# Non-interactive VSS using Class Groups and Application to DKG

Anonymous Author(s)

## ABSTRACT

We put forward the *first* non-interactive verifiable secret sharing (NI-VSS) scheme using class groups – 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 structure, which is required to make the proof-of-correctness highly efficient. This approach not only *substantially simplifies* the NI-VSS process, but also *outperforms* the state-of-art schemes significantly. For example, our implementation shows that for a 150 node system cgVSS outperforms (a simplified implementation of) Groth's protocol in overall communication complexity by 5.6x, about $9.3 - 9.7$x in the dealer time and $2.4 - 2.7$x in the receiver time per node.

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 potentially 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, we obtain a non-interactive distributed key generation (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.

## KEYWORDS

Class groups, Threshold Cryptography, Verifiable Secret Sharing

**ACM Reference Format:**

Anonymous Author(s). 2024. Non-interactive VSS using Class Groups and Application to DKG. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24).* ACM, New York, NY, USA, 21 pages. https://doi.org/XXXXXXX.XXXXXXX



*CCS '24, October 14–18, 2024, Salt Lake City, U.S.A*



ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

## 1 INTRODUCTION

In a threshold secret sharing scheme [10, 71], 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 [29] 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 [9, 22, 42], distributed key generation (DKG) [43, 47, 49, 58] for threshold cryptography [11, 35, 48, 53], and multiparty computation [16, 30, 59, 64].

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 [60], blockchain consensus certifcates [3], distributed randomness services [1, 36, 42], as well as generic secure multi-party computation [45]. 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 [67, 69]. 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 [5, 39, 43, 44, 55, 61, 65] 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 [28] due to their otherwise computational nature. However, any communication over secure and authenticated channels is not publicly verifiable [69], and so are these VSS 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

[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.

[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 [33, 34] 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.

number of malicious parties exceeds the threshold $t$. Nevertheless the notion of publicly verifiable secret sharing (PVSS) already exists [22, 29, 46, 47, 50, 66, 69, 72] assuming a public-key infrastructure (PKI) is in place and a non-interactive zero-knowledge (NIZK) proof of correct sharing is feasible. Furthermore, many of 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 [46, 47], developed in the blockchain context, follow a common template to construct an NI-VSS: consider a public-key infrastructure, in that each party $P_i$ has a secret decryption key $sk_i$ and public encryption key $pk_i$. Then the dealer, with a secret $s$, creates a share $s_i$ for party $P_i$ and then broadcasts a multi-receiver encryption vector, in that each $s_i$ is encrypted with corresponding public key $pk_i$. Every party $P_i$ can decrypt their own $s_i$ with $sk_i$, but nothing else. To enable (public) verifiability, the dealer additionally provides a proof that validates that the multi-receiver ciphertext correctly encrypts a $t$ out of $n$ Shamir's sharing [71] of $s$ with respect to the commitments of shares. Notably, if the encryption scheme is additively homomorphic, then the proof of correctness can be made practically efficient by exploiting the homomorphic structure. In particular, Gentry, Halevi, and Lyubashevsky [46] use a variant of Regev's lattice-based encryption, whereas Groth [47] 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 [17], thereby incurring significant performance overhead and design complexity. Instead, Groth [47] uses exponentiated ElGamal encryption over cyclic groups where discrete log is hard; however, since the plaintexts have to be small to facilitate efficient decryption (precisely because discrete log is hard), a so-called "chunking technique", in that a standard-sized (i.e. 256 bit) plaintext is split into small chunks (i.e. 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 [47]. While this makes their protocol more efficient compared to [46], 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 without forward secrecy) of Groth's protocol [47] 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.** Ours is the *first* work that demonstrates the efficacy of class group based techniques in the context of VSS/DKG. As our primary contribution we propose a new *simple* and *efficient* NI-VSS scheme (in Section 5), which follows the same "encrypt-and-prove" paradigm as above, but completely avoids any range-proof or "chunking" of the shares. In particular, we use a class group based additively homomorphic encryption scheme [26], 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 [47]) as evident by our implementations (cf. Section 7). Also, since deploying our NI-VSS scheme requires a PKI setup for class group based 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 [74], but provides a stronger knowledge extraction guarantee – we provide a modified analysis in Appendix B.

**Generic Transformation to NI-DKG.** Using a generic round-preserving and efficiency-preserving transformation to our NI-VSS protocol we obtain a class group based NI-DKG protocol, that we call cgDKG, for key generation in the discrete log (DLog)-based threshold settings (that supports threshold signatures such as BLS [11], Schnorr [68] etc.). However, we remark that cgDKG is susceptible to the so-called "biasing public-key" attack [43]. This is inevitable as a recent work by Katz [56] shows that it is impossible to construct NI-DKG without that. Nevertheless, as argued in [4, 16, 43] this suffices for many DLog-based applications such as threshold Schnorr signature, BLS, threshold decryption etc. In Appendix H we also sketch how this bias can be removed by using the same technique proposed by Gennaro et al. [43] using a perfectly hiding commitment (e.g. Pederson's) instead of the DLog commitments, but with the inevitable cost of more rounds.

**New Definitions with Public Verifiability**. Additionally we propose a new formal definition using the universal composability framework [19] for NI-VSS (cf. Section 5) that takes our stronger notion of public verifiability into account. In short, our stronger notion ensures that a VSS dealing is publicly verifiable even if more than threshold (potentially all) recipients are corrupt, when secrecy or robustness (aka guaranteed output delivery) can not be guaranteed.[4] We formally prove that our construction cgVSS securely realizes our ideal functionality $\mathcal{F}_{\mathsf{VSS}}$ (cf. Theorem 4). The 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 is also equipped with a similar notion of (strong) public verifiability. Our functionality $\mathcal{F}_{\mathsf{DKG}}$ differs substantially from existing ones [12, 51, 76] because of two reasons. Firstly, it is specially equipped to handle strong 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

[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.

[4] Note that, this is, in spirit, somewhat similar to the concept of publicly auditable MPC [52, 63], 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 [56] also formalizes this, but in a different manner.

the works such as Gentry et al. [46] and Groth [47], plausibly satisfy our definitions too. We do not investigate that formally.

**Benchmarking.** Finally, we implement cgVSS and compare that with the closest existing scheme by Groth's [47] 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 scheme (referred to as GrothVSS henceforth) proposed by Groth [47] 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 the dealer's computation time is about $9.3 - 9.7$x, and the receiver's 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, in contrast with prime order groups, which GrothVSS is based on. Essentially, the performance gain can be attributed to the design simplification, in that any range proof (or proof-of-chunking ala Groth [47]) is dispensed with.[6] Consequently, 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 cgDKG takes 47.9sec. We find that the class group implementations are still in their infancy and the scope for performance improvement is significant.

## 1.2 Discussion and more Related Work

**PVSS Litearture**. In their seminal paper, Chor et al. [29] introduced the notion of verifiable secret sharing (VSS). Stadler [72] first proposed publicly verifiable VSS (PVSS) and two constructions using verifiable ElGamal encryption. A long line of works [14, 69] ,[14, 18, 22, 41, 47, 50, 66, 72, 79] realized publicly verifiable and non-interactive VSS schemes. They typically employ encryption mechanisms, including Paillier [50, 66], ElGamal-in-the-exponent [47], pairing [37, 77] and lattice-based encryptions[46]. The schemes, that use Paillier encryption, suffer from long exponentiations and proof size, and one that uses ElGamal in the exponent [47] requires small exponents due to the hardness of the discrete log. The schemes involving pairing generate shares are group elements (not scalars) and are not suitable for settings such as threshold signatures. Lattice-based PVSS schemes [46] are indeed asymptotically efficient, albeit require large public keys and ciphertext sizes.

**Different notions of Public Verifiability**. The concept of public verifiability has been around for a long time. However, we observe that there is a lack of formalization in the literature, which resulted in 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 (that is the VSS recipients). Our *strong* public verifiability notion provides the following guarantees: (i) an honest voter's ballot (i.e. the dealing) cannot be falsely discarded or manipulated to a different vote even when all voting servers are compromised and (ii) if a ballot publicly verifies correctly, then it implies that a correct ballot must have been cast – these hold regardless of the number of corruptions, which may potentially be more than the threshold. For VSS schemes that are not publicly verifiable such as [5, 65], 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; or, a fraudulent voter may collude with the servers to produce an illegitimate vote that publicly verifies correctly.

**Recent work by Das et al. [34]**. Recently a weaker version of public verifiability has been formalized by Das et al. [34] in the VSS context that holds only against an adversary that can compromise up to $t$ recipients. This notion, while falling short of providing a guarantee in settings similar to above such as voting, can be relevant in some blockchain applications. Nevertheless, in this paper, we focus on the rather traditional notion of public verifiability, reinterpret and formally capture it through our UC-based definitions. To distinguish with the weaker variant (a la [33, 34]) we call it, for lack of a better term, strong public verifiability in this work. Also, their elegant interactive construction uses GrothVSS as a building block, and achieves significantly better performance as they do not use zero-knowledge proof. We note that our cgVSS is compatible with their paradigm and would yield an even more efficient scheme.

**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 $t + 1$ of 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 [1, 36, 73] (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, if the key is uniform at random then 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 mean a loss of so-called "unbiasability".

Strong public verifiability for DKG guarantees if at least one dealer is honest, the final secret-key/public-key pair would be correctly distributed – this holds regardless of the number of corruptions. However, this does not guarantee anything about privacy, as potentially more than $t$ parties can be corrupt and know the secret key. Therefore, unbiasability continues to hold even if more than $t$ parties are corrupt; and unpredictability, which relies on the privacy of the secret key, only holds when up to $t$ parties are corrupt. 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)

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

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

securely realizes this when more than $t$ parties are corrupt, we need to make another assumption, that is the adversary is non-rushing. [8] Otherwise, a rushing adversary would know the dealings 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. 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 realize the $\mathcal{F}_{\mathsf{DKG}}$ functionality against *rushing* adversary. In a nutshell, we obtain a degraded, yet meaningful security guarantee (perhaps the best possible) beyond $t$ corruption, while keeping the *full* security guarantee up to $t$ corruption (cf. Theorem 5).

**Prior works on DKG**. Several DKG protocols to support DLog-based threshold protocols have been studied [2, 21, 43, 47, 56–58] in the literature in the synchronous and asynchronous settings. However, to achieve (a weaker form of) public verifiability, the nodes need to typically perform a PVSS (instead of VSS or asynchronous VSS). Any aggregatable PVSS scheme [58] which supports homomorphic operations on the secret shares [47, 50, 66] may be employed to realize a (weaker) publicly verifiable DKG mechanism. However, since the stronger public verifiability notion was not formalized for VSS prior to our work, the resulting DKGs also lack the stronger property, as we defined. Groth [47] proposed an NI-DKG protocol 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. The resulting NI-DKG protocol plausibly satisfies our stronger notion – we do not investigate this formally.

**UC vs Game-based definition**. Our definition of VSS and DKG are both based on universal composability framework [19] – this helps us to capture the strong public verifiability property plus the special structure of our protocols compactly. In contrast, a recent work by Komlo et al. [58] provides a formal treatment of VSS and DKG using game-based definitions. The advantage is that their definitions do not assume any structure of the underlying protocol, so can be readily applicable to other kinds of secrets (such as lattice-based), whereas our definitions assume a cyclic group of prime order with discrete log hardness. They do not consider public verifiability.

**Cryptography from Class groups**. Class groups were used [26] to construct additive homomorphic encryption, that is structurally similar to ElGamal public-key encryption. In contrast to ElGamal, the special structures of class groups allow encryptions of large messages (e.g. 256 bits) in the exponent with an efficient decryption. 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 such as threshold ECDSA [24], verifiable delay functions [75], timed encryption [74] etc. A recent work by Braun, Damgård, and Orlandi [16] leveraged this to construct a multi-party computation protocol. En route they designed a threshold encryption scheme based on class groups that is secure with biased public keys; so our NI-DKG would apply there too. Also, we remark that class group based Verifiable Delay Functions are already deployed by Chia network [62] - this indicates the desire for these techniques in the real world too. We believe that our work would widen the spectrum of class group based crypto by effectively demonstrating new practical capabilities in an unexplored area (namely, VSS/DKG) and this would garner more attention for rigorous cryptanalysis and applications in the future.

**Another recent work [23]**. A recent follow-up work [23] proposes a class group-based PVSS scheme, structurally very similar to ours. Their PVSS construction differs from ours mainly in using an information theoretically hiding commitment, instead of DLog commitments used in this paper and the corresponding proof of correct secret sharing. In particular, their proof system uses a coding theoretic fact, adapted from [22], to check that the dealing indeed has a $t$ out of $n$ sharing – this requires a recently introduced *non-falsifibale* assumption, called rough order assumption over the class group [16]; we do not need this and all our assumptions are falsifiable. Their work [23] do not provide any implementation benchmarking. In concrete terms, as we estimate, their non-interactive protocol appears to achieve slightly better verification time, but similar communication complexity.

## 2 PRELIMINARIES

### 2.1 Notation

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 $\lambda$ (a typical value 256), and the statistical security parameter is denoted by $\lambda_{\mathsf{st}}$ (typical value 40). To denote that a value $x$ is polynomial in $\lambda$, 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 $\lambda$, if it vanishes faster than $1/\mathsf{poly}(\lambda)$ for any polynomial poly. For more preliminaries see Appendix A.

### 2.2 Class Groups

In this paper, we follow a presentation similar to [16]. 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 $\lambda$. 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 $g = g_q \cdot f$, such that $\langle g_q \rangle = G^q$. 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

[8] Using a timed-lock puzzle [74] to encode a party's message, one may generically remove this assumption, by making sure that the maximum allowed response time is shorter than the time required to decode the puzzle. For more discussion see Appendix I.

over respective domains. The framework specifies PPT algorithms (CG.ParamGen, 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 $\lambda$, the statistical security parameter $\lambda_{\mathsf{st}}$ and a modulus $q$, outputs the group parameters and the randomness $\rho$ 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 random $\rho$.

Definition 1 (Q-Hard subgroup membership assumption [27]). 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 class group framework, if for any PPT adversary $\mathcal{A}$, the following probability is negligible in $\lambda$.

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

Definition 2 (Low order assumption [25]). *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 $\lambda$:*

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

Definition 3 (Strong root assumption [25]). *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 $\lambda$:*

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

# 3 BUILDING BLOCKS

Here we present 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 and efficient proof of correct secret-sharing. Next, we present them in order. Among them the knowledge-of-exponent proof is required to setup the PKI for the multi-receiver encryption. Based on the PKI setup our NI-VSS protocol combines the encryption and proof of correct secret-sharing to realize $\mathcal{F}_{\mathsf{VSS}}$.

## 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 [25, 32, 74]. Similarly to those, we show that NIZK proof system is a secure argument of knowledge (Def. 6) from two new assumptions, hardness of finding low-order elements (Def. 2), and hardness of finding a root (Def. 3) 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**. Security of the proof system can be argued following the analysis of Schnorr's proof over cyclic group. The main difference from Schnorr's proof is 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 state the following theorem, a proof for which is deferred to Appendix B.

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. 2) and the strong root assumption (Def. 3) holds over the class group $\widehat{G}$, the NIZK proof system described above is secure argument of knowledge (Def. 6) in the random oracle model.*

## 3.2 Multi-receiver Encryption from Class groups

We first provide a definition of multi-receiver encryption by simply extending from prior notions [7, 8] where the adversary can corrupt $t$ out of $n$ parties and possibly know their secrets too.

Definition 4 (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 (KeyGen, Enc, 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 notion**. 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 produce a NIZK argument of knowledge (cf. Definition 6) 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 $(b \stackrel{?}{=} b')$.

**The Encryption Scheme**. We present multi-receiver encryption from class groups in this section. This is a simple adaptation of the base scheme from [26].

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

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

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

**Analysis**. The security argument of single-receiver scheme provided in [26] can be extended easily to the multi-receiver setting. Formally we can present the following theorem, a proof for which is deferred to Appendix C:

Theorem 2. *For any $\lambda, \lambda_{\mathrm{st}}$ and any modulus $q \in \mathbb{Z}$ let $(q, \lambda, \lambda_{\mathrm{st}}, \bar{s}, f, g_q, \widehat{G}, F, \mathcal{D}, \mathcal{D}_q; \rho) \leftarrow \mathsf{CG.ParamGen}(1^\lambda, 1^\lambda_{\mathrm{st}}, q)$ be a set of correctly generated class group parameters. Then, for that set of parameters as long as hard subgroup assumptions (Def. 1) holds, the above multi-receiver encryption scheme is secure according to Definition 4 for any $n, t \in \mathbb{N}$ such that $n > t$.*

## 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 [47] variant of sigma protocol, adapted to our class group setting. The overall idea, as we recall from [47], 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. Now we provide the details.

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 $pp_{\mathsf{CG}} \leftarrow \mathsf{CG.ParamGen}(1^\lambda, 1^\lambda_{\mathrm{st}}, q)$.

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 $(\mathsf{inst}, wit)$ 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 $j \in \{0, \ldots, t\}$: $A_j = \bar{g}^{a_j}$;[9]
- 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.

[9] 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.

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:
  - $\gamma \leftarrow H(\mathsf{inst})$.
  - $Y \leftarrow f^{\alpha} \cdot \left(h_1^{\gamma} \cdot h_2^{\gamma^2} \cdot \ldots \cdot h_n^{\gamma^n}\right)^{\rho} \in G$.
  - $\gamma' \leftarrow H'(\gamma, W, X, Y)$.
  - $z_r \leftarrow r\gamma' + \rho \in \mathbb{Z}$.
  - $z_s \leftarrow \gamma' \sum_{i=1}^{n} s_i \gamma^i + \alpha \in \mathbb{Z}_q$.
- Finally return $\pi_{\mathsf{CS}} \leftarrow (W, X, Y, z_r, z_s)$

PoCS.Ver$(pp_{\mathsf{PoC}}, \mathsf{inst}, \pi_{\mathsf{CS}}) \rightarrow 1/0$ :

- Parse $\pi_{\mathsf{CS}}$ 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.**

The following theorem shows that our construction (cf. Fig. 1) is a NIZK proof in ROM (as Def. 6) – a proof is given in Appendix D.

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. 6) in the random oracle model.*

# 4 OUR MODEL

**Communication Model**. For our non-interactive constructions, similar to all previous NI-VSS and NI-DKG construction schemes (such as [22, 46, 47]), we assume that every party has access to a *broadcast channel*. This is a common assumption for non-interactive publicly verifiable multiparty computation protocols [6, 70], where the adversary controls the communication channel and can delay the messages; however, it has to deliver those before the synchrony communication bound $\Delta$. The adversary is rushing, unless explicitly stated otherwise. Moreover, unlike interactive VSS/DKG constructions [5, 34, 43, 65], 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/NI-DKG requires 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 (CGE.KeyGen, CGE.mrEnc, 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 (Kex.Prove, Kex.Ver) and public parameters $pp_{\mathsf{Kex}}$, which is consistent with $pp_{\mathsf{CG}}$. Then we describe a protocol $\Pi_{\mathsf{PKI}}^{pp}$ 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 $(pk_i, \pi_i)$.
- 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 Kex.Ver returns 1 and also include $i$. Output $(sk_i, \{pk_i\}_{i \in Q})$.

The security argument is deferred to Appendix G. In the following sections, when we say that we **assume a PKI setup**, we imply that participants already executed the protocol $\Pi_{\mathsf{PKI}}^{pp}$. Without loss of generality, we assume that $Q = \{1, \ldots, n\}$ for simplicity.

# 5 NI-VSS USING CLASS GROUPS

In this section, we first provide the UC definition of (NI-)VSS that captures strong public verifiability. Then we describe our scheme.

## 5.1 Definition:NI-VSS

**Ideal Functionality** $\mathcal{F}_{\mathsf{VSS}}$. Following the UC paradigm [19] 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 initially. Based on that we mark sid either `honest` when $|C| \leq t$ or `corrupt` when $|C| > t$.

We note that a crucial feature of our ideal functionality is that the simulator is given all $\bar{g}^{s_i}$'s (and thereby $\bar{g}^s$) – this can be thought of as a "leakage". This is due to our use of DLog commitments, which does not have hiding in the indistinguishability sense. However, if $s$ is sampled from a high-min entropy distribution, then it is hard to compute due to the discrete log. We note that for our DKG application, this suffices, as $s$ is sampled from a uniform random distribution in the generic protocol (cf. Figure 6). It is plausible to use fully hiding commitments such as Pederson's instead to remove this leakage, but that would require a more complex NIZK proof, and would affect efficiency. We discuss how it captures the key properties.

- **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

(1) Upon $(\text{sid}, \texttt{Dealing}, s)$ from $P_D$: only if $P_D$ is honest, *and* $T[\text{sid}] = \top$: compute a $(n, t)$ Shamir secret sharing $(s_1, \ldots, s_n)$. Send $(\text{sid}, s_i)$ to each $i \in H$ and $(\text{sid}, \{\bar{g}^{s_i}\}_{i\in[n]})$ to everyone, and $(s_1, \ldots, s_n)$ to $P_D$. Additionally, send $\{s_i\}_{i\in C}$ to the adversary. Set $T[\text{sid}] \leftarrow (s, \bar{g}^s, \{\bar{g}^{s_i}\}_{i\in[n]})$. /**In this case, all properties, including privacy holds, as the dealer is honest and the corruption threshold is below $t + 1$.**/

(2) Upon $(\text{sid}, \texttt{Corrupt-Dealing})$ from $\mathcal{S}$: only if $P_D$ is corrupt *and* $T[\text{sid}] = \top$:

(a) If sid is `honest`: wait for $\mathcal{S}$ to send either $(\text{sid}, \{s_i\}_{i\in H}, \{\bar{h}_i\}_{i\in[n]}, s)$, in that case skip unless $s$ is consistent with $\{s_i\}_{i\in H}$ (this is possible as sid is `honest`), and then set $T[\text{sid}] \leftarrow (s, \bar{g}^s, \{\bar{h}_i\}_{i\in[n]})$; or $(\text{sid}, \perp)$, in which case set $T[\text{sid}] \leftarrow \perp$.

(b) Else (when sid is `corrupt`): wait for $\mathcal{S}$ to send either $(\text{sid}, \{s_i\}_{i\in H}, \{\bar{h}_i\}_{i\in[n]}, \star)$, in that case reconstruct $\bar{h}$ using Lagrange in the exponent from $\{\bar{h}_i\}_{i\in[n]}$ and set $T[\text{sid}] \leftarrow (\star, \bar{h}, \{\bar{h}_i\}_{i\in[n]})$; or $\mathcal{S}$ sends $(\text{sid}, \perp)$ when set $T[\text{sid}] \leftarrow \perp$.

– In any case, if $T[\text{sid}] \neq \perp$ then send $(\text{sid}, s_i)$ to each $i \in H$ and $(\text{sid}, \{\bar{h}_i\}_{i\in[n]})$ to everyone; otherwise if $T[\text{sid}] = \perp$, send $(\text{sid}, \perp)$ to everyone.

/**In this case, a corrupt dealer can not break uniqueness. When* sid *is* `corrupt` *the dealing is committed via DLog commitment.**/

(3) Upon $(\text{sid}, \texttt{Recon})$ from any party $P$ if $T[\text{sid}] = \top$ then skip. Else if $T[\text{sid}] = \perp$, then reply `Dealing-Failed` to $P$. Else:

(a) Send $(\text{sid}, \texttt{Recon})$ 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 reply with `Recon-Error` unless the shares are consistent, otherwise reconstruct using Lagrange interpolation to get $s$.

(A) If $T[\text{sid}] \neq (\star, \cdots)$ then check whether $T[\text{sid}] = (s, \cdot)$, and if that succeeds then send back $s$ to $P$; otherwise send back `Recon-Error` to $P$.

(B) Else if $T[\text{sid}] = (\star, \bar{h}, \cdots)$, check if $\bar{h} = \bar{g}^s$, if that fails send back `Recon-Error`, otherwise send back $s$ to $P$ and set $T[\text{sid}] \leftarrow (s, \bar{h})$.

/**There are responses from at least $t + 1$ parties. However, the responses must be consistent with the committed value during dealing in either case.**/

(ii) Else if there are $\geq t + 1$ $i$ for which $\tilde{s}_i = \texttt{No-Response}$ then reply with `Recon-Declined` to $P$./**Parties decline to the reconstruction request.**/

(iii) Else, reply with `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* `Recon-Declined`*, as in the previous case, there is no error in the protocol execution.**/

(4) Upon $(\text{sid}, \texttt{Verify}, \{\bar{h}_i\}_{i\in[n]})$ from any party $P$: if $T[\text{sid}] = (\cdots, \{\bar{h}_i\}_{i\in i})$ return `Dealing-Succeeded`. Otherwise, return `Dealing-Failed`. /**This facilitates "strong public verifiability", which holds regardless of whether* sid *is* `honest` *or* `corrupt` */

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

all existing definitions in the literature, and also referred to as *secrecy* in some of them.

- **Uniqueness.** For a potentially corrupt dealer, when sid is `honest` then uniqueness guarantees that, a dealing is always associated with a unique value (which maybe $\perp$ when dealing fails to verify). This is captured as $T[\text{sid}]$ 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. In the literature, similar properties have been captured and are called uniqueness (in [12, 58]), or strong commitment (in [28]).
- **Strong Public Verifiability.** This makes sense in all cases, regardless of either $P_D$ or sid are honest or corrupt. This is guaranteed by Step 4. In particular, whenever $T[\text{sid}]$ is populated, immediately after that, any party (including the public verifier $P_V$) can verify whether the dealing succeeded or not. Note that, in this step, the ideal adversary $\mathcal{S}$ does not engage. Now, if the dealer is honest, a successful verification immediately implies consistent dealing regardless of whether sid is corrupt. In contrast when both $P_D$ and sid are corrupt there is a possibility that $T[\text{sid}]$ has $(\star, \bar{h})$. In this case, nonetheless, any reconstruction effort fixes $T[\text{sid}]$ 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 contrast, the public verifiability as defined in Das et al. [34] only holds when $|C| \leq t$, which is equivalent to `honest` sid in our 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:

- $\text{Share}(pp, s) \rightarrow (\{s_i\}_{i\in[n]}, \text{cmt})$. The sharing algorithm produces $(n, t)$ Shamir's secret shares $(s_1 \ldots, s_n)$ of a value $s$ and the associated commitments cmt.
- $\text{ShareEnc}(pp, \text{cmt}, \{s_i, pk_i\}_{i\in[n]}) \rightarrow (R, \{E_i\}_{i\in[n]}, \pi_{\text{CS}})$. 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 $\pi_{\mathsf{CS}}$.
- $\mathsf{Verify}(pp, \mathsf{cmt}, R, \{E_i, pk_i\}_{i\in[n]}, \pi_{\mathsf{CS}}) \to 1/0$. This algorithm verifies the entire ciphertext tuple with respect to the proof $\pi_{\mathsf{CS}}$ and the commitment to output a decision bit.
- $\mathsf{ShareDec}(pp, sk_i, R, E_i) \to s_i$. 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 $(R, E_i)$.
- $\mathsf{CmtVer}(\mathsf{cmt}, i, s_i) \to 1/0$. 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 5 (NI-VSS). We say an instantiation of the protocol $\Pi_{\text{NI-VSS}}$ a *secure* NI-VSS if it *securely realizes* the ideal functionality $\mathcal{F}_{\text{VSS}}$ in the PKI setup.

- **Input.** Only the dealer $P_D$ has an input $s \in \mathbb{Z}_q$.
- **Dealing**. The dealer $P_D$ executes:
  - $(\{s_i\}_{i\in[n]}, \mathsf{cmt}) \leftarrow \mathsf{Share}(pp, s)$
  - 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.**

## 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)

- **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 $pp_{\mathsf{CG}}$.
  - An associated proof system of correct sharing (cf. Section 3.3) with algorithms $(\mathsf{PoCS.Prove}, \mathsf{PoCS.Ver})$ and public parameters $pp_{\mathsf{PoC}}$, which is consistent with $pp_{\mathsf{CG}}$.
- **Public parameters.** The public parameter $pp$ is defined as $pp \leftarrow pp_{\mathsf{CG}} \cup pp_{\mathsf{PoC}}$.

**Construction**

- $\mathsf{Share}(pp, s) \to (\{s_i\}_{i\in[n]}, \mathsf{cmt})$:
  - 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 $\mathsf{cmt} \leftarrow \{A_0, \ldots, A_t\}$.
- $\mathsf{ShareEnc}(pp, \mathsf{cmt}, \{s_i, pk_i\}_{i\in[n]}) \to (R, \{E_i\}_{i\in[n]}, \pi_{\mathsf{CS}})/\perp$.
  - 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:
    - $\mathsf{inst} = \Big(\{h_i\}_{i\in[n]}, (R, \{E_i\}_{i\in[n]}), \mathsf{cmt}\Big)$.
    - $\mathsf{wit} = ((s_1, \ldots, s_n), r)$.
  - Compute $\pi_{\mathsf{CS}} \leftarrow \mathsf{PoCS.Prove}(pp_{\mathsf{PoC}}, \mathsf{inst}, \mathsf{wit})$.
- $\mathsf{Verify}(pp, \mathsf{cmt}, R, \{E_i, pk_i\}_{i\in[n]}, \pi_{\mathsf{CS}}) \to 1/0$:
  - Parse $\mathsf{inst} \leftarrow \Big(\{h_i\}_{i\in[n]}, (R, \{E_i\}_{i\in[n]}), \mathsf{cmt}\Big)$.
  - Output $\mathsf{PoCS.Ver}(pp_{\mathsf{PoC}}, \mathsf{inst}, \pi_{\mathsf{CS}})$.
- $\mathsf{ShareDec}(pp, sk_i, R, E_i) \to s_i$:
  - Compute $s_i \leftarrow \mathsf{CGE.Dec}(pp_{\mathsf{CG}}, sk_i, R, E_i)$.
  - $\mathsf{CmtVer}(\mathsf{cmt}, i, s_i) \to 1/0$: 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

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. The security argument is formally captured by the following theorem, a proof of which is given in Appendix E.

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 4) and the NIZK proof of correctness is a secure proof system (Def 6).*

# 6 NI-DKG USING CLASS GROUPS

In this section, we use a folklore generic transformation [43, 47, 65] from VSS to construct NI-DKG. First we provide a new ideal functionality $\mathcal{F}_{\text{DKG}}$ in Figure 5, and then provide a VSS construction in $\mathcal{F}_{\text{DKG}}$-hybrid. Instantiating with cgVSS we obtain our NI-DKG protocol cgDKG. 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 $x_i$.

**Different guarantees by $\mathcal{F}_{\text{DKG}}$.** The functionality provides different guarantees depending on the modes. There are three main guarnatees: (i) *privacy* which means that the secret key is only known collectively by any $t+1$ parties; (ii) *robustness* which guarantees that the protocol always terminates; (iii) *strong 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 the secret $x$ – if there is at least one honest party, then $x$ is uniform over $\mathbb{Z}_q$ as desired.

When $n \geq 2t+1$ and $t' = |C| \leq t$, then the mode is set STRONG, in which the functionality achieves all of these guarantees. In contrast, the WEAK mode only offers *strong public verifiability*. 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 (see Appendix I, for a discussion on how to remove this), 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 – the rushing adversary may just choose the corrupt secrets once it obtains all the honest secrets, setting the final secret to, for example, 0 rendering it useless.

Finally, note that in either mode strong public verifiability is guaranteed as noted in Step 5. In STRONG mode public verifiability is captured easily, because the secret sharing is 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 $y$.[10] 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 [12, 51, 58, 76] 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. [43]. Nevertheless, as also shown in earlier works, this weaker definition suffices for many threshold applications such as threshold Schnorr's signature [43], BLS [4] etc. while offering efficiency benefit. We briefly discuss Appendix H the measures to remove this "biasability" with a two round protocol using a known technique [43]. We note that a recent work by Katz [56] also captures the biasability in a different manner.

**Our NI-DKG**. Our protocol uses a generic transformation from any NI-VSS scheme to a NI-DKG scheme. This transformation is sort of "folklore" and was used in [47]. 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 $\mathcal{F}_{\text{VSS}}$-hybrid in Figure 6. $\mathcal{F}_{\text{VSS}}$, when realized with cgVSS, the resulting protocol is called cgDKG. We formalize via the following theorem, a proof for which is given in Appendix F.

THEOREM 5 (SECURITY OF GENERIC DKG). *For parameters $n, t \in \mathbb{N}$ such that $n \geq 2t+1$, the generic DKG protocol securely UC-realizes $\mathcal{F}_{\text{DKG}}$ in $\mathcal{F}_{\text{VSS}}$-hybrid for the following adversary:*

- *Any $t$-bounded PPT adversary.*
- *Any $n$-bounded non-rushing PPT adversary.*[11]

# 7 EXPERIMENTATION AND PERFORMANCE ANALYSIS

**Implementation and Setup**. We implement cgVSS in C++ using the BICYCL library [15] for class groups, Miracl C++ library for cryptographic operations with ~ 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 anonymous link https://anonymous.4open.science/r/NIDKG-056A/class-group-master.zip

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 [78] (SMR) 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.

**Parameter setting**. We run the DKG protocol to generate shares of a 256 bit key and choose the PKI parameters to support 256 bit secret scalar. The shares generated are also 256 bit. This offers 128 bit security and the groups for the GrothVSS and cgVSS are chosen to offer the same level of security. The encryption scheme uses class-group based encryption similar to exponentiated ElGamal, and consists of two elements; each element in the class group in the compressed form [24] takes 1752 bits. Refer [15, 26, 27] for the relation between bit-length and the security level of class groups and further details. For elliptic curve operations, for either protocol, we use the BLS-381 elliptic curve. Each commitment is a curve element, taking 384 bits (closest to 381, when represented as bytes).

**Communication and Computation Overhead**. For cgVSS, the total bit-length length for the multi-receiver encryption and commitments is $(1752)\cdot(n+1)+384\cdot t$, for $n$ receivers and $t$ commitments.

[10] This can be done by, for example, a simple linear code check in the exponent akin to [9, 22].

[11] We clarify that non-rushing only becomes a requirement when the adversary corrupts more than $t$ parties. Hence, this additional guarantee only strengthens our result by providing "some guarantee" beyond $t$ corruption, in which setting prior works provide no guarantee.

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 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 STRONG and WEAK modes. If $n \geq 2t+1$ and $t' \leq t$ it sets the mode to STRONG, otherwise it sets to WEAK mode.

(1) Upon receiving (sid, `Dealing`) from all $n$ parties: only if sid is unmarked then mark sid `Live` and:
   (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 as "leakage", 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 $s_i$s are provided to the simulator.*/
(2) Upon (sid, $\{s_i\}_{i \in C} \in \mathbb{Z}_q$, ) from $\mathcal{S}$: only if (i) sid is marked `Live`; (ii) and the mode is STRONG:
   (a) Initialize a set $V \subseteq C$, 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 $x \leftarrow s$, $y \leftarrow \bar{g}^s$; $y_i \leftarrow \bar{g}^{x_i}$ where $x_i \leftarrow P(i)$ for all $i \in [n]$.
   (d) Finally send $(x_i, y_i, y)$ to party $i \in H$; (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 `End` and store (sid, $\{y_i\}_{i \in [n]}$) into a list $L$.
   /*This mode offers robustness, privacy and strong public verifiability.*/
(3) Upon (sid, $y$, $\{x_i\}_{i \in H}$, $\{y_i\}_{i \in [n]}$) from $\mathcal{S}$: only if (i) sid is marked `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 `End` and store (sid, $\{y_i\}_{i \in [n]}$) into a list $L$.
   /*This modes guarantees only strong public verifiability.*/
(4) Upon (sid, `Failure`) from $\mathcal{S}$: only if (i) sid is marked `Live`; *and* (ii) the mode is WEAK: then send $\perp$ to everyone and mark sid `End`./*In the WEAK mode, abort is allowed, so robustness does not hold.*/
(5) Upon (sid, `Verify`, $\{y_i\}_{i \in [n]}$) from any party P: only if sid is marked `End`: return 1 if and only if $\exists$ (sid, $\{\{y_i\}_{i \in [n]}\}$) $\in L$ and 0 otherwise./*In all modes strong public verifiability holds.*/

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

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 – size of the message the dealer broadcasts. Figure 8 shows the time taken by the dealer and the receiver in the cgVSS protocol. We use multi-exponentiation [13] to compute the product of multiple exponentiated values in proof of correct sharing.

To encrypt a GrothVSS share value, (assume) each share is divided into 24 chunks (chosen as per the paper [47]) and encrypted individually. The number of chunks can be varied with varying chunk size. For $n$ users, the total bit-length of ciphertexts is $9216 \cdot (n+1)$, including the random values. The dealer also commits to the $t$ coefficients of the polynomial, which amount to $257 \cdot t$ bits. The NIZK proof of correctness of sharing by the dealer constitutes 3 multiplicative group elements and two scalars of 384 bits each. For the proof of correct chunking, an approximate range proof is employed where the dealer forwards $2\ell + 2$ group elements for a parameter $\ell$ and $\ell + n + 1$ masked values of the chunks.[12] 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 $(2\ell + 2) \cdot 384 + (\ell + n + 1) \cdot 32$.

[12] $\ell$ is a crucial parameter used in GrothVSS [47]. It is tuned appropriately to enable an effective rejection sampling mechanism in their proof of chunking. Since we do not use proof of chunking, we do not need it.

| | Exp (Dealing) | Exp (Receiver) | Bit-length |
|---|---|---|---|
| cgVSS ($G$) | $n+4$ | $2n+t+5$ | $1752(n+1)+384t$ |
| GrothVSS ($\bar{G}$) | $(2n+1)m+1$ $+(2n+3\ell+1)$ | $n+4+mn$ $+(2mn+3n)$ $+(2l+1)$ | $9216(n+1)+257t$ $+(2\ell+2)384$ $+(\ell+n+1)32$ |

**Table 1: The number of exponentiations and the bit-length for the dealing for a threshold $t$. Here $\ell$ is a "rejection sampling parameter" chosen for the approximate range proof for GrothVSS and $m$ denotes the number of chunks. Also note that the order of class-group $G$ is $q.s$ ($s$ unknown), whereas that of $\bar{G}$ is $q$.**

Figure 7 indicates a 5.6x improvement in total broadcast message length while using cgVSS when compared to GrothVSS for 150 nodes. The comparison also indicates that the broadcast message length increases slower in cgVSS when compared to GrothVSS.

Table 1 indicates the number of exponentiations involved at the dealer and each receiver for cgVSSand GrothVSS. Though an exponentiation in the class group is longer, cgVSSgains significantly with the reduction of the number of exponentiations.

We also provide a brief discussion on the performance of LWE-based PVSS [46] in Appendix J for completeness.

**Ingredients and parameters.** We consider $n$ parties $P_1, \ldots, P_n$ are running this protocol with a threshold $t < n/2$ in $\mathcal{F}_{\text{VSS}}$ 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 $(\text{sid}, \texttt{Dealing})$, sample $s_i \xleftarrow{\$} \mathbb{Z}_q$ and send $(\text{sid}_i, \texttt{Dealing}, s_i)$ to $\mathcal{F}_{\text{VSS}}$ where $\text{sid}_i \leftarrow (\text{sid}, i)$. Then:

- Receive $(\text{sid}_i, (s_{i1}, \ldots, s_{in}))$ from $\mathcal{F}_{\text{VSS}}$.
- For all $j \in [n] \setminus \{i\}$ receive $(\text{sid}_j, s_{ji}, \bar{g}^{s_{j1}}, \ldots, \bar{g}^{s_{jn}})$ or $\perp$ from $\mathcal{F}_{\text{VSS}}$. 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 $(\text{sid}, \{y_i\}_{i \in [n]}, \{\bar{h}_{ij}\}_{i,j \in [n]})$ where $\bar{h}_{ij} = \bar{g}_i^{s_j}$.

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

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

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

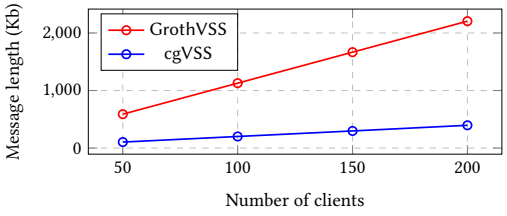
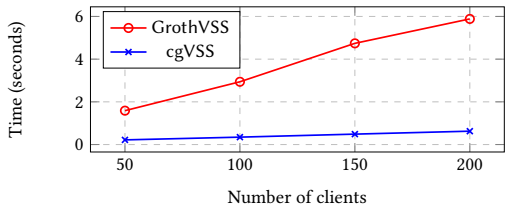


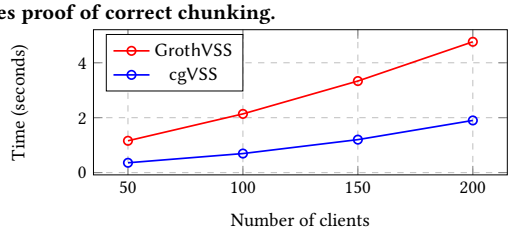

**Figure 7: Comparison of broadcast (dealing) message length where $n = 2t + 1$. cgVSS dealing consists of encryptions and proof of correct sharing, while GrothVSS also consists of proof of correct chunking.**

**NI-DKG Protocol Analysis**. We benchmark 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 [78] SMR. Figure 8 and Figure 9, also indicate 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 continues to perform significantly better than GrothDKG with an increasing number of nodes in the system. The implementation is




**(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**.**

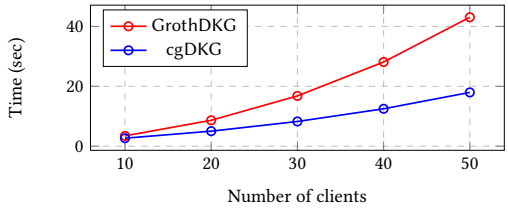



**Figure 9: Comparison of time taken to perform a DKG. GrothDKG is realized using GrothVSS where each party acts as a dealer and runs an instance of GrothVSS. The times reported are aggregates of time taken from starting of dealing and computation of public key by each node, across nodes**

also significantly simpler since the chunking and proof of correct chunking mechanisms are no longer needed.

## 8 CONCLUSION

In this work we propose a class-group based NI-VSS and NI-DKG protocols for a discrete log based key generation. 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. Importing and adapting class-group techniques to the regime of VSS/DKG is our primary contribution.

Additionally, we explore and re-interpret the semantic of public verifiability from the literature in the context of VSS/DKG, in line with auditable MPC. We provide the first formalization of a new public verifiability property (we called *strong* public verifiability), discuss its significance in specific VSS/DKG applications. We believe this new comprehensive study holds independent significance for the broader field of threshold cryptography.

# REFERENCES


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

[2] Ittai Abraham, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, and Gilad Stern. Bingo: Adaptively secure packed asynchronous verifiable secret sharing and asynchronous distributed key generation. *Cryptology ePrint Archive*, 2022.

[3] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Maofan Yin. Sync hotstuff: Simple and practical synchronous state machine replication. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 106–118. IEEE, 2020.

[4] Renas Bacho and Julian Loss. On the adaptive security of the threshold bls signature scheme. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 193–207, 2022.

[5] Michael Backes, Aniket Kate, and Arpita Patra. Computational verifiable secret sharing revisited. In *Advances in Cryptology—ASIACRYPT*, pages 590–609, 2011.

[6] Carsten Baum, Ivan Damgård, and Claudio Orlandi. Publicly auditable secure multi-party computation. In *Security and Cryptography for Networks (SCN)*, pages 175–196, 2014.

[7] Mihir Bellare, Alexandra Boldyreva, Kaoru Kurosawa, and Jessica Staddon. Multirecipient encryption schemes: How to save on bandwidth and computation without sacrificing security. *IEEE Transactions on Information Theory*, 53(11):3927–3943, 2007.

[8] Mihir Bellare, Alexandra Boldyreva, and Jessica 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*, pages 85–99. Springer, 2002.

[9] Adithya Bhat, Nibesh Shrestha, Aniket Kate, and Kartik Nayak. Optrand: Optimistically responsive reconfigurable distributed randomness. In *NDSS*, 2023.

[10] G. R. Blakley. Safeguarding cryptographic keys. In *1979 International Workshop on Managing Requirements Knowledge (MARK)*, pages 313–318, 1979.

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

[12] Dan Boneh and Victor Shoup. A graduate course in applied cryptography.

[13] Jonathan Bootle. Efficient multi-exponentiation. https://jbootle.github.io/Misc/pippenger.pdf.

[14] Fabrice Boudot and Jacques Traoré. 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*, pages 87–102, 1999.

[15] Cyril Bouvier, Guilhem Castagnos, Laurent Imbert, and Fabien 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.

[16] Lennart Braun, Ivan Damgård, and Claudio Orlandi. Secure multiparty computation from threshold encryption based on class groups. To appear at Crypto 2023, 2022.

[17] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE symposium on security and privacy (SP)*, pages 315–334. IEEE, 2018.

[18] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *Annual International Cryptology Conference*, pages 126–144. Springer, 2003.

[19] Ran 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*, pages 136–145. IEEE Computer Society, 2001.

[20] Ran Canetti, Asaf Cohen, and Yehuda Lindell. A simpler variant of universally composable security for standard multiparty computation. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, pages 3–22, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[21] Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi 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, 2020. Association for Computing Machinery.

[22] Ignacio Cascudo and Bernardo David. Scrape: Scalable randomness attested by public entities. In *ACNS*, pages 537–556, 2017.

[23] Ignacio Cascudo and Bernardo David. Publicly verifiable secret sharing over class groups and applications to dkg and yoso. Cryptology ePrint Archive, Paper 2023/1651, 2023. https://eprint.iacr.org/2023/1651.

[24] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Two-party ecdsa from hash proof systems and efficient instantiations. In *Annual International Cryptology Conference*, pages 191–221. Springer, 2019.

[25] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Bandwidth-efficient threshold ec-dsa. In *23rd IACR International Conference on Practice and Theory of Public-Key Cryptography*, pages 266–296, 2020.

[26] Guilhem Castagnos and Fabien Laguillaumie. Linearly homomorphic encryption from ddh. In *Cryptographers' Track at the RSA Conference*, pages 487–505. Springer, 2015.

[27] Guilhem Castagnos, Fabien Laguillaumie, and Ida Tucker. Threshold linearly homomorphic encryption on bfz/$2^{\text{kbfz}}$. In *Advances in Cryptology - ASIACRYPT 2022*, pages 99–129, 2022.

[28] Anirudh Chandramouli, Ashish Choudhury, and Arpita Patra. A survey on perfectly secure verifiable secret-sharing. *ACM Comput. Surv.*, 54(11s):232:1–232:36, 2022.

[29] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *FOCS*, page 383–395, 1985.

[30] Ronald Cramer, Ivan Damgård, and Ueli Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 316–334. Springer, 2000.

[31] Ivan Damgaard. On sigma protocols. *Lecture Notes*, 2010.

[32] Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. 2002.

[33] Sourav Das, Vinith Krishnan, Irene Miriam Isaac, and Ling Ren. Spurt: Scalable distributed randomness beacon with transparent setup. In *43rd IEEE Symposium on Security and Privacy*, pages 2502–2517. IEEE, 2022.

[34] Sourav Das, Zhuolun Xiang, Alin Tomescu, Alexander Spiegelman, Benny Pinkas, and Ling Ren. A new paradigm for verifiable secret sharing. Cryptology ePrint Archive, Paper 2023/1196, 2023. https://eprint.iacr.org/2023/1196.

[35] Yvo G Desmedt. Threshold cryptography. *European Transactions on Telecommunications*, 5(4):449–458, 1994.

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

[37] Roy D'Souza, David Jao, Ilya Mironov, and Omkant Pandey. Publicly verifiable secret sharing for cloud-based key management. In *INDOCRYPT 2011: 12th International Conference on Cryptology 2011*, pages 290–309. Springer, 2011.

[38] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the fiat-shamir transform. In *Progress in Cryptology-INDOCRYPT 2012: 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings 13*, pages 60–79. Springer, 2012.

[39] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 427–438, 1987.

[40] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, pages 186–194, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.

[41] Pierre-Alain Fouque and Jacques Stern. One round threshold discrete-log key generation without private channels. In *Public Key Cryptography*, pages 300–316, 2001.

[42] David Galindo, Jia Liu, Mihair Ordean, and Jin-Mann Wong. Fully distributed verifiable random functions and their application to decentralised random beacons. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 88–102. IEEE, 2021.

[43] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *EUROCRYPT'99*, page 295–310, 1999.

[44] Rosario Gennaro, Michael O. Rabin, and Tal Rabin. Simplified vss and fast-track multiparty computations with applications to threshold cryptography. In *ACM PODC*, page 101–111, 1998.

[45] Craig Gentry, Shai Halevi, Hugo Krawczyk, Bernardo Magri, Jesper Buus Nielsen, Tal Rabin, and Sophia Yakoubov. Yoso: You only speak once: Secure mpc with stateless ephemeral roles. In *Advances in Cryptology — Crypto*, pages 64–93, 2021.

[46] Craig Gentry, Shai Halevi, and Vadim Lyubashevsky. Practical non-interactive publicly verifiable secret sharing with thousands of parties. In *Advances in Cryptology—EUROCRYPT*, pages 458–487, 2022.

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

[48] Jens Groth and Victor Shoup. Design and analysis of a distributed ECDSA signing service. *IACR Cryptol. ePrint Arch.*, page 506, 2022.

[49] Kobi Gurkan, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. Aggregatable distributed key generation. In *Advances in Cryptology - EUROCRYPT 2021*, pages 147–176, 2021.

[50] Somayeh Heidarvand and Jorge L. Villar. Public verifiability from pairings in secret sharing schemes. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography*, pages 294–308, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[51] Stanislaw Jarecki, Aggelos Kiayias, Hugo Krawczyk, and Jiayu Xu. TOPPSS: cost-minimal password-protected secret sharing based on threshold OPRF. In *Applied Cryptography and Network Security (ACNS) 2017*.

[52] Sanket Kanjalkar, Ye Zhang, Shreyas Gandlur, and Andrew Miller. Publicly auditable mpc-as-a-service with succinct verification and universal setup, 2021.

[53] Aniket Kate and Ian Goldberg. Distributed private-key generators for identity-based cryptography. In *SCN*, pages 436–453, 2010.
[54] Aniket Kate, Easwar Vivek Mangipudi, Siva Maradana, and Pratyay Mukherjee. Flexirand: Output private (distributed) vrfs and application to blockchains. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, pages 1776–1790. ACM, 2023.
[55] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 177–194. Springer, 2010.
[56] Jonathan Katz. Round optimal robust distributed key generation. Cryptology ePrint Archive, Paper 2023/1094, 2023. https://eprint.iacr.org/2023/1094.
[57] Eleftherios Kokoris Kogias, Dahlia Malkhi, and Alexander Spiegelman. Asynchronous distributed key generation for computationally-secure randomness, consensus, and threshold signatures. New York, NY, USA, 2020. Association for Computing Machinery.
[58] Chelsea Komlo, Ian Goldberg, and Douglas Stebila. A formal treatment of distributed key generation, and new constructions. *Cryptology ePrint Archive*, 2023.
[59] Donghang Lu, Thomas Yurek, Samarth Kulshreshtha, Rahul Govind, Aniket Kate, and Andrew Miller. Honeybadgermpc and asynchromix: Practical asynchronous MPC and its application to anonymous communication. In *ACM CCS*, pages 887–903, 2019.
[60] Easwar Vivek Mangipudi, Udit Desai, Mohsen Minaei, Mainack Mondal, and Aniket 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.
[61] Wafa Neji, Kaouther Blibech, and Narjes Ben Rajeb. Distributed key generation protocol with a new complaint management strategy. *Security and Communication Networks*, 9(17):4585–4595, 2016.
[62] Chia Network. Chia VDF competition and implementation. https://github.com/Chia-Network/oldvdf-competition.
[63] Alex Ozdemir and Dan 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.
[64] Arpita Patra, Ashish Choudhury, and C Pandu Rangan. Efficient asynchronous verifiable secret sharing and multiparty computation. *Journal of Cryptology*, 28:49–109, 2015.
[65] Torben Pryds Pedersen. A threshold cryptosystem without a trusted party. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 522–526. Springer, 1991.
[66] Alexandre Ruiz and Jorge L. Villar. Publicly verifiable secret sharing from paillier's cryptosystem. In *WEWoRC 2005 – Western European Workshop on Research in Cryptology*, pages 98–108, 2005.
[67] Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Publicly verifiable proofs from blockchains. In *Public-Key Cryptography - PKC 2019*, pages 374–401, 2019.
[68] C. P. Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89*, pages 239–252, 1990.
[69] Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic. In *Advances in Cryptology—CRYPTO*, pages 148–164, 1999.
[70] Berry Schoenmakers and Meilof Veeningen. Universally verifiable multiparty computation from threshold homomorphic cryptosystems. In *Applied Cryptography and Network Security (ACNS)*, pages 3–22, 2015.
[71] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
[72] Markus Stadler. Publicly verifiable secret sharing. In *Advances in Cryptology—EUROCRYPT*, pages 190–199, 1996.
[73] Supra Research. Supra VRF Service. https://supraoracles.com/docs/SupraOracles-VRF-Service-Whitepaper.pdf.
[74] Sri Aravinda Krishnan Thyagarajan, Guilhem Castagnos, Fabian Laguillaumie, and Giulio Malavolta. Efficient cca timed commitments in class groups. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, 2021.
[75] Benjamin Wesolowski. Efficient verifiable delay functions. In *Advances in Cryptology–EUROCRYPT 2019*, pages 379–407, 2019.
[76] Douglas Wikström. Universally composable DKG with linear number of exponentiations. In *Security in Communication Networks, 4th International Conference, SCN 2004*, 2004.
[77] Tsu-Yang Wu and Yuh-Min Tseng. A pairing-based publicly verifiable secret sharing scheme. *Journal of systems science and complexity*, 24(1):186–194, 2011.
[78] Maofan Yin, Dahlia Malkhi, Michael K Reiter, Guy Golan Gueta, and Ittai Abraham. Hotstuff: Bft consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 347–356, 2019.
[79] Adam Young and Moti Yung. A pvss as hard as discrete log and shareholder separability. In *Public Key Cryptography*, pages 287–299, 2001.

# Appendix

## A ADDITIONAL PRELIMINARIES

### A.1 Shamir Secret Sharing

We use Shamir's secret sharing [71]. In a typical $(n, t)$-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 $L_i$s 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 $\mathsf{Shamir}_{n,t,q}(s) = (s_1, \ldots, s_n)$. Furthermore, given any $s_1, \ldots, s_{t'}$ for $n \geq t' \geq t+1$ it is possible to verify whether they are a consistent $(n, t)$ sharing.

### A.2 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 $\mathsf{cmt}(s)$.

### A.3 NIZK proofs

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 6 (Non-interactive Zero-knowledge Argument of knowledge (resp. Proof) in ROM). Let $pp$ be some public parameters that include a computational security parameter $\lambda$, and a statistical security parameter $\lambda_{\mathsf{st}}$, 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 $\pi$ 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 $\pi$, 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. secure proof system)* 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 $\lambda_{\mathsf{st}}$ 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 $\lambda, \lambda_{\mathsf{st}}$.

**Fiat-Shamir transform [40]**. Let $(\mathsf{Prove}, \mathsf{Verify})$ be a constant round public-coin honest-verifier zero-knowledge interactive proof system (a sigma [31] protocol) with unique responses. Let $H$ be a function with range equal to the space of the verifier's coins. In the random oracle model, the proof system $(\mathsf{Prove}, \mathsf{Verify})$ by applying the Fiat-Shamir transform satisfies the zero-knowledge and argument of knowledge properties defined above. See Definition 1, 2 and Theorem 1, 3 in Faust et al. [38] for more details. (They actually show that these properties hold even when adversary can ask for proofs of false instances.)

## A.4 Universal Composability

We briefly describe UC frameowrk, taken almost verbatim from [54]. In the UC framework, a PPT algorithm called the environment (which is adversarial) is trying to distinguish between a real and an ideal world. The adversary in the protocol can corrupt parties in the real world, whereas an ideal adversary, called the simulator, simulates the adversarial behavior in the ideal world. The ideal world comprises an ideal functionality (a.k.a. trusted third party) that is directly connected to all the parties, among which the simulator fully controls the corrupt ones. The honest ideal world parties are called dummy parties because they are interfaces between the environment and the ideal functionality. The objective is to design a simulator in the ideal world such that no environment providing inputs to and observing the outputs from the computing entities can distinguish between the real world and the ideal world, given the adversary's view of both worlds. The simulator typically simulates the real world to an instance of the real-world adversary by providing messages on behalf of the honest parties while accessing the ideal functionality and finally outputs whatever the adversary outputs. The simulator can schedule messaging and outputs in the ideal world to prevent trivial distinctions by timing.

We say that a protocol $\pi$ *securely realizes* an ideal functionality $\mathcal{F}$ if for any real world adversary in the real world there is an ideal adversary (or simulator) in the ideal world such that the adversary's view in both these worlds are computationally indistinguishable.

All entities are formally modeled as instances of an interactive Turing machine, or ITI. For a detailed formalization, we refer to [19, 20].

# B SECURITY OF NIZK PROOF OF EXPONENT OVER CLASS GROUP

We recall the construction from Section 3.1. Let us first provide some intuition. As detailed in Definition 6, 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 [25, 32, 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. We prove Theorem 3.1 below.

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

**Statistical Zero-knowledge in ROM**. 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 $2^{\lambda_{\text{st}}} \cdot |\mathcal{D}_q| \cdot \mathcal{B} \geq r > 2^{\lambda_{\text{st}}-1} \cdot |\mathcal{D}_q| \cdot \mathcal{B}$. This happens with probability $2^{-\lambda_{\text{st}}}$. So the statistical distance btween the simulated and real proof is bounded by $\text{negl}(\lambda_{\text{st}})$ 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 [25, 32, 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 $\rho$. In this case we can write using Euclidean GCD: $d = \alpha(s - s') + \beta(c - c')$ for integers $\alpha, \beta$. Then we have:

    $$\begin{aligned} g_q^d &= g_q^{\alpha(s-s')+\beta(c-c')} \\ &= h^{fd\alpha} g_q^{fd\beta} \end{aligned}$$

    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 $[(2^\lambda_{\text{st}} + 1) \cdot |\mathcal{D}_q| \cdot \mathcal{B}]$ 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.

□

# C SECURITY OF THE MULTI-RECEIVER ENCRYPTION SCHEME

We provide detailed proof of the security of class group-based multi-receiver encryption scheme. Specifically we prove Theorem 3.2 below.

Proof. The proof idea basically follows footsteps of the proof for the linearly homomorphic encryption scheme provided in [27], 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 $b = 0$: $\text{Game}_0$ and with $b = 1$: $\text{Game}_1$. We show that using the hard subgroup assumption (Def. 1) we can move from $\text{Game}_0$ 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 $\text{Game}_1$ to the same mental game. To start with first note that in $\text{Game}_0$ the adversary's view can be expressed as:

$$sk_2, h_1 = g_q^{sk_1}, h_2 = g_q^{sk_2}; R = g_q^r; E_1 = f^{m_1} h_1^r; E_2 = f^{m_2} h_2^r$$

$$\text{where } sk_1, sk_2, r \xleftarrow{\$} \mathcal{D}_q$$

In $\text{Hyb}_1$ we can write $E_1 = f^{m_1} R^{sk_1}$ and $E_2 = f^{m_1} R^{sk_2}$, and clearly $\text{Hyb}_1$ and $\text{Game}_0$ are identically distributed. In the next hybrid $\text{Hyb}_2$, $sk_1$ 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 $2^{-\lambda_{\text{st}}}$ 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 $2^{-\lambda_{\text{st}}}$ close to the uniform distribution over $\mathbb{Z}_{qs}$ in the exponent. Therefore, we can conclude that the statistical distance between $\text{Hyb}_1$ and $\text{Hyb}_2$ is bounded by $2^{-\lambda_{\text{st}}+1}$.

In $\text{Hyb}_3$ we change $R$ to $R = f^u g_q^r$ for $u \xleftarrow{\$} \mathbb{Z}_q$. Now we can argue that $\text{Hyb}_2$ is indistinguishable from $\text{Hyb}_3$ as long as the hard sub-group problem (Def. 1) 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:

$$sk_2, h_1 = g_q^{sk_1}, h_2 = g_q^{sk_2}; R; E_1 = f^{m_1} R^{sk_1}; E_2 = f^{m_2} R^{sk_2}$$

$$\text{where } sk_1 \xleftarrow{\$} \mathcal{D}; sk_2 \xleftarrow{\$} \mathcal{D}_q$$

Clearly when $R = g_q^r$ $\text{Hyb}_2$ is simulated, and when $R = f^u g_q^r$ then $\text{Hyb}_3$ is simulated.

In $\text{Hyb}_3$ 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 $\text{Hyb}_2$, we change the sampling of $sk_1$ from a distribution, which is $2^{-\lambda_{\text{st}}}$ 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 $s^{-\lambda_{\text{st}}}$ close to uniformly random value in $\mathbb{Z}_q$. So, $m_1 + u \cdot sk_1 \bmod q$ is $2^{-\lambda_{\text{st}}}$ close to uniform random value in $\mathbb{Z}_q$. Similarly we can arrive at $\text{Hyb}_3$ starting from $\text{Game}_1$. Hence we can conclude that $\text{Game}_0$ and $\text{Game}_1$ is computationally indistinguishable – this concludes the proof.

□

# D SECURITY OF NIZK PROOF OF CORRECTNESS OF SECRET-SHARING

Here we provide security arguments for the NIZK proof of correct secret sharing from Section 3.3. In particular, we show that the proof system satisfies *completeness*, *statistical soundness*, and *zero-knowledge* in the random oracle model as per Definition 6. Specifically we prove Theorem 3 below.

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 in ROM**. The soundness argument is essentially the same as the one given by Groth [47](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 $\lambda_{\mathsf{st}}$.

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:

- $\mathsf{Event}_1$: The adversary predicts $\gamma'$ 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.
- $\mathsf{Event}_2$: The adversary manages to find a $\gamma$ 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.
- $\mathsf{Event}_3$: 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 6. *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*

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

Proof. Assume that the adversary makes at most $Q_{H'}$ queries to $H'$, the probability with which the adversary correctly predicts a corret $\gamma'$ 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 7. *As long as $\gamma$ is chosen uniformly at random in $\mathbb{Z}_q$, we have that*

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

Proof. First note that, since $\mathsf{Event}_1$ does not happen, $\gamma'$ is computed legitimately after computing $\gamma, X, Y, Z$. Then, by definition if $\mathsf{Event}_2$ 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 $\gamma$ 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 8. *As long as $\gamma'$ is chosen uniformly at random, we have that:*

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

Proof. In this case since $\mathsf{Event}_1$ and $\mathsf{Event}_2$ 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 $\gamma$ the probability of this holding equation over the choice of a uniform random $\gamma'$ 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 $\lambda_{\mathsf{st}}$ for a typical choice of $\lambda_{\mathsf{st}} = 40$.

**Statistical Zero-knowledge in ROM.**[13] Following [47], we argue the statistical zero-knowledge of the proof of correct sharing in the ROM. The PPT simulator works as follows:

(1) Set $H(\mathsf{inst}) = \gamma$ where $\gamma$ is uniformly at random.
(2) Choose $\gamma'$ uniformly at random.
(3) Sample $z_r \xleftarrow{\$} [q \cdot |\mathcal{D}_q| \cdot 2^{\lambda}_{\mathsf{st}}]$ and $z_s \xleftarrow{\$} \mathbb{Z}_q$.
(4) Compute $X, Y, W$ from the three verification equations.
(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 $\leq \mathsf{negl}(\lambda_{\mathsf{st}})$ 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 $\lambda_{\mathsf{st}}$. □

## E SECURITY PROOF FOR cgVSS

We provide a detailed proof of Theorem 4 here.

Proof. Consider four mutually exclusive and exhaustive cases:

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

[13] We note that, in ROM it is possible to achieve both zero-knowledge and soundness statistically So we do not have to rely on any other assumption here. However, we need to limit the number of queries made to the RO.

For each Case-$i$ we construct a separate simulator $\mathcal{S}_i$. Our actual simulator $\mathcal{S}$ first obtains $pp_{\mathsf{CG}}$ 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.

**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 $\pi_{\mathsf{CS}}$ 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 $(\mathsf{sid}, \mathsf{Recon})$ 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 $\tilde{s}_i \leftarrow \mathsf{No\text{-}Response}$ 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 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** $\mathsf{Hyb}_1$. This hybrid is the same as cgVSS, except that the proof of correctness $\pi_{\mathsf{CS}}$ 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, $\mathsf{ShareEnc}'$ (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** $\mathsf{Hyb}_2$. This hybrid is the same as $\mathsf{Hyb}_1$ 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 $|C| \leq t$ 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 $\pi_{\mathsf{CS}}$ is a simulated proof.
  (5) Then give the following to $\mathcal{A}$: $\left(R, \{E_i\}_{i \in [n]}, \mathsf{cmt}, \pi_{\mathsf{CS}}\right)$

(6) The rest remains unchanged.

We prove that:

Lemma 9. $\mathsf{Hyb}_1$ *and* $\mathsf{Hyb}_2$ *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 $pp_{\mathsf{CG}}$ from the challenger.
- Send $C$ to the challenger obtain $\{pk_i\}_{i\in H}$.
- Sample appropriate $pp_{\mathsf{Kex}}$ 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 $\Pi_{\mathsf{PKI}}^{pp}$ to obtain $\{pk_i\}_{i\in C}$
- Obtain $pp_{\mathsf{CG}}$ 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 $pp_{\mathsf{PoC}}$ for the proof of correct sharing scheme such that they are consistent with $pp_{\mathsf{CG}}$. 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 $\pi_{\mathsf{CS}}$ 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 $\mathsf{Hyb}_1$ return 0 to the challenger, and in case $\mathcal{A}$ concludes $\mathsf{Hyb}_2$, then send 1.

It is easy to argue that if $b = 0$, $\mathcal{A}$'s view is the same as in $\mathsf{Hyb}_1$, and when $b = 1$, that is the same as in $\mathsf{Hyb}_2$. So the probability of $\mathcal{A}$'s breaking $\mathsf{Hyb}_1$ and $\mathsf{Hyb}_2$ is upper bounded by the probability of the reduction's breaking the security of the encryption. This concludes the proof. □

Finally we note that $\mathsf{Hyb}_2$ 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 $\Pi_{\mathsf{PKI}}$ 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 $\pi_{\mathsf{CS}}$, 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 $(\mathsf{sid}, \mathsf{Recon})$ query, communicate with the adversary about reconstruction. When adversary sends back $\{s'_i\}_{i\in C'}$ for $C' \subseteq C$, define $\tilde{s}_i \leftarrow \mathsf{No\text{-}Response}$ 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 $\pi_{\mathsf{CS}}$ 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 `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 $\pi_{\mathsf{CS}}$ – 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 $(\mathsf{sid}, \mathsf{Recon})$ 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 $(\mathsf{sid}, s)$ in Step 1. This means, there is no privacy of $s$ and the simulation becomes straightforward for this part. However, the $(\mathsf{sid}, \mathsf{Recon})$ 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 $T[\mathsf{sid}] = \perp$.

# F SECURITY OF NI-DKG

We provide the security analysis of our generic NI-DKG protocol (Figure 6) in $\mathcal{F}_{\mathsf{VSS}}$-hybrid by proving Theorem 5 below.

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 $(\text{sid}_i, \{\bar{h}_{ij}\}_{j\in[n]}, \{s_{ij}\}_{j\in C})$ to the adversary.
- For all $i \in C$ (assuming, for simplicity, no $\perp$ is returned) receive $(\text{sid}_i, s_i, \{s_{ij}\}_{j\in H}, \{\bar{h}_{ij}\}_{j\in[n]})$ or $\perp$ from $\mathcal{F}_{\text{VSS}}$. Reconstruct $\{\bar{h}^i\}_{i\in C}$ using Lagrange in the exponent. Send $\{s_i\}_{i\in C}$ to $\mathcal{F}_{\text{DKG}}$.
- Get back $\{(y, \{y_i\}_{i\in[n]}), \{x_i\}_{i\in C}\}$ from $\mathcal{F}_{\text{DKG}}$, which it outputs.
- Store $(\text{sid}, \{y_i\}_{i\in[n]}, \{\bar{h}_{ij}\}_{i,j\in[n]})$.
- In response to $(\text{sid}, \texttt{Verify}, \{y_i\}_{i\in[n]})$ then look up an $(\text{sid}, \{y_i\}_{i\in[n]}, \{\bar{h}_{ij}\}_{i,j\in[n]})$, 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 $i \in C$ $(\text{sid}_i, \{s_{ij}\}_{j\in H}, \{\bar{h}_{ij}\}_{j\in[n]})$ (or $\perp$, but for simplicity we assume it does not receive any $\perp$) from multiple instances of $\mathcal{F}_{\text{VSS}}$ before it sends anything to the adversary. Then the simulator works as follows:

- Obtain $\{s_i\}_H$ from $\mathcal{F}_{\text{DKG}}$.
- For all $i \in H$: send $(\text{sid}_i, \texttt{Dealing}, s_i)$ to $\mathcal{F}_{\text{VSS}}$. 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 $(\text{sid}, \{x_i\}_{i\in H}, \{y_i\}_{i\in[n]})$ to $\mathcal{F}_{\text{DKG}}$.
- Store $(\text{sid}, \{y_i\}_{i\in[n]}, \{\bar{h}_{ij}\}_{i,j\in[n]})$.
- It may send $(\text{sid}, \texttt{Failure})$ in certain cases, for example if $t' = n$ and all corrupt party returns $\perp$.
- In response to $(\text{sid}, \texttt{Verify}, \{y_i\}_{i\in[n]})$ then look up an $(\text{sid}, \{y_i\}_{i\in[n]}, \{\bar{h}_{ij}\}_{i,j\in[n]})$, 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.

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.

□

# G SECURITY OF PKI

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. Def. 6) we obtain that if $\text{Kex.Ver}(pk_i, \pi_i)$ 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.

# H 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. [43]. This is because a rushing adversary can observe the first $t$ verified secret sharings and then perform a valid $t+1$st 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 [61] 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 [43] 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.

# I DEALING WITH RUSHING ADVERSARIES USING TIMED LOCK PUZZLE GENERICALLY

Recall that a rushing adversary waits for all messages from the honest parties to arrive in a given round and then choose its own message, which can depend on the honest party's message arbitrarily. This makes adversary more powerful than a non-rushing adversary, who sends the messages without waiting for honest party's messages in a round.

We can hope to relax the assumption of the adversary being non-rushing by making use of a "folklore" trick of employing time-lock puzzles [74]. The idea is simple: parties may encode their message in time-lock puzzles for a stipulated time that is greater than the maximum allowed response time, which is bounded by $\Delta$ (the synchrony communication bound) for the parties – this is possible to define as we assume a synchronous setting and thus have a predefined $\Delta$. Timed-lock puzzles guarantee that until the stipulated time, no adversary can learn any information about the honest parties' messages from their puzzles. Since the allowed response time is shorter than the timing hardness of the honest puzzles, the adversary has to commit to its messages without knowing any information about the honest parties' messages. A practical downside of this approach is that everyone is required to solve the puzzles

in each round which is a computationally intensive task. However, this can also be amortized if we allow parties to send the messages in plain once it obtains everyone else's commitment to the puzzles – this requires an extra round of communication for each existing round. Even though adversarial puzzles may remain unopened and need to be solved, Thyagarajan et al. [74] give a construction of class-group-based time-lock puzzles that allow for batched solving of these puzzles, resulting in computational effort for solving just a single puzzle.

## J COMMENT ON LWE BASED PVSS TIMINGS

To also give a sense of how the scheme compares to other existing state-of-the-art PVSS schemes, we briefly discusss the performance reported by Gentry et al. [46] for their *LWE-based* PVSS scheme. 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. In comparison, for example, for a 200 node network, the dealer and receiver times of our cgVSS are 0.62 sec and 1.9 sec respectively. However, it appears that in terms of size / communication complexity, their performance would be much worse compared to ours, as lattice-based scheme are usually known to suffer from this. A precise calculation of this does not seem immediate and they do not provide any benchmarking.

Note that, 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 more details we refer to their paper [46].