone. Additionally send $\{s_i\}_{i\in C}$ to the adversary. Set $T[\mathsf{sid}] \leftarrow (s, \bar{g}^s, \{\bar{g}^{s_i}\}_{i\in[n]})$. /**In this case all properties, including privacy holds, as dealer is honest and the corruption threshold is below $t+1$. Also this facilitates "guaranteed dealing".**/
2) Upon $(\mathsf{sid}, \texttt{Corrupt-Dealing})$ from $\mathcal{S}$: only if P_D is corrupt *and* $T[\mathsf{sid}] = \top$:
   a) If sid is $\texttt{honest}$: wait for $\mathcal{S}$ to send either $(\mathsf{sid}, \{s_i\}_{i\in H}, \{\bar{h}_i\}_{i\in[n]}, s)$, in that case set $T[\mathsf{sid}] \leftarrow (s, \bar{g}^s, \{\bar{h}_i\}_{i\in[n]})$; or $(\mathsf{sid}, \perp)$, in which case set $T[\mathsf{sid}] \leftarrow \perp$.
   b) Else (when sid is $\texttt{corrupt}$): wait for $\mathcal{S}$ to send either $(\mathsf{sid}, \{s_i\}_{i\in H}, \{\bar{h}_i\}_{i\in[n]}, \star)$, in that case reconstruct $\bar{h}$ using Lagrange in the exponent from $\{\bar{h}_i\}_{i\in[n]}$ and set $T[\mathsf{sid}] \leftarrow (\star, \bar{h}, \{\bar{h}_i\}_{i\in[n]})$; or $\mathcal{S}$ sends $(\mathsf{sid}, \perp)$ when set $T[\mathsf{sid}] \leftarrow \perp$.
   c) In any case, if $T[\mathsf{sid}] \neq \perp$ then send (sid, s_i) to each $i \in H$ and $(\mathsf{sid}, \{\bar{h}_i\}_{i\in[n]})$ to everyone; otherwise if $T[\mathsf{sid}] = \perp$, send $(\mathsf{sid}, \perp)$ to everyone.
   /**In this case, a corrupt dealer can not break uniqueness. When* sid *is* $\texttt{corrupt}$ *the dealing is committed via DLog commitment.**/
3) Upon $(\mathsf{sid}, \texttt{Recon})$ from any party P if $T[\mathsf{sid}] = \top$ then skip. Else if $T[\mathsf{sid}] = \perp$, then reply $\texttt{Dealing-Failed}$ to P. Else:
   a) Send $(\mathsf{sid}, \texttt{Recon})$ to the simulator, and when $\mathcal{S}$ responds back with the same message, send it to the honest parties $\{P_i\}_{i\in H}$.
   b) Wait for $\mathcal{S}$ to reply with $\{\tilde{s}_i\}_{i\in C}$, where each $\tilde{s}_i \in \{s_i, \perp, \texttt{No-Response}\}$ and the honest parties P_i to reply with $\tilde{s}_i \in \{s_i, \texttt{No-Response}\}$. Now based on the replies there are three cases:
      i) In total at least $t+1$ parties replies with $s_i \notin \{\perp, \texttt{No-Response}\}$ then reconstruct using Lagrange interpolation to get s.
         A) If $T[\mathsf{sid}] \neq (\star, \cdots)$ then check whether $T[\mathsf{sid}] = (s, \cdot)$, and if that succeeds then send back s to P; otherwise send back $\texttt{Recon-Error}$ to P.
         B) Else if $T[\mathsf{sid}] = (\star, \bar{h}, \cdots)$, check if $\bar{h} = \bar{g}^s$, if that fails send back $\texttt{Recon-Error}$, otherwise send back s to P and set $T[\mathsf{sid}] \leftarrow (s, \bar{h})$.
         /**There are responses from at least $t+1$ parties. However, the responses must be consistent with the committed value during dealing in either cases.**/
      ii) Else if there are $\geq t+1$ i for which $\tilde{s}_i = \texttt{No-Response}$ then reply with $\texttt{Recon-Declined}$ to P./**Parties decline to the reconstruction request.**/
      iii) Else, reply with $\texttt{Recon-Error}$ to P./**In this case there are not enough values, and also not enough explicit decline. This implies there is an error in the execution. It is important to distinguish this from* $\texttt{Recon-Declined}$*, as in the previous case there is no error in the protocol execution.**/
4) Upon $(\mathsf{sid}, \texttt{Verify}, \{\bar{h}_i\}_{i\in[n]})$ from any party P: if $T[\mathsf{sid}] = (\cdots, \{\bar{h}_i\}_{i\in i})$ return $\texttt{Dealing-Succeeded}$. Otherwise, return $\texttt{Dealing-Failed}$. /**This facilitates "strong public verifiability", in any case.**/

Figure 2: Our VSS ideal functionality $\mathcal{F}_{\mathsf{vss}}$

- **Guaranteed Dealing.** This is a new property we observe. This implies that, in no circumstances, an honest dealer can be prevented from committing to a correct dealing. This property is captured by our ideal functionality in the same step as above, when $T[\mathsf{sid}]$ is *immediately populated* in Step 1. Once this is populated with a correct value, this is *uniquely defined*, and can not be reconstructed to anything else, (not even $\perp$). Note that, any attempt to reconstruct to anything else invokes a $\texttt{Recon-Error}$ message. Weaker properties, called *completeness* and *termination* are defined by Das et al. [35]. Their completeness guarantee comes close to this property, however, only considers an honest majority. In fact, their interactive VSS protocol only achieves that, falling short of the guaranteed dealing that we consider. Their termination is basically the same as guaranteed output delivery, as considered in the MPC literature, and is only achievable in an honest majority setting.

Real World NI-VSS protocol. In the real world we describe a generic NI-VSS protocols protocol $\Pi_{\text{NI-VSS}}$ *assuming a PKI setup*. So there are n recipients $P_1, \ldots, P_n$, each P_i knows a secret key sk_i, corresponding to which there is a public key pk_i. There are two other parties, a dealer P_D and a public verifier P_V who do not hold any secret. Everyone knows all public keys $\{pk_i\}_{i\in[n]}$, in addition to the public parameters pp. For a threshold t such that $n \geq 2t+1$ an NI-VSS protocol consists of the following algorithms:

- $\mathsf{Share}(pp, s) \rightarrow (\{s_i\}_{i\in[n]}, \mathsf{cmt}(s), \{\mathsf{cmt}(s_i)\}_{i\in[n]})$. The sharing algorithm produces t out of n Shamir's secret shares of a value s such that $(s_1 \ldots, s_n) = \mathsf{Shamir}_{n,t,q}(s)$ and the associated commitments $\mathsf{cmt}(s), \mathsf{cmt}(s_1), \ldots, \mathsf{cmt}(s_n)$. Define

$\mathsf{cmt} \leftarrow (\mathsf{cmt}(s), \mathsf{cmt}(s_1), \ldots, \mathsf{cmt}(s_n))$.

- $\mathsf{ShareEnc}(pp, \mathsf{cmt}, \{s_i, pk_i\}_{i\in[n]}) \rightarrow (R, \{E_i\}_{i\in[n]}, \pi_{\mathsf{CS}})$. On input n many shares $s_1, s_2, \ldots$, the associated commitments cmt, and corresponding public keys , this algorithm outputs a multi-receiver ciphertext $(R, E_1, E_2, \ldots, E_n)$ plus a proof of correct sharing π_{CS}.
- $\mathsf{Verify}(pp, \mathsf{cmt}, R, \{E_i, pk_i\}_{i\in[n]}, \pi_{\mathsf{CS}}) \rightarrow 1/0$. This algorithm verifies the entire ciphertext tuple with respect to the proof π_{CS} and the commitment to output a decision bit.
- $\mathsf{ShareDec}(pp, sk_i, R, E_i) \rightarrow s_i$. The decryption algorithm uses a specific secret-key sk_i to decrypt ciphertext (R, E_i). Note that, only the party who posses sk_i can decrypt (R, E_i).
- $\mathsf{CmtVer}(\mathsf{cmt}, i, s_i) \rightarrow 1/0$. This algorithm checks the consistency of the i-th opening s_i with commitment cmt.

In the real world protocol, $\Pi_{\text{NI-VSS}}$ The parties execute these algorithms and interact as described in Figure 3. Corruption in real world is attributed to adversary denoted by $\mathcal{A}$. We consider n-bounded PPT adversaries.

Definition 6 (NI-VSS)**.** We say an instantiation of the protocol $\Pi_{\text{NI-VSS}}$ a *secure* NI-VSS if it *securely realizes* the ideal functionality $\mathcal{F}_{\mathsf{VSS}}$ in the PKI setup.

5.2. Our NI-VSS Protocol: cgVSS

In this section we provide a concrete instantiation of a $\Pi_{\text{NI-VSS}}$ protocol based on the multi-receiver encryption scheme (cf. Section 3.2), a corresponding proof of correct sharing (cf. Section 3.3) in the class group setting, assuming a PKI setup for class-groups (cf. Section 4.1). The instantiation is provided in Figure 4. We call our instantiation cgVSS. We prove the following theorem:

Theorem 4 (Security of cgVSS)**.** cgVSS *is is a secure NI-VSS assuming a PKI for class-groups as long as the underlying multi-receiver encryption scheme is secure (Def 5) and the NIZK proof of correctness is a secure proof system (Def 1).*

Proof. Consider four mutually exclusive and exhaustive cases:

- CASE-1 When P_D is honest and $|C| \le t$.
- CASE-2 When P_D is corrupt and $|C| \le t$.
- CASE-3 When P_D is corrupt and $|C| > t$.
- CASE-4 When P_D is honest and $|C| > t$.

For each Case-i we construct a separate simulator $\mathcal{S}_i$. Our actual simulator $\mathcal{S}$ first obtains pp_{CG} by running CG.ParamGen and then invokes the PKI setup with the adversary by choosing secret keys $\{sk_i\}_{i\in H}$ for the honest parties. At the end it receives all public keys $\{pk_i\}_{i\in[n]}$. Then it simply runs $\mathcal{S}_i$ with input $pp_{\mathsf{CG}}, \{pk_i\}_{i\in[n]}$ based on which case it is in – since we are in the static corruption model, this will be known in the beginning. We describe each simulators in details now, and argue that why the simulation is correct.

- **Input.** Only the dealer P_D has an input $s \in \mathbb{Z}_q$.
- **Dealing**. The dealer P_D executes:
  - $(\{s_i\}_{i\in[n]}, \{a_j\}_{j\in[t]}, \mathsf{cmt}) \leftarrow \mathsf{Share}(pp, s)$
  - Compute $(R, \{E_i\}_{i\in[n]}, \pi_{\mathsf{CS}}) \leftarrow \mathsf{ShareEnc}(pp, \{s_i, pk_i\}_{i\in[n]})$
  - Broadcast dealing $D = (R, \{E_i\}_{i\in[n]}, \mathsf{cmt}, \pi_{\mathsf{CS}})$ to all receivers $\{P_i\}_{i\in[n]}$.
- **Receiving**. Each recipient P_i for $i \in [n]$, on receiving D performs the following steps:
  - $e \leftarrow \mathsf{Verify}(pp, \{pk\}_{i\in[n]}, D)$ where $D = (R, \{E_i\}_{i\in[n]}, \mathsf{cmt}, \pi_{\mathsf{CS}})$
  - If $e = 1$ then $s_i \leftarrow \mathsf{ShareDec}(pp, sk_i, R, E_i)$ and define $y_i \leftarrow s_i$ as its share corresponding to the dealing D; otherwise, if $e = 0$ reject dealing D, and set $y_i \leftarrow \perp$.
  - Each recipient has a private output y_i.
  - Parse cmt as $(A_0, \ldots, A_t)$. The common public output is $(\bar{h}_1, \ldots, \bar{h}_n)$ where each $\bar{h}_i \leftarrow \prod_{j=0}^{j=t} A_j^{i^j}$.
- **Reconstruction.** Any party P can broadcast a reconstruction request. On receiving a reconstruction request each recipient P_i may broadcast share y_i. On receiving the shares y_j, the requester P executes:
  - For each j, if $y_j \neq \perp$ then check $b_j \leftarrow \mathsf{CmtVer}(\mathsf{cmt}, j, s_j)$. Set $y_j \leftarrow \perp$ if $b_j = 0$.
  - If there are at least $t + 1$ j (including when $P = P_j$) for which $y_j \neq \perp$, then reconstruct y by choosing any $t + 1$ y_j's (maybe chosen in a lexicographic order). Otherwise set $y \leftarrow \perp$.

Figure 3: The generic NI-VSS protocol in the PKI.

Case-1. For simplicity suppose that $|C| = t$ (the other cases can extended in a straightforward manner). The simulator $\mathcal{S}_1$ obtains $(\{s_i\}_{i\in C}, \{\bar{h}_i\}_{i\in[n]})$ in this case and works as follows:

- Define $(A_0, \ldots, A_t)$ by using a linear transformation from n evaluations to $t+1$ coefficients in the exponent. Let $\mathsf{cmt} \leftarrow (A_0, \ldots, A_t)$.
- Let $s'_i \leftarrow 0$ for all $i \in H$. and $s'_i \leftarrow s_i$ for all $i \in C$.
- Compute $(R, \{E_i\}_{i\in[n]}, \pi_{\mathsf{CS}}) \leftarrow \mathsf{ShareEnc}'(pp, \mathsf{cmt}, \{s'_i, pk_i\}_{i\in[n]})$ where $\mathsf{ShareEnc}'$ is the same as ShareEnc (as described in Fig. 4) except that the proof π_{CS} is generated using the zero-knowledge simulator $\mathcal{S}_{\mathsf{PoC}}$.
- Send $(R, \{E_i\}_{i\in[n]}, \pi_{\mathsf{CS}}, \mathsf{cmt})$ to the adversary.
- On receiving (sid, `Recon`) from the ideal functionality, it forwards the reconstruction request to the adversary and then when adversary sends back $\{s'_i\}_{i\in C'}$ for $C' \subseteq C$, define $\tilde{s}_i \leftarrow$ `No-Response` for all $i \in C \backslash C'$. Then for each $i \in C'$ checks whether $\bar{g}^{s'_i} = \bar{h}_i$. If not, then set $\tilde{s}_i \leftarrow \perp$, else set $\tilde{s}_i \leftarrow s'_i$. Finally sends $\{\tilde{s}_i\}_{i\in C}$ to $\mathcal{F}_{\mathsf{VSS}}$.

To argue the simulation is correct we start from the real protocol and through a number of hybrids gradually move to the ideal world. The hybrids are described as follows:

- **Hybrid** Hyb_1**.** This hybrid is the same as cgVSS, except that the proof of correctness π_{CS} is now simulated, and thus is independent of the witness $\mathsf{wit} = ((s_1, \ldots, s_n), r)$. Syntactically, instead of ShareEnc in Step 2, ShareEnc$'$ (as defined above) is run. This step is statistically close to the real world execution of cgVSS, which follows from the statistical zero-knowledge property of the proof of correctness.
- **Hybrid** Hyb_2**.** This hybrid is the same as Hyb_1 except that now, the secret s is not known, and the honest party's shares are defined to be 0.

  We provide the details below with the changes highlighted in blue.

  1) Denote the set of corrupt parties by $C \subset [n]$ such that $|C| \le t$ and $n \notin C$ (the dealer is not corrupt). Define the set of honest parties as $H \leftarrow [n] \setminus C$.
  2) Sample a uniform random $s \xleftarrow{\$} \mathbb{Z}_q$. Run
  $$(\{s_i\}_{i\in[n]}, \{a_j\}_{j\in[t]}, \mathsf{cmt}) \leftarrow \mathsf{Share}(pp, s),$$
  where $\mathsf{cmt} = (A_0, \ldots, A_t)$.
  3) For all $i \in [n]$ sample $s'_i \xleftarrow{\$} \mathbb{Z}_q$ if $i \in C$, and $s'_i \leftarrow 0$ if $i \in H$. Furthermore, use A_0 and $\{s_i\}_{i\in C}$ to re-define $\mathsf{cmt} = (A_0, A_1, \ldots, A_t)$ using Lagrange interpolation in the exponent.
  4) Compute $(R, \{E_i\}_{i\in[n]}, \pi_{\mathsf{CS}}) \leftarrow \mathsf{ShareEnc}'(pp, \mathsf{cmt}, \{s'_i, pk_i\}_{i\in[n]})$, where π_{CS} is a simulated proof.
  5) Then give the following to $\mathcal{A}$: $\big(R, \{E_i\}_{i\in[n]}, \mathsf{cmt}, \pi_{\mathsf{CS}}\big)$
  6) The rest remains unchanged.

  We prove that:

  Lemma 5. Hyb_1 *and* Hyb_2 *are computationally indistinguishable as long as the underlying multi-receiver encryption scheme is secure.*

  Proof. For any adversary $\mathcal{A}$ that distinguishes between the hybrids we construct an *admissible* reduction which breaks the security of underlying multi-receiver encryption scheme as follows:

  – Obtain pp_{CG} from the challenger.
  – Send C to the challenger obtain $\{pk_i\}_{i\in H}$.
  – Sample appropriate pp_{Kex} for the NIZK argument of knowledge to be used in the PKI setup. Let $pp \leftarrow \{pp_{\mathsf{CG}} \cup pp_{\mathsf{Kex}}\}$. Then run a PKI setup protocol Π^{pp}_{PKI} to obtain $\{pk_i\}_{i\in C}$
  – Obtain pp_{CG} and n public keys $pk_1, \ldots, pk_n$ from the encryption challenger. Send C to the challenger to get back $\{sk_i\}_{i\in C}$. Sample additional public parameters to compute pp_{PoC} for the proof of correct sharing scheme such that they are consistent with pp_{CG}. Give $pp_{\mathsf{CG}} \cup pp_{\mathsf{PoC}} \cup \{pk_i\}_{i\in[n]} \cup \{sk_i\}_{i\in C}$ to $\mathcal{A}$.
  – Send $\vec{m}_0$ and $\vec{m}_1$ to the challenger where $\vec{m}_0$ and $\vec{m}_1$ are computed as follows:
    ∗ Sample $s \xleftarrow{\$} \mathbb{Z}_q$ and $s_i \xleftarrow{\$} \mathbb{Z}_q$ for all $i \in C$. Using Lagrange interpolation compute $\{s_i\}_{i\in H}$.
    ∗ For all $b \in \{0,1\}$ and all $i \in C$ set $m_{b,i} = s_i$.
    ∗ For all $i \in H$ set $m_{0,i} \leftarrow s_i$ and $m_{1,i} \leftarrow 0$.
  – When the challenger returns $(R, \{E_i\}_{i\in[n]})$, compute:
    ∗ $\mathsf{cmt} \leftarrow (A_0, \ldots, A_t)$ computed by linear transformation to coefficients in the exponent.
    ∗ Use the zero-knowledge simulator $\mathcal{S}_{\mathsf{PoC}}$ of the proof of correct sharing to generate a simulated proof π_{CS} using the instance:
    $$\big(\{h_i\}_{i\in[n]}, (R, \{E_i\}_{i\in[n]}, \mathsf{cmt})\big)$$
  – Send the following to $\mathcal{A}$:
  $$\big(R, \{E_i\}_{i\in[n]}, \mathsf{cmt}, \pi_{\mathsf{CS}}\big)$$
  – When $\mathcal{A}$ concludes Hyb_1 return 0 to the challenger, and in case $\mathcal{A}$ concludes Hyb_2, then send 1.

  It is easy to argue that if $b = 0$, $\mathcal{A}$'s view is the same as in Hyb_1, and when $b = 1$, that is the same as in Hyb_2. So the probability of $\mathcal{A}$'s breaking Hyb_1 and Hyb_2 is upper bounded by the probability of the reduction's breaking the security of the encryption. This concludes the proof. □

Finally we note that Hyb_2 is identical to the ideal world is we just use the simulator $\mathcal{S}_1$ instead, as the change is only syntactic. This concludes the analysis for this case. □

Case-2. The simulator $\mathcal{S}_2$ works as follows:

- Run Π_{PKI} with appropriate parameters to get $\{pk_i\}_{i\in[n]}$ and $\{sk_i\}_{i\in H}$.
- Once receive $(R, \{E_i\}_{i\in[n]}, \pi_{\mathsf{CS}}, \mathsf{cmt})$ from the adversary. Parse cmt as $(A_0, \ldots, A_t)$. Verify π_{CS}, if that fails then send $(\mathsf{sid}, \perp)$ to $\mathcal{F}_{\mathsf{VSS}}$ in Step 2a. Otherwise:
  – Decrypt using $\{sk_i\}_{i\in H}$ to obtain $\{s_i\}_{i\in H}$. The use Lagrange to compute $\{s_i\}_{i\in C}$. Since there are at least $t+1$ honest parties, reconstruction of all s_i is possible. Compute $\{\bar{h}_i\}_{i\in[n]}$ by computing linear transformation to n evaluations in the exponent from $(A_0, \ldots, A_t)$. Send $(\mathsf{sid}, \{s_i\}_{i\in H}, \{\bar{h}_i\}_{i\in[n]}, s)$ to $\mathcal{F}_{\mathsf{VSS}}$ in Step 2a.
- For a $(\mathsf{sid}, \mathtt{Recon})$ query, communicate with the adversary about reconstruction. When adversary sends back $\{s'_i\}_{i\in C'}$ for $C' \subseteq C$, define $\tilde{s}_i \leftarrow \mathtt{No\text{-}Response}$ for all $i \in C \setminus C'$. Then for each $i \in C'$ checks whether $\bar{g}^{s'_i} = \bar{h}_i$. If not, then set $\tilde{s}_i \leftarrow \perp$, else set $\tilde{s}_i \leftarrow s'_i$. Finally sends $\{\tilde{s}_i\}_{i\in C}$ to $\mathcal{F}_{\mathsf{VSS}}$.

To analyze the correctness of simulation we rely on the soundness of the NIZK proof of correct sharing and the binding of DLog commitments. In particular, the soundness ensures that as long as the proof π_{CS} verifies correctly, the dealer's message encrypts a share s correctly. Therefore, using Lagrange interpolation to construct all shares is indeed correct. Furthermore, the binding of cmt ensures that, as long as dealing succeeds the session's secret is uniquely defined, and hence any effort to reconstruct to another value is bound to fail.

Case-3. This is similar to Case-2, but now sid is $\mathtt{corrupt}$ and hence the simulator, after PKI has less than $t+1$ secret keys $\{sk_i\}_{i\in H}$ for $|H| < t+1$. Therefore, once it obtains the message $(R, \{E_i\}_{i\in[n]}, \mathsf{cmt}, \pi_{\mathsf{CS}})$, it first checks the proof π_{CS} – if that fails then send $(\mathsf{sid}, \perp)$ in Step 2b, otherwise it obtains $< t+1$ values $\{s_i\}_{i\in H}$ and send $(\mathsf{sid}, \{s_i\}_{i\in H}, \{\bar{h}_i\}_{i\in[n]}, \star)$, where $\{\bar{h}_i\}_{i\in[n]}$ are computed from cmt by linear transformation in the exponent. Furthermore, when $(\mathsf{sid}, \texttt{Recon})$ is received, then it ensures that the response is consistent with the committed value $\bar{h} = \bar{g}^s$. So, the correctness follows again from the soundness of the proof of correct sharing plus the binding of the DLog commitment. We skip the details.

Case-4. In this case, the simulator obtains (sid, s) in Step 1. This means, there is no privacy of s and the simulation becomes straightforward for this part. However, the (sid, `Recon`) is handled just like the above. This ensures guaranteed dealing, because as long as there are at least $t+1$ parties who are willing to reconstruct, the value s can be (uniquely) reconstructed. Furthermore, even if reconstruction is not possible, that is either `Recon-Declined` or `Recon-Error` is returned, the corrupt parties can not prevent a successful (public) verification, as in this case $T[\mathsf{sid}] \neq \perp$. `Dealing-Failed` is returned by $\mathcal{F}_{\mathsf{VSS}}$ if and only if $T[\mathsf{sid}] = \perp$.

6. NI-DKG using Class Groups

An NI-DKG protocol can be thought of as a symmetric extension of NI-VSS, with the crucial difference that no one knows the secret in NI-DKG. Indeed, following prior works (e.g. [7], [8], [28]), we construct NI-DKG by using our NI-VSS scheme in Figure 4. First we provide a simple new definition of DKG in the universal composability (UC) framework [71]. The ideal functionality $\mathcal{F}_{\mathsf{DKG}}$ is depicted in Fig. 5. To avoid confusion with encryption public keys we denote the output of the functionality as follows: joint (resp. individual) public key by y (resp. y_i) and secret keys by x_i. Then we provide a generic construction (which is essentially the same as [8]) in Fig. 6 in the $\mathcal{F}_{\mathsf{VSS}}$ hybrid model and prove that it satisfies the UC definition.

Different guarantees by $\mathcal{F}_{\mathsf{DKG}}$. The functionality provides different guarantees depending on the modes. In particular, when $n \geq 2t+1$ and $t' = |C| \leq t$, then the mode is set STRONG, in which the functionality achieves guarantees such as *privacy* and *robustness* and *public verifiability*. In contrast, the WEAK mode only offers *public verifiability*. Informally, public verifiability guarantees that, from the transcript of the protocol anyone (even outside the system) can verify whether $\{y_i\}_{i\in[n]}$'s exponents are indeed t out of n secret sharing of y. The lack of robustness in WEAK mode is captured in Step 4, which allows the simulator to abort only in this case. Since this is not allowed in STRONG mode, that offers robustness.

Privacy follows from the fact that, in STRONG mode the simulator never obtains the secret keys for the honest parties, whereas in WEAK mode, the simulator gets their initial dealings (Step 1(a)ii) and hence can learn all secrets. However, it is important to note that, the secret can not be biased in this case, as the simulator only obtains the secrets after it sends the commitments of the corrupt party's secrets. On the flip side this puts a restriction on our adversarial model, as the adversary has to be non-rushing in this case, and in that case the so-called key-biasing is out of scope. One may contemplate a weaker definition where the simulator gets honest party's secret *before* it sends the corrupt party's commitments. This would let us work with rushing adversaries as well. However, in that case it is not only impossible to prevent biasing against public-key, but also against secret-key. In particular, the rushing adversary may just choose the corrupt secrets once it obtains all the honest secrets, setting the final secret to, for example, 0. Such a guarantee seems to be useless.

Finally, note that in either mode public verifiability is guaranteed as noted in Step 5. In STRONG mode public

- **Ingredients.** The NI-VSS algorithms described below uses the following ingredients.
  - A multi-receiver encryption scheme (cf. Section 3.2) with algorithms $(\mathsf{CGE.KeyGen}, \mathsf{CGE.mrEnc}, \mathsf{CGE.Dec})$ and public parameters pp_{CG}.
  - An associated proof system of correct sharing (cf. Section 3.3) with algorithms $(\mathsf{PoCS.Prove}, \mathsf{PoCS.Ver})$ and public parameters pp_{PoC}, which is consistent with pp_{CG}.
- **Public parameters.** The public parameter pp is defined as $pp \leftarrow pp_{\mathsf{CG}} \cup pp_{\mathsf{PoC}}$.

<u>**Construction**</u>

- $\mathsf{Share}(pp, s) \rightarrow (\{s_i\}_{i\in[n]}, \mathsf{cmt})$:
  - Sample $a_j \xleftarrow{\$} \mathbb{Z}_q, j \in [t]$.
  - Set $a_0 \leftarrow s$.
  - Define $P(x) = a_0 + a_1 x + \ldots + a_t x^t$.
  - For each $i \in [n]$: set $s_i \leftarrow P(i)$.
  - Compute for all $j \in \{0, \ldots, t\}$: $A_j \leftarrow \bar{g}^{a_j}$.
  - Set $\mathsf{cmt} \leftarrow \{A_0, \ldots, A_t\}$.
- $\mathsf{ShareEnc}(pp, \mathsf{cmt}, \{s_i, pk_i\}_{i\in[n]}) \rightarrow (R, \{E_i\}_{i\in[n]}, \pi_{\mathsf{CS}})/\perp$.
  - Sample $r \xleftarrow{\$} \mathcal{D}$
  - Compute $(R, \{E_i\}_{i\in[n]}) \leftarrow \mathsf{CGE.mrEnc}(pp_{\mathsf{CG}}, \{h_i, s_i; r\}_{i\in[n]})$.
  - Define:
    - $\mathsf{inst} = \big(\{h_i\}_{i\in[n]}, (R, \{E_i\}_{i\in[n]}), \mathsf{cmt}\big)$.
    - $\mathsf{wit} = \big((s_1, \ldots, s_n), r\big)$.
  - Compute $\pi_{\mathsf{CS}} \leftarrow \mathsf{PoCS.Prove}(pp_{\mathsf{PoC}}, \mathsf{inst}, \mathsf{wit})$.
- $\mathsf{Verify}(pp, \mathsf{cmt}, R, \{E_i, pk_i\}_{i\in[n]}, \pi_{\mathsf{CS}}) \rightarrow 1/0$:
  - Parse $\mathsf{inst} \leftarrow \big(\{h_i\}_{i\in[n]}, (R, \{E_i\}_{i\in[n]}), \mathsf{cmt}\big)$.
  - Output $\mathsf{PoCS.Ver}(pp_{\mathsf{PoC}}, \mathsf{inst}, \pi_{\mathsf{CS}})$.
- $\mathsf{ShareDec}(pp, sk_i, R, E_i) \rightarrow s_i$:
  - Compute $s_i \leftarrow \mathsf{CGE.Dec}(pp_{\mathsf{CG}}, sk_i, R, E_i)$.
  - $\mathsf{CmtVer}(\mathsf{cmt}, i, s_i) \rightarrow 1/0$: Parse cmt as $A_0, \ldots, A_t$. Check if $\bar{g}^{s_i} \stackrel{?}{=} \prod_{j=0}^{j=t} A_j^{i^j}$.

Figure 4: Concrete instantiation of cgVSS

verifiability is captured easily, because the secret sharing executed by the functionality itself, and the list L has an entry only if that is done correctly. However, it is more involved to see in the WEAK mode, because in that mode the entries in L is defined by the simulator. Nevertheless, in Step 3a, the ideal functionality checks that whether the values returned by the simulator indeed forms a t out of n secret sharing of y.[9] So, similar to VSS, we can refer to this as *strong* public verifiability, as opposed to simply public verifiability, which was considered only in a setting equivalent to our STRONG mode.

Our definition compared to state-of-art. Our definition differs from prior UC-based DKG definitions [9], [47], [48], [49] significantly. This is because, first we formally capture the strong variant of public verifiability separately for the first time (as far as we know). We handle two modes STRONG and WEAK within a single functionality in a more fine-grained manner. Furthermore, our definition (only in the STRONG mode) allows biasing of the final public key y in a manner, as described in Gennaro et al. [7]. Nevertheless, as also shown in earlier works, this weaker definition suffices for many threshold applications such as threshold Schnorr's signature [7], BLS [44] etc. while offering efficiency benefit. In fact, as we show in Appendix E, that our definition satisfies the so-called oracle-aided simulatability requirement as defined by Bacho and Loss [44] which is sufficient for many important applications. We note that, though Bacho and Loss showed that a number of prior interactive DKG (such as JF-DKG [81]) satisfies the required oracle-aided simulatability, our definition is the first formalization that captures the biasability of the joint public key in the UC model. We briefly discuss Appendix D the measures to remove this "biasability" with a two round protocol. We note that a recent work by Katz [50] also captures this in a slightly different manner.

We note that, although the joint public key y can be biased towards a specific value, it is not possible for the adversary to execute a more devastating attack. For example, the adversary can not predict x with non-negligible probability. This is captured by our functionality, as given the $\bar{g}^{s_i}$ values for honest i, the simulator may fix $y = \bar{g}^x$, but then $x = \sum_{i \in H} s_i + \sum_{i \in C} s_i$. So, predicting x implies knowing $\sum_{i \in H} s_i$ which is hard due to discrete log over $\bar{G}$.

Our protocol uses a generic transformation from any NI-VSS scheme to a NI-DKG scheme. This transformation is sort of "g=folklore" and was used in [8]. The basic idea is quite simple: each party P_i now runs an NI-VSS instance using her own secret s_i; after the completion of the protocol, s_i is computed by *linearly combining* own share of s_i with shares of s_j received from other P_j. We present the generic protocol in $\mathcal{F}_{\mathsf{VSS}}$-hybrid in Figure 6. When $\mathcal{F}_{\mathsf{VSS}}$ is instantiated with cgVSS we call the resulting DKG protocol cgDKG.

We prove the following theorem formally.

9. This can be done by, for example, a simple linear code check in the exponent akin to [6].

Theorem 6 (Security of Generic DKG). *For parameters $n, t \in \mathbb{N}$ such that $n \geq 2t + 1$, the generic DKG protocol securely UC-realizes $\mathcal{F}_{\mathsf{DKG}}$ in $\mathcal{F}_{\mathsf{VSS}}$-hybrid for the following adversary:*

- *Any t-bounded PPT adversary.*
- *Any n-bounded non-rushing PPT adversary.*

Proof. We analyze two different modes. First let us consider the STRONG mode when $n \geq 2t + 1$ and $t' = |C| \leq t$. For simplicity assume $t' = t$.

Specifically, for any PPT adversary $\mathcal{A}$ that corrupts a set C of size $\leq t$ in the real protocol cgDKG, we construct a PPT simulator $\mathcal{S}$ in the ideal world. The simulator simulates the honest party's response and the ideal functionality $\mathcal{F}_{\mathsf{VSS}}$'s response to the adversary. It works as follows:

- Obtain $\{\bar{h}_i\}_{i \in H}$ from $\mathcal{F}_{\mathsf{DKG}}$. For each $i \in H$:
  - Choose $\{s_{ij}\}_{j \in C}$ uniformly at random. Note that these value together with $\bar{h}_i$ uniquely defines all $\{s_{ij}\}_{j \in [n]}$.
  - Compute $\bar{h}_{ij}$ for all $j \in [n]$ using Lagrange in the exponent.
  - Send $(\mathsf{sid}_i, \{\bar{h}_{ij}\}_{j \in [n]}, \{s_{ij}\}_{j \in C})$ to the adversary.
- For all $i \in C$ (assuming, for simplicity, no $\perp$ is returned) receive $(\mathsf{sid}_i, s_i, \{s_{ij}\}_{j \in H}, \{\bar{h}_{ij}\}_{j \in [n]})$ or $\perp$ from $\mathcal{F}_{\mathsf{VSS}}$. Reconstruct $\{\bar{h}^i\}_{i \in C}$ using Lagrange in the exponent. Send $\{s_i\}_{i \in C}$ to $\mathcal{F}_{\mathsf{DKG}}$.
- Get back $\{(y, \{y_i\}_{i \in [n]}), \{x_i\}_{i \in C}\}$ from $\mathcal{F}_{\mathsf{DKG}}$, which it outputs.
- Store $(\mathsf{sid}, \{y_i\}_{i \in [n]}, \{\bar{h}_{ij}\}_{i,j \in [n]})$.
- In response to $(\mathsf{sid}, \texttt{Verify}, \{y_i\}_{i \in [n]})$ then look up an $(\mathsf{sid}, \{y_i\}_{i \in [n]}, \{\bar{h}_{ij}\}_{i,j \in [n]})$, if not found output 0, otherwise check whether each $y_i = \prod_{j \in [n]} \bar{h}_{ij}$ for all $i \in [n]$. If all of them satisfies, then output 1, else output 0.

The simulation is correct because we are in the setting when $t' \leq t$, which means the simulator can choose the corrupt party's shares uniformly at random given each honest party's commitments.

In the WEAK mode, we assume a non-rushing adversary. So, the simulator obtains for all $i \in C$ $(\mathsf{sid}_i, \{s_{ij}\}_{j \in H}, \{\bar{h}_{ij}\}_{j \in [n]})$ (or $\perp$, but for simplicity we assume it does not receive any $\perp$) from multiple instances of $\mathcal{F}_{\mathsf{VSS}}$ before it sends anything to the adversary. Then the simulator works as follows:

- Obtain $\{s_i\}_H$ from $\mathcal{F}_{\mathsf{DKG}}$.
- For all $i \in H$: send $(\mathsf{sid}_i, \texttt{Dealing}, s_i)$ to $\mathcal{F}_{\mathsf{VSS}}$. Get back $\{s_{ij}\}_{j \in H}$ and $\{\bar{h}_{ij}\}_{j \in [n]}$.
- For all $i \in H$ compute $x_i \leftarrow \sum_j s_{ij}$ and for all $i \in [n]$ compute $y_i \leftarrow \prod_{j \in [n]} \bar{h}_{ij}$.
- Send $(\mathsf{sid}, \{x_i\}_{i \in H}, \{y_i\}_{i \in [n]})$ to $\mathcal{F}_{\mathsf{DKG}}$.
- Store $(\mathsf{sid}, \{y_i\}_{i \in [n]}, \{\bar{h}_{ij}\}_{i,j \in [n]})$.
- It may send $(\mathsf{sid}, \texttt{Failure})$ in certain cases, for example if $t' = n$ and all corrupt party returns $\perp$.
- In response to $(\mathsf{sid}, \texttt{Verify}, \{y_i\}_{i \in [n]})$ then look up an $(\mathsf{sid}, \{y_i\}_{i \in [n]}, \{\bar{h}_{ij}\}_{i,j \in [n]})$, if not found output 0, otherwise check whether each $y_i = \prod_{j \in [n]} \bar{h}_{ij}$ for all

The ideal functionality $\mathcal{F}_{\mathsf{DKG}}$ interacts with $n+1$ ideal parties $P_1, \ldots, P_n, P_v$ and an ideal adversary, the simulator $\mathcal{S}$. The functionality is also parameterized with a reconstruction threshold $t < n$ and a group $\langle \bar{g} \rangle = \bar{G}$ of prime order q where discrete log is hard. Since we assume a static corruption setting, we consider another parameter $t' = |C|$, that denotes the number of corrupted parties (also define $H = [n] \setminus C$). The functionality works in two modes STRONG and WEAK. If $n \geq 2t+1$ and $t' \leq t$ it sets the mode to STRONG, otherwise it sets to WEAK mode. It works as follows:

1) Upon receiving $(\mathsf{sid}, \mathtt{Dealing})$ from all n parties: only if sid is unmarked then:
   a) For each $i \in H$ choose a uniform random $s_i \xleftarrow{\$} \mathbb{Z}_q$. Then:
      i) If mode is STRONG then send $\{\bar{g}^{s_i}\}_{i \in H}$ to $\mathcal{S}$./**In this mode $\mathcal{S}$ gets only the commitments, so privacy holds.**/
      ii) Else, when mode is WEAK wait for the simulator to send $\{\bar{g}^{s_i}\}_{i \in C}$. Then send $\{s_i\}_{i \in H}$ to $\mathcal{S}$. /**In this mode, privacy is not guaranteed since s_is are provided to the simulator.**/
2) Upon $(\mathsf{sid}, \{s_i\}_{i \in C} \in \mathbb{Z}_q,)$ from $\mathcal{S}$: only if (i) sid is marked $\mathtt{Live}$; (ii) and the mode is STRONG:
   a) Initialize a set V, and include i into V only if $s_i \neq \perp$.
   b) Compute $s = \sum_{i \in H \cup V} s_i$.
   c) Choose uniform random t-degree $P(x) \in \mathbb{Z}_q^t[x]$ subject to $P(0) = s$. Set $x \leftarrow s$ and $y \leftarrow \bar{g}^s$. Also set $y_i \leftarrow \bar{g}^{x_i}$ where $x_i \leftarrow P(i)$ for all $i \in [n]$.
   d) Finally send (x_i, y_i, y) to party $i \in H$; $(\mathsf{sid}, y, \{y_i\}_{i \in H \cup V}, \{x_i\}\}_{i \in V})$ to $\mathcal{S}$ and y to P_v.
   e) Mark sid $\mathtt{End}$ and store $(\mathsf{sid}, \{y_i\}_{i \in [n]})$ into a list L.

   /**This mode offers robustness, privacy and public verifiability.**/
3) Upon $(\mathsf{sid}, y, \{x_i\}_{i \in H}, \{y_i\}_{i \in [n]})$ from $\mathcal{S}$: only if (i) sid is marked $\mathtt{Live}$; *and* (ii) the mode is WEAK:
   a) Let $y = \bar{g}^x$ and $\{y_i = \bar{g}^{x_i}\}_{i \in [n]}$, where x and $\{x_i\}_{i \in C}$ are unknown. Check whether x_i's are a t out of n Shamir's secret sharing of x. This can be checked in the exponent, for example, by choosing a random linear code in the orthogonal space defined by $y_0, \ldots, y_n$. If this check fails skip. Else go to the next step.
   b) Send(x_i, y_i, y) to party $i \in H$.
   c) Send $y, \{y_i\}_{i \in [n]}$ to P_v.
   d) Mark sid $\mathtt{End}$ and store $(\mathsf{sid}, \{y_i\}_{i \in [n]})$ into a list L.

   /**This modes guarantees only public verifiability.**/
4) Upon $(\mathsf{sid}, \mathtt{Failure})$ from $\mathcal{S}$: only if (i) sid is marked $\mathtt{Live}$; *and* (ii) the mode is WEAK: then send $\perp$ to everyone and mark sid $\mathtt{End}$./**In the WEAK mode, abort is allowed, so robustness does not hold.**/
5) Upon $(\mathsf{sid}, \mathtt{Verify}, \{y_i\}_{i \in [n]})$ from any party P: return 1 if and only if $\exists$ $(\mathsf{sid}, \{\{y_i\}_{i \in [n]}\}) \in L$ and 0 otherwise./**In all modes public verifiability holds.**/

Figure 5: The ideal functionality $\mathcal{F}_{\mathsf{DKG}}$

$i \in [n]$. If all of them satisfies, then output 1, else output 0.

In this case, the simulator obtains honest party's dealings, only after it sends corrupt party's commitments. This is exactly the reason a non-rushing restriction is needed. Apart from that, the simulation is very similar to the STRONG case. □

7. Experimentation and Performance Analysis

Implementation and Setup. We implement cgVSS in C++ using the BICYCL library [82] for class groups, Miracl C++ library for cryptographic operations with $\sim$ 1858 lines of code. For comparison, we adapt and realize a version of the implementation of GrothVSS without forward secrecy in Rust in ~ 4178 lines of code (available at the link https://github.com/Entropy-Foundation/class-group)

We run the experiments with each node realized on a Google Cloud Platform (GCP) instance with an Intel Xeon 2.8GHz CPU with 16 cores and 16GB RAM. We use HotStuff state machine replication [83] to realize the broadcast. Our SMR instance is realized over four GCP instances separate from the DKG nodes. All the reported timings are averages over 10 runs of the protocols.

Communication/Storage Overhead. In cgVSS, the dealer generates 256-bit shares for each party in the system and encrypts them. The encryption of each share consists of two elements (c, d), where c is the exponentiated randomness. In the multi-receiver encryption mechanism, the randomness can be reused across multiple receivers. Hence while encrypting the share values for n receivers, the dealer uses one element for randomness and n elements for the second element of the encryption tuples. Each element in the compressed form takes 1752 bits. The dealer commits to the t coefficients of the polynomial. Hence the total bit-length length for the multi-receiver encryption and commitments is $(1752) \cdot (n+1) + 384 \cdot t$. For the proof of correctness, the dealer also forwards 5 elements, including two class group elements, one elliptic curve element, and two scalars. Figure 7 shows the total bit-length of the dealing (the broadcast message). For 100 users, the dealer broadcasts a message (dealing) of length 201.55Kb whereas, for 150

Ingredients and parameters. We consider n parties $P_1, \ldots, P_n$ are running this protocol with a threshold $t < n/2$ in $\mathcal{F}_{\mathsf{VSS}}$ hybrid. We also consider a separate public verifier P_v. The functionality is parameterized by a cyclic group of prime order p with generator $\bar{g}$.

Protocol

Dealing. Each party P_i, upon a dealing request $(\mathsf{sid}, \mathtt{Dealing})$, sample $s_i \xleftarrow{\$} \mathbb{Z}_q$ and send $(\mathsf{sid}_i, \mathtt{Dealing}, s_i)$ to $\mathcal{F}_{\mathsf{VSS}}$ where $\mathsf{sid}_i \leftarrow (\mathsf{sid}, i)$. Then:

- For all $j \in [n] \setminus \{i\}$ receive $(\mathsf{sid}_j, s_{ji}, \bar{g}^{s_{j1}}, \ldots, \bar{g}^{s_{jn}})$ or $\perp$ from $\mathcal{F}_{\mathsf{VSS}}$. Let U denote the set of $j \in [n]$ for which $\perp$ is not returned. Also append i to U.
- Compute the secret key share $x_i \leftarrow \sum_{j \in U} s_{ji}$ and individual public key $y_i \leftarrow \prod_{j \in U} \bar{g}^{s_{ji}}$.
- Finally compute the system public key y by Lagrange in the exponent from $(y_1, \ldots, y_n)$.
- Store $(\mathsf{sid}, \{y_i\}_{i \in [n]}, \{\bar{h}_{ij}\}_{i,j \in [n]})$ where $\bar{h}_{ij} = \bar{g}_i^{s_j}$.

Public Verifying. Any party $P \in \{P_1, \ldots, P_n, P_v\}$ upon input $(\mathsf{sid}, \mathtt{Verify}, \{y_i\}_{i \in [n]})$:

- Look up for $(\{y_i\}_{i \in [n]}, \{\bar{h}_{ij}\}_{i,j \in [n]})$.
- For all $i \in [n]$ Send $(\mathsf{sid}_i, \mathtt{Verify}, \{\bar{h}_{ij}\}_{j \in [n]})$ to $\mathcal{F}_{\mathsf{VSS}}$.
- If there is at least one $\mathtt{Dealing\text{-}Succeeded}$ response, then output 1, otherwise output 0.

Figure 6: Our DKG protocol cgDKG in $\mathcal{F}_{\mathsf{VSS}}$-hybrid

users, it is 297.82Kb (vs 1.66MB for GrothVSS).

Computation Overhead. Figure 8 shows the time taken by the dealer and the receiver in the cgVSS protocol. The dealer's computation time includes the time to generate the multi-receiver ciphertext and the NIZK proof of correctness whereas a receiver's computation time includes the decryption time and the time for proof verification. We use multi-exponentiation to compute the product of multiple exponentiated values in the generation and verification procedures of the proof of correct sharing. For a 100 party system, the dealer takes 117 msec for generating the ciphertext and 230 msec to generate the proof, whereas for a 150 party system, it takes 176 msec for encryption and 312msec for the proof generation. The decryption takes 38 msec, while the proof verification takes 661 msec for a 100 user system and 1.18 sec for a 150 party system (the decryption time stays the same irrespective of the number of parties). Figure 8b shows the total receiver times taken by the party to verify the sharing and decrypt their shares.

In GrothVSS, to encrypt a share value, (assume) each share is divided into 24 chunks and encrypted individually. The ElGamal encryption constitutes two group elements; however, since the randomness is re-used across different users, the total number of elements for randomness is 24,

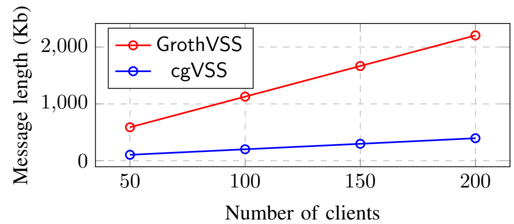
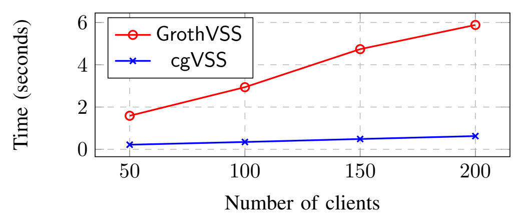
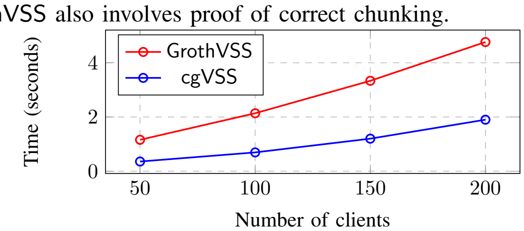


Figure 7: Comparison of broadcast (dealing) message length where $n = 2t + 1$. cgVSS dealing consists of encryptions and proof of correct sharing, while GrothVSS also consists of proof of correct chunking.



(a) Comparison of dealer times. cgVSS dealer time consists of times for encryption and proof of correct sharing, while GrothVSS also involves proof of correct chunking.



(b) Comparison of receiver times. cgVSS receiver time consists of decryption time and verification of correct sharing, while GrothVSS also involves verification of correct chunking.

Figure 8: Comparison of dealer and receiver times for cgVSS and GrothVSS.

amounting to $24 * 381 = 9144$ bits. For n users, the total bit-length of ciphertexts is $9144 \cdot (n+1)$, including the random values. The dealer also commits to the t coefficients of the polynomial, which amount to $257 \cdot t$. The dealer generates the NIZK proof of correctness of sharing, which constitutes 3 multiplicative group elements and two scalars of 381 bits each. GrothVSS uses the BLS12-381 curve, and hence the elements are 381 bits each. The dealer also generates proof of the correctness of chunking by showing that each 'chunk' is in a small range of values. For this, an approximate range proof is employed where the dealer forwards a set of elements, including $2\ell + 2$ group elements for a parameter ℓ and $\ell + n + 1$ masked values of the chunks. Taking a conservative estimate of 32 bits for the masked chunk value summations, we have the total bit-length of the approximate range proof to be $(2\ell + 2) \cdot 381 + (\ell + n + 1) \cdot 32$.

The total bit-length of the broadcast message (see Figure 7) for GrothVSS for a 150 party is 1.66Mb. This indicates a 5.6x improvement in total broadcast message length while using cgVSS when compared to GrothVSS for a 150 party system. The comparison also indicates that the broadcast message length increases slower in cgVSS when compared to GrothVSS. In GrothVSS, for a 100 party system, the dealer takes 194 msec for generating ciphertexts, 74 msec for generating the proof of correct sharing, and 2.67 sec for generating proof of correct chunking; the corresponding numbers for a 150 party system are 282 msec, 110 msec and 4.34 sec respectively. For decrypting their share, each receiver decrypts all the corresponding chunks, which amounts to 338 msec. For verification, in a 100 party system, the receiver takes 389 msec for proof of correct sharing and 1.485 sec for proof of correct chunking; for a 150 party system, the receiver takes 895 msec for proof of correct sharing and 2.19 sec for the proof of correct chunking.

To also give a sense of how the scheme compares to other existing state-of-the-art PVSS schemes, we briefly mention the timing reported by Gentry et al. [40] for their LWE-based PVSS scheme. We present their reported numbers, though their performance has been evaluated on a more powerful machine (with 32 cores and 250GB RAM) compared to our benchmarks (10 core 16GB RAM machine). For 128 parties, their system takes 4.2 sec for generating ciphertexts and 22.9 sec for generating the proof of correctness of sharing totaling 27.1 sec of dealer time, whereas for 256 parties, the total dealer time is 28.1 sec. The receiver takes 1.4 msec to decrypt and 15.3 sec to verify the dealing totaling 15.301 sec. The total receiver time for 256 parties is 15.901 sec.

End-to-end Protocol Analysis. We realize the cgDKG and GrothDKG protocols and compare them. Figure 9 compares the time taken by each node in each DKG instance; it is the time taken from the start of dealing to the computation of the system public key after verifying $t+1$ valid dealings. The nodes publish the encrypted shares and commitments using the HotStuff [83] SMR. The SMR is realized separately from the DKG nodes, which communicate with the SMR through RPC calls. For 10 nodes, GrothDKG takes 3.434 seconds, with cgDKG taking 2.656 seconds. For a 50 node network, GrothDKG takes 43.058 seconds while cgDKGtakes 17.950 seconds. From Figure 8 and Figure 9, it can be observed that the SMR takes significant time in the overall end-to-end scenario, and the optimizations in SMR usage (block rate, dummy blocks etc) would improve the performance.

In summary, our performance analysis demonstrates that cgDKG is efficient and continues to perform significantly better than GrothDKG with an increasing number of nodes in the system. Moreover, as we improve class-group implementation in the future, we expect the performance of our cgDKG to improve further.

8. Conclusion

In this work we propose a class-group based NI-VSS protocol. In particular, we show how the unique structures provided by class-groups can be used to achieve not only a significantly simpler protocol, but also a more efficient one. The generic transformation yields a simpler and more

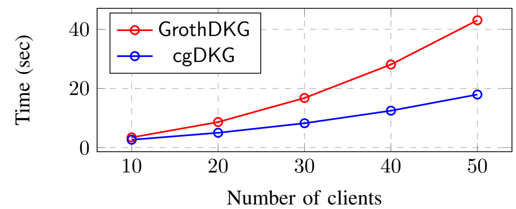


Figure 9: Comparison of time taken to perform a DKG. GrothDKG is realized using GrothVSS where each party acts as a dealer and runs an instance of GrothVSS. The times reported are aggregates of time taken from starting of dealing and computation of public key by each node, across nodes; 10 such DKG runs are aggregated.

efficient NI-DKG protocol as well. Incorporating class-group techniques to the regime of VSS/DKG is the primary contribution of this work.

Additionally, we explore and re-interpret the semantic of "public verifiability" from the literature in the context of VSS/DKG. With the growing prominence of threshold cryptography, the importance of the pubic verifiability property cannot be overstated. We provide the first formalization of a new public verifiability property (we called *strong* public verifiability to distinguish it from the prior notions), noticing its significance in specific VSS/DKG applications, and also the lack of rigorous formalization in the literature. We believe our new comprehensive and detailed formalism holds independent significance for the broader field of threshold cryptography.

References

[1] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[2] G. R. Blakley, "Safeguarding cryptographic keys," in *1979 International Workshop on Managing Requirements Knowledge (MARK)*, 1979, pp. 313–318.

[3] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, "Verifiable secret sharing and achieving simultaneity in the presence of faults," in *FOCS*, 1985, p. 383–395.

[4] D. Galindo, J. Liu, M. Ordean, and J.-M. Wong, "Fully distributed verifiable random functions and their application to decentralised random beacons," in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2021, pp. 88–102.

[5] I. Cascudo and B. David, "Scrape: Scalable randomness attested by public entities," in *ACNS*, 2017, pp. 537–556.

[6] A. Bhat, N. Shrestha, A. Kate, and K. Nayak, "Optrand: Optimistically responsive reconfigurable distributed randomness," in *NDSS*, 2023.

[7] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," in *EUROCRYPT'99*, 1999, p. 295–310.

[8] J. Groth, "Non-interactive distributed key generation and key resharing," Cryptology ePrint Archive, Paper 2021/339, 2021, https://eprint.iacr.org/2021/339.

[9] C. Komlo, I. Goldberg, and D. Stebila, "A formal treatment of distributed key generation, and new constructions," *Cryptology ePrint Archive*, 2023.

[10] K. Gurkan, P. Jovanovic, M. Maller, S. Meiklejohn, G. Stern, and A. Tomescu, "Aggregatable distributed key generation," in *Advances in Cryptology - EUROCRYPT 2021*, 2021, pp. 147–176.

[11] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme," in *Public Key Cryptography — PKC 2003*, 2002, pp. 31–46.

[12] Y. G. Desmedt, "Threshold cryptography," *European Transactions on Telecommunications*, vol. 5, no. 4, pp. 449–458, 1994.

[13] J. Groth and V. Shoup, "Design and analysis of a distributed ECDSA signing service," *IACR Cryptol. ePrint Arch.*, p. 506, 2022.

[14] A. Kate and I. Goldberg, "Distributed private-key generators for identity-based cryptography," in *SCN*, 2010, pp. 436–453.

[15] R. Cramer, I. Damgård, and U. Maurer, "General secure multi-party computation from any linear secret-sharing scheme," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2000, pp. 316–334.

[16] A. Patra, A. Choudhury, and C. Pandu Rangan, "Efficient asynchronous verifiable secret sharing and multiparty computation," *Journal of Cryptology*, vol. 28, pp. 49–109, 2015.

[17] D. Lu, T. Yurek, S. Kulshreshtha, R. Govind, A. Kate, and A. Miller, "Honeybadgermpc and asynchromix: Practical asynchronous MPC and its application to anonymous communication," in *ACM CCS*, 2019, pp. 887–903.

[18] L. Braun, I. Damgård, and C. Orlandi, "Secure multiparty computation from threshold encryption based on class groups," To appear at Crypto 2023, 2022. [Online]. Available: https://eprint.iacr.org/2022/1437

[19] E. V. Mangipudi, U. Desai, M. Minaei, M. Mondal, and A. Kate, "Uncovering impact of mental models towards adoption of multi-device crypto-wallets," Cryptology ePrint Archive, Paper 2022/075, 2022, https://eprint.iacr.org/2022/075.

[20] I. Abraham, D. Malkhi, K. Nayak, L. Ren, and M. Yin, "Sync hotstuff: Simple and practical synchronous state machine replication," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 106–118.

[21] DRand, "DRand - Distributed randomness beacon - Cryptography," https://drand.love/docs/cryptography/#setup-phase.

[22] "Dfinity-Internet Computer for Geeks," https://internetcomputer.org/whitepaper.pdf.

[23] D. Galindo, J. Liu, M. Ordean, and J.-M. Wong, "Fully distributed verifiable random functions and their application to decentralised random beacons," in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2021, pp. 88–102.

[24] C. Gentry, S. Halevi, H. Krawczyk, B. Magri, J. B. Nielsen, T. Rabin, and S. Yakoubov, "Yoso: You only speak once: Secure mpc with stateless ephemeral roles," in *Advances in Cryptology — Crypto*, 2021, pp. 64–93.

[25] B. Schoenmakers, "A simple publicly verifiable secret sharing scheme and its application to electronic," in *Advances in Cryptology—CRYPTO*, 1999, pp. 148–164.

[26] A. Scafuro, L. Siniscalchi, and I. Visconti, "Publicly verifiable proofs from blockchains," in *Public-Key Cryptography - PKC 2019*, 2019, pp. 374–401.

[27] M. Backes, A. Kate, and A. Patra, "Computational verifiable secret sharing revisited," in *Advances in Cryptology—ASIACRYPT*, 2011, pp. 590–609.

[28] T. P. Pedersen, "A threshold cryptosystem without a trusted party," in *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, 1991, pp. 522–526.

[29] W. Neji, K. Blibech, and N. Ben Rajeb, "Distributed key generation protocol with a new complaint management strategy," *Security and Communication Networks*, vol. 9, no. 17, pp. 4585–4595, 2016.

[30] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, 1987, pp. 427–438.

[31] A. Kate, G. M. Zaverucha, and I. Goldberg, "Polynomial commitments," *Tech. Rep*, 2010.

[32] R. Gennaro, M. O. Rabin, and T. Rabin, "Simplified vss and fast-track multiparty computations with applications to threshold cryptography," in *ACM PODC*, 1998, p. 101–111.

[33] A. Chandramouli, A. Choudhury, and A. Patra, "A survey on perfectly secure verifiable secret-sharing," vol. 54, no. 11s, 2022. [Online]. Available: https://doi.org/10.1145/3512344

[34] S. Das, V. Krishnan, I. M. Isaac, and L. Ren, "Spurt: Scalable distributed randomness beacon with transparent setup," in *43rd IEEE Symposium on Security and Privacy*. IEEE, 2022, pp. 2502–2517.

[35] S. Das, Z. Xiang, A. Tomescu, A. Spiegelman, B. Pinkas, and L. Ren, "A new paradigm for verifiable secret sharing," Cryptology ePrint Archive, Paper 2023/1196, 2023, https://eprint.iacr.org/2023/1196. [Online]. Available: https://eprint.iacr.org/2023/1196

[36] M. Stadler, "Publicly verifiable secret sharing," in *Advances in Cryptology—EUROCRYPT*, 1996, pp. 190–199.

[37] C. Gentry, S. Halevi, and V. Lyubashevsky, "Practical non-interactive publicly verifiable secret sharing with thousands of parties," in *Advances in Cryptology—EUROCRYPT*, 2022, pp. 458–487.

[38] 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, pp. 98–108.

[39] S. Heidarvand and J. L. Villar, "Public verifiability from pairings in secret sharing schemes," in *Selected Areas in Cryptography*, R. M. Avanzi, L. Keliher, and F. Sica, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 294–308.

[40] C. Gentry, S. Halevi, and V. Lyubashevsky, "Practical non-interactive publicly verifiable secret sharing with thousands of parties," in *Advances in Cryptology–EUROCRYPT 2022*. Springer, 2022, pp. 458–487.

[41] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *2018 IEEE symposium on security and privacy (SP)*. IEEE, 2018, pp. 315–334.

[42] G. Castagnos and F. Laguillaumie, "Linearly homomorphic encryption from ddh," in *Cryptographers' Track at the RSA Conference*. Springer, 2015, pp. 487–505.

[43] S. A. K. Thyagarajan, G. Castagnos, F. Laguillaumie, and G. Malavolta, "Efficient cca timed commitments in class groups," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '21, 2021.

[44] R. Bacho and J. Loss, "On the adaptive security of the threshold bls signature scheme," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 193–207.

[45] S. Kanjalkar, Y. Zhang, S. Gandlur, and A. Miller, "Publicly auditable mpc-as-a-service with succinct verification and universal setup," 2021.

[46] A. Ozdemir and D. Boneh, "Experimenting with collaborative zk-snarks: Zero-knowledge proofs for distributed secrets," Cryptology ePrint Archive, Paper 2021/1530, 2021, https://eprint.iacr.org/2021/1530.

[47] D. Boneh and V. Shoup, "A graduate course in applied cryptography." [Online]. Available: https://toc.cryptobook.us/book.pdf

[48] S. Jarecki, A. Kiayias, H. Krawczyk, and J. Xu, "TOPPSS: cost-minimal password-protected secret sharing based on threshold OPRF," in *Applied Cryptography and Network Security (ACNS) 2017*.

[49] D. Wikström, "Universally composable DKG with linear number of exponentiations," in *Security in Communication Networks, 4th International Conference, SCN 2004*, 2004.

[50] J. Katz, "Round optimal robust distributed key generation," Cryptology ePrint Archive, Paper 2023/1094, 2023, https://eprint.iacr.org/2023/1094. [Online]. Available: https://eprint.iacr.org/2023/1094

[51] E. Fujisaki and T. Okamoto, "A practical and provably secure scheme for publicly verifiable secret sharing and its applications," in *Advances in Cryptology — EUROCRYPT'98*, 1998, pp. 32–46.

[52] F. Boudot and J. Traoré, "Efficient publicly verifiable secret sharing schemes with fast or delayed recovery," in *Information and Communication Security*, 1999, pp. 87–102.

[53] ——, "Efficient publicly verifiable secret sharing schemes with fast or delayed recovery," in *Information and Communication Security: Second International Conference, ICICS'99, Sydney, Australia, November 9-11, 1999. Proceedings 2*, 1999, pp. 87–102.

[54] A. Young and M. Yung, "A pvss as hard as discrete log and shareholder separability," in *Public Key Cryptography*, 2001, pp. 287–299.

[55] P.-A. Fouque and J. Stern, "One round threshold discrete-log key generation without private channels," in *Public Key Cryptography*, 2001, pp. 300–316.

[56] J. Camenisch and V. Shoup, "Practical verifiable encryption and decryption of discrete logarithms," in *Annual International Cryptology Conference*. Springer, 2003, pp. 126–144.

[57] A. Ruiz and J. L. Villar, "Publicly verifiable secret sharing from paillier's cryptosystem," 2005.

[58] T.-Y. Wu and Y.-M. Tseng, "A pairing-based publicly verifiable secret sharing scheme," *Journal of systems science and complexity*, vol. 24, no. 1, pp. 186–194, 2011.

[59] R. D'Souza, D. Jao, I. Mironov, and O. Pandey, "Publicly verifiable secret sharing for cloud-based key management," in *INDOCRYPT 2011: 12th International Conference on Cryptology 2011*. Springer, 2011, pp. 290–309.

[60] R. Canetti, R. Gennaro, S. Goldfeder, N. Makriyannis, and U. Peled, "Uc non-interactive, proactive, threshold ecdsa with identifiable aborts," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2020.

[61] E. Kokoris Kogias, D. Malkhi, and A. Spiegelman, "Asynchronous distributed key generation for computationally-secure randomness, consensus, and threshold signatures." New York, NY, USA: Association for Computing Machinery, 2020.

[62] I. Abraham, P. Jovanovic, M. Maller, S. Meiklejohn, G. Stern, and A. Tomescu, "Reaching consensus for asynchronous distributed key generation," in *PODC'21*, 2021, pp. 363–373.

[63] I. Abraham, P. Jovanovic, M. Maller, S. Meiklejohn, and G. Stern, "Bingo: Adaptively secure packed asynchronous verifiable secret sharing and asynchronous distributed key generation," *Cryptology ePrint Archive*, 2022.

[64] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *International conference on the theory and application of cryptology and information security*. Springer, 2001, pp. 514–532.

[65] Supra Research, "Supra VRF Service," https://supraoracles.com/docs/SupraOracles-VRF-Service-Whitepaper.pdf.

[66] Chainlink, "Chainlink VRF: On-Chain Verifiable Randomness." https://blog.chain.link/chainlink-vrf-on-chain-verifiable-randomness/.

[67] B. Wesolowski, "Efficient verifiable delay functions," in *Advances in Cryptology–EUROCRYPT 2019*, 2019, pp. 379–407.

[68] G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker, "Two-party ecdsa from hash proof systems and efficient instantiations," in *Annual International Cryptology Conference*. Springer, 2019, pp. 191–221.

[69] G. Castagnos, F. Laguillaumie, and I. Tucker, "Threshold linearly homomorphic encryption on bfz/2$^{\text{kbfz}}$," in *Advances in Cryptology - ASIACRYPT 2022*, 2022, pp. 99–129.

[70] G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker, "Bandwidth-efficient threshold ec-dsa," in *23rd IACR International Conference on Practice and Theory of Public-Key Cryptography*, 2020, pp. 266–296.

[71] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*. IEEE Computer Society, 2001, pp. 136–145.

[72] R. Canetti, A. Cohen, and Y. Lindell, "A simpler variant of universally composable security for standard multiparty computation," in *Advances in Cryptology – CRYPTO 2015*, R. Gennaro and M. Robshaw, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 3–22.

[73] R. Canetti, "Universally composable signature, certification, and authentication," in *Proceedings. 17th IEEE Computer Security Foundations Workshop, 2004.*, 2004, pp. 219–233.

[74] I. Damgård and E. Fujisaki, "A statistically-hiding integer commitment scheme based on groups with hidden order," Y. Zheng, Ed., 2002. [Online]. Available: https://doi.org/10.1007/3-540-36178-2_8

[75] M. Bellare, A. Boldyreva, and J. Staddon, "Randomness re-use in multi-recipient encryption schemeas," in *Public Key Cryptography—PKC 2003: 6th International Workshop on Practice and Theory in Public Key Cryptography Miami, FL, USA, January 6–8, 2003 Proceedings 6*. Springer, 2002, pp. 85–99.

[76] M. Bellare, A. Boldyreva, K. Kurosawa, and J. Staddon, "Multirecipient encryption schemes: How to save on bandwidth and computation without sacrificing security," *IEEE Transactions on Information Theory*, vol. 53, no. 11, pp. 3927–3943, 2007.

[77] B. Schoenmakers and M. Veeningen, "Universally verifiable multiparty computation from threshold homomorphic cryptosystems," in *Applied Cryptography and Network Security (ACNS)*, 2015, pp. 3–22.

[78] C. Baum, I. Damgård, and C. Orlandi, "Publicly auditable secure multi-party computation," in *Security and Cryptography for Networks (SCN)*, 2014, pp. 175–196.

[79] C. Komlo, I. Goldberg, and D. Stebila, "A formal treatment of distributed key generation, and new constructions," *IACR Cryptol. ePrint Arch.*, p. 292, 2023. [Online]. Available: https://eprint.iacr.org/2023/292

[80] A. Chandramouli, A. Choudhury, and A. Patra, "A survey on perfectly secure verifiable secret-sharing," *ACM Comput. Surv.*, vol. 54, no. 11s, pp. 232:1–232:36, 2022.

[81] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*. IEEE, 1987, pp. 427–438.

[82] C. Bouvier, G. Castagnos, L. Imbert, and F. Laguillaumie, "I want to ride my bicycl: Bicycl implements cryptography in class groups," Cryptology ePrint Archive, Paper 2022/1466, 2022, https://eprint.iacr.org/2022/1466.

[83] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: Bft consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, 2019, pp. 347–356.

[84] R. Bacho and J. Loss, "On the adaptive security of the threshold bls signature scheme," in *ACM CCS*, 2022, p. 193–207.

Appendix A. Security of NIZK proof of exponent over class group

We recall the construction from Section 3.1. We restate the theorem.

Theorem 7. *For any $\lambda, \lambda_{\mathsf{st}} \in \mathbb{N}$ and any modulus $q \in \mathbb{Z}$, for a correctly generated class group parameters $pp_{\mathsf{CG}} \leftarrow \mathsf{CGE.KeyGen}(1^\lambda, 1^\lambda_{\mathsf{st}}, q)$ as long as the low order assumptions (Def. 3) and the strong root assumption (Def. 4) holds over the class group $\widehat{G}$, the NIZK proof system described above is secure argument of knowledge in the random oracle model.*

Proof. Since completeness is immediate. We focus on statistical zero-knowledge and knowledge of argument in order.

Statistical Zero-knowledge. We build a simulator $\mathcal{S}$ as follows:

1) Input: $(g_q, h) \in G^q \times G^q$.
2) Sample a uniform random $s \xleftarrow{\$} [2^{\lambda_{\mathsf{st}}} \cdot \mathcal{B} \cdot |\mathcal{D}_q|]$.
3) Sample a uniform random $c \xleftarrow{\$} \mathcal{B}$
4) Compute $a \leftarrow g_q^s \cdot (h^c)^{-1}$.
5) Program the random oracle $H(g_q, h, a) = c$.
6) Output $\pi \leftarrow (c, s)$

Now note that, the proof (c, s) satisfies the verification test, because (i) s is in the correct range; (ii) the equation in Step 4 holds and (iii) the random oracle is correctly programmed in Step 5. Note that, if a malicious verifies makes Q_H many RO queries, then the probability of successfully obtaining a correct input-output pair is bounded by $Q_H^2/2|\mathcal{B}|$. Setting $Q_H^2 = 2^{\lambda - \lambda_{\mathsf{st}}}$ this probability is $\mathsf{negl}(\lambda_{\mathsf{st}})$.

Again, if the above event does not happen then the only event when the simulated s and an actual s produced by the prover does not match when $2^{\lambda_{\mathsf{st}}} \cdot |\mathcal{D}_q| \cdot \mathcal{B} \geq r > 2^{\lambda_{\mathsf{st}}-1} \cdot |\mathcal{D}_q| \cdot \mathcal{B}$. This happens with probability $2^{-\lambda_{\mathsf{st}}}$. So the statistical distance btween the simulated and real proof is bounded by $\mathsf{negl}(\lambda_{\mathsf{st}})$ as required.

Knowledge of Argument. We can use the low order assumption and strong root assumption to argue knowledge of argument similar to prior works [43], [70], [74]. The idea is to use the standard forking/rewinding technique to obtain two challenges c, c' for the same a, and subsequently two different s, s' such that we have: $g_q^s \cdot h^{-c} = g_q^{s'} \cdot h^{-c'}$. Let $d = \gcd(s - s', c - c')$. Then we define

$$\gamma = g_q^{\frac{s-s'}{d}} \cdot (h^{-1})^{\frac{c-c'}{d}}$$

Clearly, $\gamma^d = 1$. Now there are two cases:

- Case-1: $\gamma \neq 1$. In this case, we have an element $\gamma \in \widehat{G}$ which has order $d < c - c' < \mathcal{B}$. That implies a break of $\mathcal{B}$-low order assumption. So the probability of this case is negligible.
- Case-2: $\gamma = 1$. In this case we have $g_q^{\frac{s-s'}{d}} = h^{\frac{c-c'}{d}}$. Define $f = \frac{c-c'}{d}$ Then there are two sub-cases:
  - Case-2.(a): $f \neq 2^\rho$ for any integer ρ. In this case we can write using Euclidean GCD: $d = \alpha(s - s') + \beta(c - c')$ for integers α, β. Then we have:
$$\begin{aligned} g_q^d &= g_q^{\alpha(s-s')+\beta(c-c')} \\ &= h^{f d \alpha} g_q^{f d \beta} \end{aligned}$$
    This means we can write:
$$g_q = (h^\alpha g_q^\beta)^f$$
    So, for $Y = g_q$ we get $X = h^\alpha g_q^\beta$ and f is not a power of two – this solves the strong root assumption over $\widehat{G}$. We note that, g_q may not be a random element in $\widehat{G}^q$, but in G^q. However, in the reduction we can choose G to be a random power of $\widehat{G}$ to resolve this, since we know that the order of G divides the order of $\widehat{G}$ (while both remains unknown) this is possible.
  - Case-2.(b): $f = 2^\rho$. In this case let $w = \frac{s-s'}{d} \in \mathbb{Z}$ (since d is the gcd of $s - s'$ and $c - c'$). Then we have $h^f = g_q^w$. However, since the group G^q has an odd order (the order s divides $\widehat{s}$), the integer $f = 2^\rho$ must divide w, otherwise we would have an element that has an even order. Therefore, we can write $g_q^{\frac{w}{f}} = h$, where $\frac{w}{f} \in \mathbb{Z}$ which is a witness. Note that the witness is in the range $[(2^\lambda_{\mathsf{st}} + 1) \cdot |\mathcal{D}_q| \cdot \mathcal{B}]$ instead of the original range $\mathcal{B}$. But that suffices as they are equal modulo the order of G^q.

This concludes the proof for knowledge of argument. □

Appendix B. Security of the multi-receiver encryption scheme

We provide detailed proof of our class group-based multi-receiver encryption scheme from the HSM assumption. We restate the theorem below:

Theorem 8. *For any $\lambda, \lambda_{\mathsf{st}}$ and any modulus $q \in \mathbb{Z}$ let $(q, \lambda, \lambda_{\mathsf{st}}, \bar{s}, f, g_q, \widehat{G}, F, \mathcal{D}, \mathcal{D}_q; \rho) \leftarrow \mathsf{CG.ParamGen}(1^\lambda, 1^\lambda_{\mathsf{st}}, q)$ be a set of correctly generated class group parameters. Then, for that set of parameters as long as hard subgroup assumptions (Def. 2) holds, the above multi-receiver encryption scheme is secure according to Definition 5 for any $n, t \in \mathbb{N}$ such that $n > t$.*

Proof. The proof idea basically follows footsteps of the proof for the linearly homomorphic encryption scheme provided in [69], with adequate changes for the multi-receiver case. For simplicity of exposition, we assume that $n = 2$ and $t = 1$ – extending to the general case is straightforward. Suppose that $\mathcal{A}$ corrupts sk_2, and outputs two message vectors $\vec{m}_0 = (m_1, m_2)$ and $\vec{m}_1 = (m_1', m_2)$, where the second element is the same by condition. Let us call the indistinguishability game with $b = 0$: Game_0 and with $b = 1$: Game_1. We show that using the hard subgroup assumption (Def. 2) we can move from Game_0 to a mental game (via a sequence of hybrids) where the message m_1 is statistically hidden. A similar sequence of hybrid can be constructed to move from Game_1 to the same mental game. To start with first note that in Game_0 the adversary's view can be expressed as:

$$sk_2, h_1 = g_q^{sk_1}, h_2 = g_q^{sk_2}; R = g_q^r; E_1 = f^{m_1} h_1^r; E_2 = f^{m_2} h_2^r$$
$$\text{where} \quad sk_1, sk_2, r \xleftarrow{\$} \mathcal{D}_q$$

In Hyb_1 we can write $E_1 = f^{m_1} R^{sk_1}$ and $E_2 = f^{m_1} R^{sk_2}$, and clearly Hyb_1 and Game_0 are identically distributed.

In the next hybrid Hyb_2, sk_1 is sampled as $sk_1 \xleftarrow{\$} \mathcal{D}$ instead of $\mathcal{D}_q$. However, since the adversary is given sk_1 only in the exponents of g_q and R, both of which are in G^q, information-theoretically $\mathcal{A}$ only sees $sk_1 \bmod s$. Also drawing $sk_1 \xleftarrow{\$} \mathcal{D}_q$ induces a distribution with is $2^{-\lambda_{\mathsf{st}}}$ close to the uniform distribution over $\mathbb{Z}_s$ in the exponent and similarly drawing $sk_1 \xleftarrow{\$} \mathcal{D}$ induces a distribution with is $2^{-\lambda_{\mathsf{st}}}$ close to the uniform distribution over $\mathbb{Z}_{qs}$ in the exponent. Therefore, we can conclude that the statistical distance between Hyb_1 and Hyb_2 is bounded by $2^{-\lambda_{\mathsf{st}}+1}$.

In Hyb_3 we change R to $R = f^u g_q^r$ for $u \xleftarrow{\$} \mathbb{Z}_q$. Now we can argue that Hyb_2 is indistinguishable from Hyb_3 as long as the hard sub-group problem (Def. 2) holds. The reduction simply plugs in the challenge value into R as there is no dependency on any of exponent. In particular, the adverasry's view is computed as:

$$sk_2, h_1 = g_q^{sk_1}, h_2 = g_q^{sk_2}; R; E_1 = f^{m_1} R^{sk_1}; E_2 = f^{m_2} R^{sk_2}$$
$$\text{where} \quad sk_1 \xleftarrow{\$} \mathcal{D}; sk_2 \xleftarrow{\$} \mathcal{D}_q$$

Clearly when $R = g_q^r$ Hyb_2 is simulated, and when $R = f^u g_q^r$ then Hyb_3 is simulated.

In Hyb_3 we note that the adversary receives $E_1 = f^{m_1 + u \cdot sk_1} h_1^r$. Given adversary's view information theoretically h_1^r is fixed. Hence an unbounded adversary can obtain $m_1 + u \cdot sk_1 \bmod q$ (since the order of $\langle f \rangle = F$ is q). Now, note that in Hyb_2, we change the sampling of sk_1 from a distribution, which is $2^{-\lambda_{\mathsf{st}}}$ close $\mathbb{Z}_{q.s}$. Now, $sk_1 \bmod qs$ can be written as $(sk_1 \bmod q, sk_1 \bmod s)$ using Chinese remainder theorem – in that $sk_1 \bmod q$ is uniform random in $\mathbb{Z}_q$ as long as $sk_1 \bmod qs$ is uniform random in $\mathbb{Z}_{qs}$. Furthermore, $sk_1 \bmod q$ is independent of $sk_1 \bmod s$. Therefore, although an unbounded adversary obtains a fixed $sk_1 \bmod s$ from the public key $h_1 = g_q^{sk_1}$ ($\langle g_q \rangle = G^q$ has order s), $sk_1 \bmod q$ is indeed $s^{-\lambda_{\mathsf{st}}}$ close to uniformly random value in $\mathbb{Z}_q$. So, $m_1 + u \cdot sk_1 \bmod q$ is $2^{-\lambda_{\mathsf{st}}}$ close to uniform random value in $\mathbb{Z}_q$. Similarly we can arrive at Hyb_3 starting from Game_1. Hence we can conclude that Game_0 and Game_1 is computationally indistinguishable – this concludes the proof.

□

Appendix C. Security of NIZK proof of correctness of secret-sharing

Theorem 9. *For any security parameters $\lambda, \lambda_{\mathsf{st}} \in \mathbb{N}$ and any modulus $q \in \mathbb{N}$, our NIZK construction described in Fig. 1 is a secure proof system (as described in Def. 1) in the random oracle model.*

Proof. We prove *perfect completeness*, statistical soundness in ROM and statistical zero-knowledge in ROM.

Completeness. The completeness can be seen from checking the verification equations:

- $W \cdot R^{\gamma'} = g^{\rho + r\gamma'} = g_q^{z_r}$;
- $X \cdot (\prod_{j=0}^{t} A_j^{\sum_{i=1}^{n} i^k \gamma^j})^{\gamma'}$
  $= X \cdot \left(A_0^{(\gamma+\gamma^2+\ldots)} \cdot A_1^{(\gamma+2\gamma^2+\ldots)} \cdot A_2^{(\gamma+2^2\gamma^2+\ldots)} \ldots\right)^{\gamma'}$
  $= X \cdot \left(\bar{g}^{a_0(\gamma+\gamma^2+\ldots)} \cdot \bar{g}^{a_1(\gamma+2\gamma^2+\ldots)} \cdot \bar{g}^{a_2(\gamma+2^2\gamma^2+\ldots)} \ldots\right)^{\gamma'}$
  $= X \cdot \left(\bar{g}^{(a_0+a_1+\ldots)\gamma + (a_0+2a_1+2^2a_2+\ldots)\gamma^2+\ldots}\right)^{\gamma'}$
  $= X \cdot \left(\bar{g}^{s_1\gamma + s_2\gamma^2+\ldots}\right)^{\gamma'} = \bar{g}^{\alpha + \gamma' \sum_{i=1}^{n} s_i \gamma^i} = \bar{g}^{z_s}$;
- $(\prod_{i=1}^{n} E_i^{\gamma^i})^{\gamma'} \cdot Y$
  $= \left(f^{\gamma'(\sum_{i=1}^{n} s_i\gamma^i)} \cdot \prod_{i=1}^{n} h_i^{r\gamma'\gamma^i}\right) \cdot \left(f^{\alpha} \cdot \prod_{i=1}^{n} h_i^{\rho\gamma^i}\right)$
  $= f^{\alpha + \gamma' \sum_{i=1}^{n} s_i \gamma^i} \cdot \prod_i h_i^{(r\gamma'+\rho)\gamma^i} = f^{z_s} \cdot \prod_{i=1}^{n} (h_i^{\gamma^i})^{z_r}$

Statistical Soundness. The soundness argument is essentially the same as the one given by Groth [8](As mentioned in Groth's paper, we do not actually need simulation soundness.) but adjusted to our class group setting. The soundness holds unconditionally with overwhelming probability ($\geq 1 - \mathsf{negl}(\lambda_{\mathsf{st}})$) in the random oracle model with appropriately chosen λ_{st}.

In particular, we consider an unbounded adversary, which can, however, make only bounded number of RO queries – we assume it makes Q_H queries to H and $Q_{H'}$ queries to H'. This adversary attempts to produce a "bad" protocol instance $\{h_i, E_i, A_j, R\}_{i \in [n], j \in [t]}$ which is not in $\mathfrak{R}_{\mathsf{CS}}$. Let us elaborate what that means. First, note that the DLog commitments $A_j = \bar{g}^{a_j}$ are perfectly binding for the coefficients $a_j \in \mathbb{Z}_q$ of the hidden polynomial P. Let $P(i) = s_i$ for all $i \in [n]$. Furthermore $R = \bar{g}_q^r$ information theoretically fixes $r \in \mathbb{Z}_s$. Also suppose each E_i has the form $f^{\tilde{s}_i} h_i^r$. Therefore, a "bad' instance must have at least one "bad" $E_i = f^{\tilde{s}_i} h_i^r$ such that $\tilde{s}_i \neq s_i \in \mathbb{Z}_q$. Now if the verification passes, that means the proof $\pi_{\mathsf{CS}} \leftarrow (W, X, Y, z_r, z_s)$ is well-formed, which means it satisfies the three verification equations. So, the only way the unbounded adversary wins are in the following three events:

- Event_1: The adversary predicts γ' correctly before fixing W, X, Y. If this is possible, then the adversary can easily choose uniform random z_r, z_s in the correct range plus $\gamma = H(\mathsf{inst})$. From these values it can easily compute X, Y, Z from the verification equations, such that they all satisfy.
- Event_2: The adversary manages to find a γ such that $\sum_{i=1}^{n} \tilde{s}_i \gamma^i = \sum_{i=1} s_i \gamma^i$ even when $\tilde{s}_j \neq s_j$ for (possibly more that one) j. In this case, if the first two equations verify (which does not depend on this fact), then by this fact the third equation also verifies. So this constitutes a break of the soundness.
- Event_3: In this event $\sum_{i=1}^{n} \tilde{s}_i \gamma^i \neq \sum_{i=1} a_i \gamma^i$, yet all three equations verify correctly.

Now, we show that:

$$\begin{aligned}\Pr[\mathsf{Event}_1 \vee \mathsf{Event}_2 \vee \mathsf{Event}_3] &\\ \leq \Pr[\mathsf{Event}_1] + \Pr[\mathsf{Event}_2] + \Pr[\mathsf{Event}_3] &\\ \leq 3\Pr[\mathsf{Event}_1] + 2\Pr[\mathsf{Event}_2 \mid \neg\mathsf{Event}_1] &\\ + \Pr[\mathsf{Event}_3 \mid \neg(\mathsf{Event}_1 \vee \mathsf{Event}_2)] \leq \mathsf{negl}(\lambda_{\mathsf{st}}) &\end{aligned}$$

where the first inequality follows from a union bound, the second one from simple partitioning and the third one from the three lemmas we prove next.

Lemma 10. *As long as the adversary makes Q many queries to the RO such that $Q^2/2q$ is $\mathsf{negl}(\lambda_{\mathsf{st}})$, we have that*

$$\Pr[\mathsf{Event}_1] \le \mathsf{negl}(\lambda_{\mathsf{st}})$$

Proof. Assume that the adversary makes at most $Q_{H'}$ queries to H', the probability with which the adversary correctly predicts a corret γ' is upper bounded by $Q_{H'}^2/2q$. For $Q^2/2q = \mathsf{negl}(\lambda_{\mathsf{st}})$ we can set $Q^2 = O(2^{\lambda-\lambda_{\mathsf{st}}})$, since $q = O(2^\lambda)$. □

Lemma 11. *As long as γ is chosen uniformly at random in $\mathbb{Z}_q$, we have that*

$$\Pr[\mathsf{Event}_2 \mid \neg\mathsf{Event}_1] \le \mathsf{negl}(\lambda)$$

Proof. First note that, since Event_1 does not happen, γ' is computed legitimately after computing γ, X, Y, Z. Then, by definition if Event_2 happens, we have that:

$$\sum_{i=1}^n \tilde{s}_i\gamma^i = \sum_{i=1}^n s_i\gamma^i$$

and there is a $\tilde{s}_j \neq s_j$. Denote for each i: $s_i^\delta = s_i - \tilde{s}_i \in \mathbb{Z}_q$. So we can write:

$$\sum_{i=1}^n s_i^\delta \gamma^i = 0$$

and by the premise this n-degree polynomial $P_\delta = \sum_{i=1}^n s_i^\delta x^i$ is not identically zero. Using Schwartz-Zippel lemma we conclude that as long as γ is chosen uniformly at random from $\mathbb{Z}_q$ (which is true as we are in ROM), the probability of the the polynomial defined by $P_\delta(\gamma) = 0 \in \mathbb{Z}_q$ is at most n/q which is $\mathsf{negl}(\lambda)$. □

Lemma 12. *As long as γ' is chosen uniformly at random, we have that:*

$$\Pr[\mathsf{Event}_3 \mid \neg(\mathsf{Event}_1 \vee \mathsf{Event}_2)] \le \mathsf{negl}(\lambda)$$

Proof. In this case since Event_1 and Event_2 are not happening, we can assume that all verification equations pass even when there exists an j for which $\tilde{s}_j \neq s_j$. In particular, the first two equations ensure that $z_r = r\gamma' + \rho \bmod s$ and $z_s = \sum_{i=1}^n s_i\gamma^i + \alpha \bmod q$. However, since q and s are co-prime we can write $z_r = r\gamma' + \rho + s\xi$ over integer. Now the third equation over G (which has order qs) can be written as.

$$(\prod_{i=1}^n E_i^{\gamma^i})^{\gamma'} \cdot Y = f^{z_s} \cdot \prod_{i=1}^n (h_i^{\gamma^i})^{r\gamma'+\rho+s\xi}$$

Now, since each h_i is in G^q, which has order s, we have $h_i^s = 1$ we can re-write the equation as:

$$(\prod_{i=1}^n E_i^{\gamma^i})^{\gamma'} \cdot Y = f^{z_s} \cdot \prod_{i=1}^n (h_i^{\gamma^i})^{r\gamma'+\rho}$$

Now, expressing each E_i as $f^{\tilde{s}_i}h_i^r$ we can re-write the equation as:

$$f^{\sum_{i=1}^n \tilde{s}_i\gamma^i\gamma'} \cdot \prod_{i=1}^n (h_i^{\gamma^i})^{r\gamma'} \cdot Y = f^{z_s} \cdot \prod_{i=1}^n (h_i^{\gamma^i})^{r\gamma'+\rho}$$

Clearly, Y must be of the form $Y = f^\beta \cdot \prod_{i=1}^n h_i^{\gamma^i\rho}$ for some $\beta \in \mathbb{Z}_q$. Using the value of z_s we obtain:

$$\gamma' \sum_{i=1}^n \tilde{s}_i\gamma^i + \beta = \gamma' \sum_{i=1}^n s_i\gamma^i + \alpha \bmod q$$

Again, defining $s_i^\delta = s_i - \tilde{s}_i \bmod q$ we obtain:

$$\gamma' \sum_{i=1}^n s_i^\delta\gamma^i + \alpha - \beta = 0 \bmod q$$

Unless the last equation is identically 0, for a fixed γ the probability of this holding equation over the choice of a uniform random γ' is at most $1/q$ which is $\mathsf{negl}(\lambda)$. □

This concludes the proof. Finally note that, for example, a reasonable choice can be $q = O(2^{256})$ and $Q_H = O(2^{100})$, then the overall probability is smaller than 2^{-40} which is negligible in λ_{st} for a typical choice of $\lambda_{\mathsf{st}} = 40$.

Statistical Zero-knowledge. Following [8], we argue the statistical zero-knowledge of the proof of correct sharing in the ROM. The simulator works as follows:

1) Set $H(\mathsf{inst}) = \gamma$ where γ is uniformly at random.
2) Choose γ' uniformly at random.
3) Sample $z_r \xleftarrow{\$} [q \cdot |\mathcal{D}_q| \cdot 2^\lambda_{\mathsf{st}}]$.
4) Compute X, Y, W from the three verification equation.
5) Finally program $\gamma' = H'(W, X, Y, z_r, z_s)$.

Clearly, the verification succeeds always. However, similar to the proof of exponent, if the verifier asks a RO query on $H'(W, X, Y, z_r, z_s)$ then the simulation fails. But as long as the verifier makes a bounded number of queries, this probability can be made $\le \mathsf{negl}(\lambda_{\mathsf{st}})$ by adjusting the parameters. Finally, we note that the simulated value z_r is identically distributed to the real z_r as long as $\rho < q \cdot |\mathcal{D}_q|(2^\lambda_{\mathsf{st}} - 1)$. The probability of happening otherwise is upper bounded by $2^{-\lambda_{\mathsf{st}}}$, which is negligible in λ_{st}. □

Appendix D. Mitigating the biasing public key attack

cgDKG (and Groth's NI-DKG) suffer from the same public key biasing attack as the one presented by Gennaro et al. [7]. This is because a rushing adversary can observe the first t verified secret sharings and then perform a valid t+1st sharing to bias the public key while delaying the messages of the other honest parties in the system. The adversary can first compute the partial public of the t honest parties and choose the $t + 1^{st}$ party (which the adversary controls) to bias the public key.

To overcome this, we use an approach [29] where the knowledge of the commitments does not aid the adversary in biasing the public key. After verifying the dealings, the parties use the first set of $t+1$ verified dealers to compute their secret key share. Each party now publishes the public key computed as exponentiation of the secret key with a *different* generator $g' \in \mathbb{G}_1$ than g_1, the one used in the initial commitment phase. After computing the qualified set, each party P_k broadcasts the value $(g')^{x_k}$ along with a NIZK proof that the exponent in $(g')^{x_k}$ is the same as the one computed using the verified dealings. The parties finally compute the public key of the DKG instance as $y = \prod_{k \in T} (g')^{x_k}$, where T is the set of parties that have forwarded their public key, the set T has at least $t+1$ parties as only a maximum of t parties are corrupted by the adversary. This adds one round of communication to the DKG protocol. A previously suggested approach [7] to overcome the biasing attack is to use perfectly hiding Pedersen's commitments. These commitments are published in the initial commit phase while the public key is computed in the next phase (round) using discrete log commitments, which are published along with proof of the equality of the exponents (shared secret). This approach also needs an extra round for the parties to agree on the public key. However, the mentioned approach of using a different generator for the public key is more efficient as no blinding factors (and the corresponding exponentiations) are needed.

Appendix E. Algebraic Simulatability of $\mathcal{F}_{\mathsf{dkg}}$

In this section we argue that any DKG protocol that satisfies our definition, via the ideal functionality $\mathcal{F}_{\mathsf{dkg}}$, satisfies a static variant of *oracle-aided algebraically simulatable* DKG as defined by Bacho and Loss [84]. As a consequence, our DKG can be applied to the static settings of all applications for which their definition suffices – this includes Schnorr and BLS signature. Note that, the standard properties such as *consistency*, *correctness* and *unforgeability* are easily satisfied by our ideal functionality. The only thing that remains is to show *algebraic simulatability*.

In particular, we need to argue that for any DKG that satisfies our definition, it is possible to construct a simulator $\mathcal{S}_{\mathsf{oa}}$ such that the simulator's output is indistinguishable with an adversary $\mathcal{A}$'s view that corrupts at most t parties. Consider a cyclic group $\langle \bar{g} \rangle = \bar{G}$ of order q. Then we note that on input k values $\bar{g}^{x_1}, \bar{g}^{x_2}, \ldots, \bar{g}^{x_k}$, $\mathcal{S}_{\mathsf{oa}}$ may access a discrete log oracle (which, on input $h \in \bar{G}$, returns $x \in \mathbb{Z}_p$ such that $h = \bar{g}^x$) at most $k-1$ times. Now, let us formally present the definition of algebraic simulatability adapted from Bacho and Loss [44] in our setting of static corruption.

Definition 7 (Algebraic Simulatability)**.** *A NI-DKG protocol among n parties with threshold $t \leq 2n+1$ is said to have k-algebraic simulatability if for any PPT adversary $\mathcal{A}$ that corrupts at most t parties, there exists an algebraic PPT simulator $\mathcal{S}_{\mathsf{oa}}$ that makes $k-1$ queries to a discrete log (for base $\bar{g}$) oracle and satisfies the following properties:*

- *On input* $\mathsf{ip} = \left\{(\bar{g}^{z_1}, \ldots, \bar{g}^{z_k} \in \bar{G}), C \subset [n]\right\}$ *such that* $|C| \leq t$, *$\mathcal{S}_{\mathsf{oa}}$ simulates the honest parties for a protocol execution. At the end of the simulation, $\mathcal{S}_{\mathsf{oa}}$ outputs the public key $pk = \bar{g}^{sk}$.*
- *Let $\bar{g}_i \in \bar{G}$ denote the i-th query to the discrete log (to the base $\bar{g}$) oracle. Define the corresponding algebraic coefficient as $(\hat{a}_i, a_{0,1}, \ldots, a_{0,k})$ where $\bar{g}_i = \bar{g}^{\hat{a}_i} \cdot \prod_{i=1}^{k} (\bar{g}^{z_j})^{a_{i,j}}$. Also denote the algebraic coefficient of pk as $(\hat{a}, a_{0,1}, \ldots, a_{0,k})$. Then the following matrix, called simulatability matrix for $\mathcal{S}_{\mathsf{oa}}$ over $\mathbb{Z}_q$ is invertible:*

$$\begin{bmatrix} a_{0,1} & \cdots & a_{0,k} \\ \vdots & \cdots & \vdots \\ a_{k-1,1} & \cdots & a_{k-1,k} \end{bmatrix}$$

- *Denote by $\mathsf{View}_{\mathcal{A},pk,Real}$ the view of $\mathcal{A}$ in a real protocol execution conditioned on all honest parties outputting pk as the joint public key; and let $\mathsf{View}_{\mathcal{A},pk,\mathcal{S}_{\mathsf{oa}}(\mathsf{ip})}$ denote the view of $\mathcal{A}$ in the simulated execution when simulator $\mathcal{S}_{\mathsf{oa}}$ has input ip, conditioned on $\mathcal{S}_{\mathsf{oa}}$ outputting the same pk. Then, for all $pk \in \bar{G}$ and all input ip, $\mathsf{View}_{\mathcal{A},pk,Real}$ is computationally indistinguishable from $\mathsf{View}_{\mathcal{A},pk,\mathcal{S}_{\mathsf{oa}}(\mathsf{ip})}$.*

Now note that, if a DKG scheme satisfies our UC-definition (Fig. 5), there is a PPT simulator $\mathcal{S}$ which can simulate any given PPT $\mathcal{A}$ with overwhelming probability. Therefore, as long as we can construct $\mathcal{S}_{\mathsf{oa}}$ using $\mathcal{S}$ in a way such that $\mathcal{S}$ interacts with $\mathcal{A}$ directly while $\mathcal{S}_{\mathsf{oa}}$ plays the role of ideal functionality $\mathcal{F}_{\mathsf{dkg}}$ to $\mathcal{S}$ and this is indistinguishable from a real execution from $\mathcal{A}$'s point of view then the implication holds. For simplicity we assume that there are exactly t corruptions, that implies size of honest set $|H| = t+1$ and size of corrupt set $|C| = t$. Now we describe how $\mathcal{S}_{\mathsf{oa}}$ can use $\mathcal{S}$ below:

1) Input: $\{\bar{g}^{s_i}\}_{i \in H}$ and the corrupt set $C = [n] \setminus H$.
2) Give $\mathcal{S}$ oracle access $\mathcal{A}$ for the corrupt set C.
3) Send $\{\bar{g}^{s_i}\}_{i \in H}$ to $\mathcal{S}$.
4) Once $\mathcal{S}$ returns $\{s_i\}_{i \in C}$ do as follows:
   a) Let $V \subseteq$ be the set that includes $s_i \neq \perp$.
   b) Compute $pk \leftarrow \prod_{i \in V \cup H} \bar{g}^{s_i}$.
   c) Compute $pk_i = \bar{g}^{P(i)}$ for all $i \in [n]$ by choosing a t-degree polynomial $P(X) = \sum_{i=0}^{t} c_i X^i$ implicitly subject to $\bar{g}^{c_0} = pk$ and each coefficient $\bar{g}^{c_i} = \prod_{j \in H} (\bar{g}^{s_j})^{r_j}$ for uniform random $r_j \xleftarrow{\$} \mathbb{Z}_q$.
   d) Using the discrete log oracle compute $sk_i = P(i)$ from $\bar{g}^{P(i)}$ for all $i \in C$.
   e) Finally send $(pk, \{pk_i\}_{i \in [n]}, \{sk_i\}_{i \in C})$ to $\mathcal{S}$.

Now, first note that, the polynomial $P(X)$ is uniformly random in $\mathbb{Z}_q[X]$, subject to $c_0 = \sum_{i \in H} s_i + \sum_{i \in V} s_i$, as coefficients $c_1, c_2, \ldots$ are chosen as random linear combinations of $\{s_i\}_{i \in H}$ in the exponent. So, clearly the simulatability matrix can have a form:

$$\begin{bmatrix} c_0 & 0 & \cdots & 0 \\ c_0 & r_{1,1} & \cdots & r_{1,t+1} \\ \vdots & \ddots & \cdots & \vdots \\ c_0 & r_{t,1} & \cdots & r_{t,t+1} \end{bmatrix}$$

which is invertible over $\mathbb{Z}_q$ as long as $c_0 \neq 0$. Since our ideal functionality ensures that honestly chosen s_i are uniformly random, this happens only with probability $1/q \leq \mathsf{negl}(\lambda)$. Finally, we note that although in our ideal functionality s_i for honest i are chosen uniformly at random, the simulator $\mathcal{S}$ does not depend on that. Basically it works for any s_i, and hence when fed with arbitrary ip by $\mathcal{S}_{\mathsf{oa}}$, nothing changes from $\mathcal{A}$'s view in the real and ideal executions. Of course for a low-entropy distribution would be not useful in the application. In fact, a closer look into the application of algebraic simulatability of their proof of BLS (Lemma 4.2) reveals that these are OMDL instances and hence are uniformly random.

With this, we can conclude that any DKG protocol that satisfies our definition with respect to $\mathcal{F}_{\mathsf{dkg}}$, satisfies the static oracle aided algebraic simulatability definition from Bacho and Loss [44].