An Improved Threshold Homomorphic Cryptosystem Based on Class Groups

Lennart Braun, Guilhem Castagnos, Ivan Damgård, Fabien Laguillaumie, Kelsey Melissaris, Claudio Orlandi, Ida Tucker

▶ **To cite this version:**

Lennart Braun, Guilhem Castagnos, Ivan Damgård, Fabien Laguillaumie, Kelsey Melissaris, et al.. An Improved Threshold Homomorphic Cryptosystem Based on Class Groups. SCN 2024 - 14th International Conference on Security and Cryptography for Networks, Sep 2024, Amalfi, Italy. pp.24-46, 10.1007/978-3-031-71073-5_2 . hal-04820390

HAL Id: hal-04820390
https://inria.hal.science/hal-04820390v1

Submitted on 5 Dec 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

An Improved Threshold Homomorphic Cryptosystem Based on Class Groups

Lennart Braun[1], Guilhem Castagnos[2], Ivan Damgård[1], Fabien Laguillaumie[3], Kelsey Melissaris[1], Claudio Orlandi[1], and Ida Tucker[4]

[1] Aarhus University
[2] University of Bordeaux
[3] University of Montpellier
[4] Zondax
{braun,ivan,kelsey,orlandi}@cs.au.dk, idatucker91@gmail.com, guilhem.castagnos@math.u-bordeaux.fr, fabien.laguillaumie@lirmm.fr

Abstract. We present distributed key generation and decryption protocols for an additively homomorphic cryptosystem based on class groups, improving on a similar system proposed by Braun, Damgård, and Orlandi at CRYPTO '23. Our key generation is similarly constant round but achieves lower communication complexity than the previous work. This improvement is in part the result of relaxing the reconstruction property required of the underlying integer verifiable secret sharing scheme. This eliminates the reliance on potentially costly proofs of knowledge in unknown order groups. We present a new method to batch zero-knowledge proofs in unknown order groups which strengthens these improvements. We also present a protocol which is proven secure against adaptive adversaries in the single inconsistent player (SIP) model. Our protocols are secure in the universal composability (UC) framework and provide guaranteed output delivery. We demonstrate the relative efficiency of our techniques by presenting the running times and communication costs associated with our implementation of the statically secure protocol and provide a direct comparison with alternate state of the art constructions.

Table of Contents

An Improved Threshold Homomorphic Cryptosystem Based on Class Groups . . . 1
Lennart Braun, Guilhem Castagnos, Ivan Damgård, Fabien Laguillaumie, Kelsey Melissaris, Claudio Orlandi, and Ida Tucker

1 Introduction . . . 3
- 1.1 Related Work . . . 3
- 1.2 Technical Overview . . . 4
  - Efficient Verifiable Secret Sharing in the CL Framework . . . 4
  - Efficient Zero-Knowledge Proofs . . . 4
  - Distributed Key Generation and Decryption . . . 5

2 Preliminaries . . . 5
- 2.1 Notation . . . 5
- 2.2 The CL Framework for Unknown Order Groups . . . 5
- 2.3 Zero-Knowledge Proofs . . . 6
- 2.4 Universal Composition with a Single Inconsistent Player . . . 7

3 Secret Sharing and Commitments over the Integers . . . 8
- 3.1 Secret Sharing over the Integers . . . 8
- 3.2 Weak Feldman VSS . . . 8
- 3.3 Pedersen Commitments for the CL Framework . . . 9
- 3.4 Pedersen VSS . . . 11

4 Zero-Knowledge Arguments for Unknown Order Groups . . . 13
- 4.1 Using the Rough-Order Assumption . . . 13
- 4.2 Batched Zero-Knowledge Arguments . . . 14
  - Parameters and Communication Cost . . . 17

5 The Linearly Homomorphic Encryption Scheme . . . 17

6 Protocol with Static Security . . . 18
- 6.1 The Static Protocol . . . 18
- 6.2 Static Security . . . 20
  - Correctness of $\Pi_{\mathsf{TE}}^{\mathsf{static}}$ . . . 20
  - Description of the Simulator in Figure 6 . . . 22
  - Indistinguishability of the Simulation . . . 22

7 Protocol with Adaptive Security . . . 24
- 7.1 Protocol . . . 24
  - Key Generation . . . 24
  - Decryption . . . 24
- 7.2 Security Proof . . . 26
  - Correctness . . . 26
  - Indistinguishability of the Simulation . . . 26
- 7.3 Pedersen Setup Protocol . . . 28

8 Evaluation . . . 29
  - Comparison . . . 30

References . . . 32

1 Introduction

In [CL15], Castagnos and Laguillaumie proposed what is now known as the CL framework. In a nutshell, they proposed a way to construct a special type of class group from which one can build additively homomorphic cryptosystems. A major advantage of these schemes is that we can choose the plaintext space (almost) freely, in contrast to, e.g., Paillier where the plaintext space must be $\mathbb{Z}_N$ where N is an RSA modulus. Moreover, the construction requires no hidden trapdoors, such as the factorization of an RSA modulus. In [BDO23], Braun, Damgård, and Orlandi proposed a threshold version of the CL cryptosystem, including a distributed key generation protocol and showed that it could be used for secure computation. In particular, they proposed an MPC protocol in the "You Only Speak Once" (YOSO) model without requiring trusted set-up.

To get malicious security, [BDO23] uses efficient Σ-protocols for parties to prove correct behaviour, where the zero-knowledge proofs only add a constant factor overhead, compared to just sending the statement to be proved. However, there was one notable exception to this, in the key generation protocol. This protocol makes use of a class group element g_{F} constructed to have large unknown order, and instructs each party P_i to provide an element $g_{\mathsf{F}}^{\alpha_i}$, for a random α_i, and perform a Feldman-style verifiable secret sharing (VSS) of α_i among the parties. The public key is then defined to be $\prod_{i \in S} g_{\mathsf{F}}^{\alpha_i}$ where S is the set of parties which successfully executed the VSS, while the secret key is the corresponding sum of the α_i's [5]. Now, in order to prove that the VSS is secure, P_i is required to give zero-knowledge proof of knowledge (PoK) of α_i, and the security proof makes essential use of the fact that one can extract α_i from the proof given by a corrupt P_i. The only known way to accomplish this without making additional computational assumptions is to use a Σ protocol with binary challenge (the verifier sends only a 1-bit challenge), and repeat this λ times where λ is the security parameter. This is unfortunate, as it is the only place where the zero-knowledge proofs lead to a multiplicative overhead of λ.

In this paper we observe that, while the PoK of α_i is required for the protocol to be a secure VSS, we actually do not need a full-fledged VSS for the key generation protocol to work. We show that by simply dropping the PoK from the protocol and slightly modifying the way the public key is constructed from the $g_{\mathsf{F}}^{\alpha_i}$'s, we get a UC secure key generation protocol, and this modified cryptosystem provides the same security guarantees as that of [BDO23]. In fact, the protocols in [BDO23] were only proven statically secure. In this paper we further present threshold key generation and decryption protocols that are adaptively secure in the single inconsistent player (SIP) model. In this model, the simulator required for security is promised that a certain player will not be corrupted (or equivalently, it is allowed to abort if that player is corrupted). SIP security is obviously not enough for a threshold cryptosystem to be secure on its own. However, the notion remains very useful, since a protocol using a SIP-secure subprotocol can often be shown adaptively secure in the standard sense. An example of this is the adaptively secure MPC protocol from [DN03] that uses a SIP-secure threshold decryption protocol for the Paillier cryptosystem.

1.1 Related Work

We build distributed key generation and decryption protocols for the HSM-CL encryption scheme by Castagnos et al. [CLT18]. The encryption scheme is linearly homomorphic with message space $\mathbb{F}_q$ and based on class groups in the so-called CL framework [CL15]. In [CLT22] Castagnos et al. presented a variant of the HSM-CL encryption scheme with plaintext space $\mathbb{Z}_{2^k}$ and a corresponding threshold scheme. In contrast with the $\mathbb{F}_q$ variant, this does not allow for a transparent setup. As explained above, our work improves upon the statically secure threshold version of HSM-CL for $\mathbb{F}_q$ by Braun et al. [BDO23] where it was used to build MPC. Compared to their construction, we use a weaker reconstruction property of the Feldman VSS allowing us to remove the PoK. Moreover, we also present a protocol that is adaptively secure in the SIP model and apply a batching technique to decrease the costs of the necessary zero-knowledge proofs. In concurrent and independent work Wong et al. [WMC24] also presented a threshold version of HSM-CL. They work with property based security definitions, whereas we provide potentially stronger UC-security results. Moreover, they provide static security only, whereas we consider both static and adaptive security.

[5] This allows the adversary to bias the public key in a limited way, but it is shown in [BDO23] that the resulting cryptosystem is still as secure as if the key was unbiased.

1.2 Technical Overview

Efficient Verifiable Secret Sharing in the CL Framework *Improved Efficiency from Weak Reconstruction.* Adapting distributed key generation protocols from prime order to unknown order groups requires verifiable secret sharing (VSS) over the integers. Braun et al. [BDO23] define a Feldman-style VSS in the CL framework in which a dealer shares the secret α among N parties with Shamir's secret sharing over $\mathbb{Z}$ and broadcasts commitments $C_0, \ldots, C_t$ to the coefficients of the sharing polynomial f for verification. This VSS dealer defines $\mathsf{f}(0) := \Delta \cdot \alpha$ with $\Delta := N!$ to prevent leakage of $\alpha \pmod i$. Reconstruction via interpolation with *integer* Lagrange coefficients incurs yet another multiplicative factor of Δ. Reconstruction first obtains $\overline{\alpha} = \Delta^2 \cdot \alpha$ and subsequently divides by Δ^2 to recover the shared secret α. Verification via evaluation of f in the exponent ensures share consistency modulo the (unknown) group order. One concern which then arises, is that a malicious dealer could distribute shares such that $\overline{\alpha}$ is not divisible by Δ^2 over the integers, causing reconstruction to fail. Braun et al. [BDO23] show that, under the Unknown Order Assumption (*ORD*), this can only happen with negligible probability. They introduce a *proof of knowledge* (PoK) of α for broadcasted $C_0 = g_{\mathsf{F}}^{\alpha}$ to facilitate this reduction. The reduction then extracts α from the PoK and reconstructs $\overline{\alpha}$ from the shares. If these values are unequal then $\Delta^2 \cdot \alpha - \overline{\alpha}$ is a multiple of the group order and a solution to the *ORD* problem.

To avoid the expensive PoK, we prove a weakened property (Lemma 1) that guarantees the reconstruction of $\overline{\alpha}$ such that $C_0^{\Delta^2} = g_{\mathsf{F}}^{\overline{\alpha}}$. We observe that weak reconstruction is sufficient for our threshold encryption scheme so we eliminate the PoK in our modified Feldman VSS.

Statistical Security from Pedersen Commitments. We generalize the above to a statistically private Pedersen-style VSS [Ped92] in the CL framework. To this end, we present Pedersen commitments in the CL framework, as our VSS dealer broadcasts *Pedersen* commitments to the coefficients of f. Consequently, at setup, we need to generate an additional random group element h_{P}, whose discrete logarithm must be unknown. In a prime-order group h_{P} can be sampled uniformly. However, it is unclear how a class group element can be sampled at random without knowing the corresponding discrete logarithm. Hence, we generate h_{P} via a multiparty protocol. This introduces the possibility of adversarial bias over the distribution of h_{P}. We prove security of the commitments when an adversary can induce such bias over the commitment key and use these commitments to construct our Pedersen VSS. We can then prove that this VSS guarantees weak reconstruction and statistical privacy.

Efficient Zero-Knowledge Proofs *Soundness with a Rough Order.* Braun et al. [BDO23] introduced the Rough Order Assumption (RO_C) stating that class groups sampled normally are indistinguishable from those sampled with a C-rough order (that is, the order has no prime factors less than C). Assuming a C-rough group order, efficient Σ-style zero-knowledge proofs were constructed, and shown to be unconditionally set-membership sound (not knowledge sound) as is sufficient for many use cases. Specifically, their result states that no malicious prover can prove a false statement with probability greater than $1/C$. It was then claimed that, under the rough order assumption, the proofs are *computationally sound, even for a normally sampled class group*, i.e., that no malicious PPT prover can prove a false statement with probability greater than $1/C + \mathsf{negl}(\lambda)$.

We observe that, while this claim seems intuitively reasonable, it is not clear how it can be proven as stated: the natural reduction would get a statement and a proof from a malicious prover, and it would guess that the group is sampled normally if the proof verifies and the statement is false, and otherwise guess that the order is C-rough. However, this requires the reduction to decide if the statement is false, and since the statement typically involves discrete logs that the reduction does not know, the decision cannot be done efficiently.

Fortunately, this technicality can be fixed: we show that the zero-knowledge proofs can be used in a larger context *as if they were computationally sound.* The reasoning is as follows: in a higher-level protocol it is usually possible to decide efficiently if it has been broken, and this implies that a reduction to RO_C becomes possible. In more detail, the idea is to prove the higher level protocol secure in a rough-order group using the unconditional soundness of the zero-knowledge proofs. Then one argues that, when using a normal-order group, the adversary's advantage for breaking the high-level protocol is at most negligibly greater. This holds, since otherwise this adversary implies a distinguisher for RO_C.

In Section 4.1 we discuss this in greater detail. Theorem 7 states how to use the RO_C assumption to prove security. This patches the technicality discussed above and implies that the proofs of [BDO23] are, effectively, computationally sound under the RO_C assumption.

Batched Proofs. Protocols often require multiple zero-knowledge proofs for the same relation. For example, a dealer in our VSS must prove that the broadcast values $C_0, \ldots, C_t$ are elements of $\langle g_q \rangle$ with $t+1$ simultaneous proofs of the discrete logarithm base g_q. In Section 4.2 we migrate a technique from [Gen+04] for batched Schnorr proofs in prime-order groups to the unknown-order setting. To batch proofs of b instances the technique computes a polynomial $\mathsf{f}(X) = r + \sum_{k=1}^{b} w_k \cdot X^k$ with coefficients the witnesses $w_1, \ldots, w_b$ and computes a response to the challenge chl as its evaluation $\mathsf{res} = \mathsf{f}(\mathsf{chl})$. From $b+1$ transcripts the witnesses can be reconstructed via Lagrange interpolation. To prove soundness we use integer secret sharing techniques to reconstruct a $C!$-multiple of f which, in a C-rough-order group, implies the existence of the witnesses. Even though res does grow with b, as we cannot perform modular reduction, the batched proofs are still more efficient than b independent proofs.

Distributed Key Generation and Decryption *Allowing More Bias.* Our ideal functionality for threshold encryption $\mathcal{F}_{\mathsf{TE}}$ is adapted from [BDO23]. Following their approach we permit the adversary to bias the public key in order to achieve a more efficient protocol. We generalize the allowed bias. Given an unbiased public key $\mathsf{pk}^*_{\mathsf{cl}} := g_q^{\alpha}$ the adversary specifies secret key bias δ. The resulting public key is $\mathsf{pk}_{\mathsf{cl}} := (\mathsf{pk}^*_{\mathsf{cl}})^a \cdot g_q^{\delta}$ with secret key $\mathsf{sk}_{\mathsf{cl}} := a \cdot \alpha + \delta$ for an arbitrary constant a co-prime to q. We prove that the underlying encryption scheme remains IND-CPA secure with biased keys of this form. In our protocols we set $a := \Delta^2$ which allows us to rely on the *weak* reconstruction properties of the VSS.

Static Protocol. We present a protocol $\Pi^{\mathsf{static}}_{\mathsf{TE}}$ in Section 6 that UC-realizes $\mathcal{F}_{\mathsf{TE}}$ in the presence of an adversary statically corrupting at most $t < N/2$ parties. This protocol is the result of applying the efficiency improvements, specifically the updated Feldman VSS and batched zero-knowledge proofs, discussed above. Notably key generation does not require parties to *prove knowledge* of their contribution to the secret key and instead, during simulation, the adversarial key bias can be computed from the shares held by the honest majority. We implemented the protocol based on BICYCL [Bou+23] and show that its performance compares favorably against threshold Paillier [Fri+23] and [WMC24].

Adaptive Protocol. We then present a protocol $\Pi^{\mathsf{adaptive}}_{\mathsf{TE}}$ in Section 7 that UC-realizes $\mathcal{F}_{\mathsf{TE}}$ under the *adaptive* corruption of at most $t < N/2$ parties in the single inconsistent player (SIP) model. The SIP model requires simulation to produce an indistinguishable view of the environment, including the states of any corrupted parties, and specifies a randomly chosen party that cannot be corrupted. During the first round of distributed key generation, each party P_i samples a contribution to the secret key α_i and shares this contribution with Pedersen VSS. In a second round the parties reveal a multiplicative share of the public key $D_i := g_q^{\alpha_i}$ and prove that the exponent matches their first round commitment $C_{i,0}$. During simulation, the contribution α_h of some randomly chosen party P_h is computed in the exponent from the outputs leaked to the simulator. The view of P_h is *inconsistent* because the simulator cannot compute the discrete logarithm α_h of D_i to include in the party's state. Therefore, simulation fails if P_h is corrupted. The commit-and-reveal strategy for distributed key generation statistically hides the inconsistent party and implies that the probability P_h is corrupted, causing the simulation to fail, is at most $t/N + \mathsf{negl}(\sigma)$.

2 Preliminaries

2.1 Notation

We denote by λ and σ the computational and statistical security parameters, respectively. In our protocols N is the number of parties and the threshold t is a bound on the number of corrupted parties. We define $\Delta := N!$. For integers $a \leq b$ we write $[a, b] := \{a, \ldots, b\}$, $[a, b) := \{a, \ldots, b-1\}$, and $[b] := [1, b]$. We denote the uniform distribution over a finite set M with $\mathcal{U}(M)$.

2.2 The CL Framework for Unknown Order Groups

We use class groups as a black box with the so-called CL framework for groups of unknown order [CL15] that specifies two algorithms, CLGen and CLSolve. The former, CLGen, on input the security parameter

1^λ and a prime $q \geq 2^\lambda$, outputs a set of public parameters $\mathsf{pp_{cl}} = (q, \bar{s}, f, g_q, \widehat{G}, F; \rho)$ which describe a class group, where ρ is the randomness used by CLGen . In this tuple the group of squares $\widehat{G}$ contains a cyclic subgroup $G \subseteq \widehat{G}$ which factors into the direct product $G = G^q \times F \subseteq \widehat{G}$ of $F = \langle f \rangle$, the unique subgroup of order q, and $G^q = \langle g_q \rangle$, the subgroup of qth powers. The order of G^q is unknown but an upper bound $\bar{s} > |\widehat{G}^q|$ is given. The latter, $\mathsf{CLSolve}$, efficiently computes discrete logarithms in the subgroup F.

Our protocols are given in the $\mathcal{F}_{\mathsf{CL}}$-hybrid model (given in Figure 1). Parties query $\mathcal{F}_{\mathsf{CL}}$ during initialization to generate $\mathsf{pp_{cl}}$. Our constructions also make use of a distribution $\mathcal{D}_q$ over $\mathbb{Z}$ that induces an almost-uniform distribution $\{g_q^x \mid x \leftarrow \mathcal{D}_q\}$ in G^q. We can instantiate it by sampling from $\mathcal{U}([2^\sigma \cdot \bar{s}])$.

Figure 1: CL Parameter Generation Functionality $\mathcal{F}_{\mathsf{CL}}$

Gen On input $(\mathsf{Gen}, 1^\lambda, q)$ from all parties, $\mathcal{F}_{\mathsf{CL}}$ runs the CL setup algorithm $\mathsf{pp_{cl}} \leftarrow \mathsf{CLGen}(1^\lambda, q)$ with randomness ρ, then stores and outputs the public parameters $\mathsf{pp_{cl}} = (q, \bar{s}, f, g_q, \widehat{G}, F; \rho)$.

In this work we construct a threshold version of the HSM-CL encryption scheme [CLT18] which we informally discuss briefly here. The public key has the form $\mathsf{pk_{cl}} = g_q^{\mathsf{sk_{cl}}}$ and the secret key is sampled as $\mathsf{sk_{cl}} \leftarrow \mathcal{D}_q$. Messages $m \in \mathbb{F}_q$ are encrypted as $\mathsf{ct} = (g_q^r, \mathsf{pk_{cl}}^r \cdot f^m) \leftarrow \mathsf{Enc}(\mathsf{pk_{cl}}, m; r)$ and ciphertext can be decrypted as $m = \mathsf{CLSolve}(\mathsf{ct}_2 \cdot \mathsf{ct}_1^{-\mathsf{sk_{cl}}})$. The HSM-CL encryption scheme is linearly homomorphic and its security is proven under the hard subgroup membership assumption (HSM) [CLT18], stating that random elements of G are indistinguishable from random elements of G^q.

The security of our protocols is based on the Unknown Order assumption (ORD) and the Rough Order assumption (RO_C) introduced by [BDO23]. The ORD assumption states that no efficient adversary can compute a multiple of $\mathrm{ord}(h) = e \neq 0$ for any group element $h \in (\widehat{G} \setminus F)$. The RO_C assumption states that class groups of rough order, i.e., of an order with no small prime factors, are indistinguishable from those sampled normally.

Definition 1 (Unknown Order Assumption [Cou+21], version from [BDO23]). *Let λ be the security parameter, $q \geq 2^\lambda$ a prime and $\mathcal{A}$ a PPT algorithm. The experiment generates public parameters $\mathsf{pp_{cl}} := (q, \bar{s}, f, g_q, \widehat{G}, F; \rho) \leftarrow \mathsf{CLGen}(1^\lambda, q)$ and runs $\mathcal{A}(\mathsf{pp_{cl}})$. We say that $\mathcal{A}$ solves the* unknown order (ORD) problem *if it outputs a group element $h \in (\widehat{G} \setminus F)$[6] and an integer $e \neq 0$ such that $h^e = 1$. We define the advantage $\mathsf{Adv}_{\mathcal{A}}^{ORD}(\lambda)$ as the success probability. The ORD assumption holds if $\mathsf{Adv}_{\mathcal{A}}^{ORD}(\lambda)$ is negligible in λ for all PPT $\mathcal{A}$.*

Definition 2 (Rough Order Assumption [BDO23]). *Let λ be the security parameter, $q \geq 2^\lambda$ a prime, $C \in \mathbb{N}$ and $\mathcal{A}$ a PPT algorithm. Define $\mathcal{D}_C^{\mathsf{rough}}(\lambda, q)$ as the uniform distribution over the set $\{\rho \in \{0,1\}^\lambda \mid (q, \bar{s}, f, g_q, \widehat{G}, F; \rho) \leftarrow \mathsf{CLGen}(1^\lambda, q; \rho) \wedge (\forall \text{ primes } p < C \ : \ p \nmid \mathrm{ord}(\widehat{G}))\}$. We say that $\mathcal{A}$ solves the C-*rough order (RO_C) problem *if the advantage below is non-negligible.*

$$\mathsf{Adv}_{\mathcal{A}}^{RO_C}(\lambda) := \left| \Pr\left[1 \leftarrow \mathcal{A}(1^\lambda, \mathcal{U}(\{0,1\}^\lambda))\right] - \Pr\left[1 \leftarrow \mathcal{A}(1^\lambda, \mathcal{D}_C^{\mathsf{rough}}(\lambda, q))\right] \right|$$

We say the RO_C assumption holds if $\mathsf{Adv}_{\mathcal{A}}^{RO_C}(\lambda) \leq \mathsf{negl}(\lambda)$ for all PPT algorithms $\mathcal{A}$.

2.3 Zero-Knowledge Proofs

Our protocols use zero-knowledge proofs for a number of relations between class group elements. Most of the time *(set-membership) soundness* is sufficient as we do not need to extract a witness. We require

[6] This can by easily verified by checking $h^q \neq 1$.

proofs for the following relations:

$$\mathsf{R}_{\mathsf{DLog}} := \left\{(g,h);x \;\middle|\; g,h \in \widehat{G} \wedge x \in \mathbb{Z} \wedge h = g^x\right\} \tag{1}$$

$$\mathsf{R}_{\mathsf{EqDLog}} := \left\{(g_1,g_2,h_1,h_2);x \;\middle|\; ((g_i,h_i);x) \in \mathsf{R}_{\mathsf{DLog}} \text{ for } i=1,2\right\} \tag{2}$$

$$\mathsf{R}_{\mathsf{Ped}} := \left\{(g,h,C);(m,r) \;\middle|\; g,h,C \in \widehat{G} \wedge m,r \in \mathbb{Z} \wedge C = g^r \cdot h^m\right\} \tag{3}$$

$$\mathsf{R}_{\mathsf{Ped\text{-}DLog}} := \left\{(g,h,C,D);(m,r) \;\middle|\; \begin{array}{l}((g,h,C);(m,r)) \in \mathsf{R}_{\mathsf{Ped}} \\ \wedge\, ((g^{\Delta^2},D);m) \in \mathsf{R}_{\mathsf{DLog}}\end{array}\right\} \tag{4}$$

The simplest option is to use the Σ-like protocols by Braun et al. [BDO23] for statements of the form $Y = \prod_k X_k^{w_k}$ where $Y, X_k \in \widehat{G}$. In Section 4.2 we show how to extend their approach to obtain more efficient proofs for batched relations. The proofs are unconditionally sound if $\mathrm{ord}(\widehat{G})$ is C-rough and, under the RO_C assumption, can be used to prove security of a higher level protocol even when the class group is sampled normally. We provide a more detailed discussion about soundness under the RO_C assumption in Section 4.1. When knowledge soundness is required we use binary challenges with repetitions.

To simplify the description and reduce the round complexity of our protocols we apply the Fiat-Shamir transform [FS87] for non-interactivity. The single exception is during the one-time setup of our adaptively secure protocol, where we need a straight-line extractable proof of knowledge (see Section 7.3) and apply the Fischlin transformation [Fis05].

In the descriptions of our protocols we use the following abstract notation for non-interactive proof systems of a witness relation $\mathsf{R}_{\mathsf{Rel}}$. For the statement-witness pair $(x;w) \in \mathsf{R}_{\mathsf{Rel}}$ a proof is computed as $\pi \leftarrow \mathsf{Prove}(x,w)$ and is verified with $\{\top,\bot\} \leftarrow \mathsf{Verify}(x,\pi)$.[7] A simulated proof is the output $\pi \leftarrow \mathsf{SimProve}(x)$ and implicitly includes programming the random oracle in the Fiat-Shamir case. We denote a zero-knowledge proof by $\Pi^{\mathsf{ZK}}_{\mathsf{Rel}}$, a zero-knowledge proof of knowledge by $\Pi^{\mathsf{PoK}}_{\mathsf{Rel}}$ and a straight-line extractable zero-knowledge proof of knowledge by $\Pi^{\mathsf{PoK,sle}}_{\mathsf{Rel}}$. The straight-line extractor is written as $w \leftarrow \Pi^{\mathsf{PoK,sle}}_{\mathsf{Rel}}.\mathsf{Extract}(x,\pi)$. For zero-knowledge proofs of batched relations we write $\overline{\Pi}^{\mathsf{ZK}}_{\mathsf{Rel}}$. While this can be instantiated using the usual AND-proof of Σ-like protocols we show a more efficient approach in Section 4.2.

2.4 Universal Composition with a Single Inconsistent Player

We prove security in the universal composition (UC) framework [Can01]. We very briefly recall the UC model to communicate the complexity of adaptive corruption and then describe the single inconsistent player (SIP) model.

In the UC framework a protocol Π is compared against the ideal functionality $\mathcal{F}$ which essentially carries out a computation as a trusted third party. Parties can determine their output by either executing the protocol Π, as in the "real world", or querying the functionality $\mathcal{F}$, as in the "ideal world." A protocol is considered UC-secure if it is proven to be *as good as* the ideal functionality, i.e., that an instance of Π is indistinguishable from, and therefore can be used in place of, an instance of $\mathcal{F}$. The composition theorem then guarantees that such a protocol Π can be arbitrarily composed with other protocols.

The attacker, attempting to distinguish an execution of Π in the real world from an invocation of $\mathcal{F}$ in the ideal world, is our environment $\mathcal{Z}$. In both the real and ideal worlds $\mathcal{Z}$ provides input to and receives output from the honest parties and controls an adversary $\mathcal{A}$ against an execution of Π. In the ideal world this execution of Π is simulated by an ideal adversary $\mathcal{S}$ that computes honest party messages from any leakage allowed by $\mathcal{F}$.

To corrupt a party P_i the environment issues a corruption instruction and receives the internal state of the freshly corrupted party.[8] A static corruption instruction is issued prior to any protocol messages and thus $\mathcal{S}$ does nothing more than not simulate the corrupted party. Upon adaptive corruption $\mathcal{S}$ must produce the internal state of the corrupted party. Moreover, this state must be consistent with the protocol transcript meaning that it must contain all of the secrets and randomness used to compute the

[7] When implemented in practice the non-interactive proofs should take context, including identifiers for the session, the party, and the step of the protocol, as additional input to avoid replay attacks, etc.

[8] We omit the details of the exact path of this corruption instruction as it is not completely necessary to present a high level description of the difficulty of adaptive corruption.

messages sent by the party. Often, $\mathcal{S}$ cannot compute the state of *every* honest party as the messages of at least one party were not computed according to Π but instead computed to ensure that the protocol output corresponds with that of $\mathcal{F}$. The single inconsistent player (SIP) model [Can+99; AF04] permits $\mathcal{S}$ to simulate the messages of a random honest party P_h with the guarantee that P_h remains honest and simulation fails if P_h is corrupted by $\mathcal{Z}$. In the case of t-limited adversaries the success probability of $\mathcal{S}$ is reduced by a factor of t/N.

Our adaptively secure distributed key generation and threshold decryption protocol $\Pi_{\mathsf{TE}}^{\mathsf{adaptive}}$ is SIP-UC-secure. As discussed in [AF04] SIP-UC security allows us to avoid an assumption of secure erasure or adaptively secure communication channels, for example non-committing encryption. The SIP-UC composition theorem is restricted; it guarantees arbitrary composition with protocols executed by the same set of parties with the same inconsistent player.

3 Secret Sharing and Commitments over the Integers

3.1 Secret Sharing over the Integers

We recall Shamir's secret sharing over the integers as defined in Braun et al. [BDO23]. With the processes appearing in Definition 3 a secret α sampled from the bounded interval $[0, 2^\ell)$ can be shared among N parties such that α can be efficiently recovered from any $t+1$ distinct shares yet remains statistically private given any t or fewer shares.

Definition 3 (Shamir's Secret Sharing over $\mathbb{Z}$ [BDO23]). *Let N be the number of parties, t the corruption threshold, ℓ a bound on the secret size, and σ a statistical security parameter. Moreover, let $\ell_0 \in \mathbb{N}$ be a parameter. To share a secret $\alpha \in [0, 2^\ell)$ the dealer proceeds as follows:*

1. *Let $\tilde{\alpha} := \alpha \cdot \Delta$ with $\Delta := N!$.*
2. *Sample $\mathbf{r} = (r_1, \ldots, r_t) \in_R [0, 2^{\ell_0+\sigma})^t$*
3. *Set $\mathsf{f}(X) := \tilde{\alpha} + r_1 \cdot X + \cdots + r_t \cdot X^t$.*
4. *Send $y_i := \mathsf{f}(i)$ over a private channel to party P_i for $i \in [N]$.*

We denote the above process, in which α is shared using the random coins $\mathbf{r}$, as $\mathsf{Share}^\ell(\alpha, \mathbf{r})$ and denote the share of P_i as $\mathsf{Share}_i^\ell(\alpha, \mathbf{r})$. If it is clear from the context, we might omit ℓ.

Following [BDO23] and [Fri+23], the secret sharing scheme provides statistical privacy if $\ell_0 > \ell + \log_2(\Delta) + 2 \cdot \log_2(t+1) + 2$.

Given any subset of at least $t+1$ shares $\{(x_i := i, y_i)\}_{i \in S}$ held by a subset of parties $S \subseteq [N]$ with $|S| \geq t+1$, the sharing polynomial can be recovered via Lagrange interpolation over the integers. The Lagrange coefficients $L_i^S(X) \in \mathbb{Z}$ are scaled by $\Delta = N!$ as in Equation (5) to guarantee that computation occurs over $\mathbb{Z}$.

$$\Delta \cdot \mathsf{f}(X) = \sum_{i \in S} y_i \cdot L_i^S(X) \quad \text{with} \quad L_i^S(X) = \Delta \cdot \prod_{j \in S \setminus \{i\}} \frac{x_j - X}{x_j - x_i} \tag{5}$$

This process allows parties to recover $\Delta \cdot \mathsf{f}(0) = \Delta^2 \cdot \alpha$ and subsequently α through division by Δ^2. Note that this is only well-defined if a dealer behaves honestly. Variants of verifiable secret sharing (VSS) that ensure correctness with a potentially malicious dealer are presented in the following sections.

3.2 Weak Feldman VSS

In Definition 4 we recall a Feldman-style verifiable secret sharing (VSS) scheme. The variant of Feldman VSS appearing in this section has been modified from its original form in [BDO23] as discussed below. Lemma 1 guarantees a weakened reconstruction property, that every set of $t+1$ shares can be used to recover the same multiple of the shared secret. Without change, Lemma 2 from [BDO23] guarantees security. In the following we assume that the public parameters $\mathsf{pp}_{\mathsf{cl}}$ are given along with a designated group element $g_{\mathsf{F}} \in (\widehat{G} \setminus F)$.

Definition 4 (Feldman VSS (Protocol 2 in [BDO23]). *In Feldman VSS the dealer shares a secret $\alpha \in [0, 2^\ell)$ via $(y_1, \ldots, y_N) \leftarrow \mathsf{Share}(\alpha; \mathbf{r})$ (Definition 3), privately distributes the share y_i to each P_i, and, for $k \in [t]$ and $\Delta := N!$, broadcasts values $C_0 := g_{\mathsf{F}}^\alpha$ and $C_k := g_{\mathsf{F}}^{\Delta \cdot r_k}$ and proves that $C_0, \ldots, C_t \in \langle g_{\mathsf{F}} \rangle$. The sharing procedure is denoted $(y_1, \ldots, y_N; C_0, \ldots, C_t) \leftarrow \mathsf{F\text{-}Share}(\alpha; \mathbf{r}; g_{\mathsf{F}})$.*

Party P_i can verify the share y_i by checking Equation (6), and broadcasts a complaint against the dealer if verification fails. Verification with respect to Equation (6) is denoted $\{\bot, \top\} \leftarrow$ F-Check$(i, y_i; C_0, \ldots, C_t; g_{\mathsf{F}})$.

$$g_{\mathsf{F}}^{\Delta \cdot y_i} \stackrel{?}{=} C_0^{\Delta^2} \cdot \prod_{k=1}^{t} (C_k)^{(i^k)} \tag{6}$$

Equation (6) can also be used to verify that parties publish correct shares during reconstruction.

The protocol presented in [BDO23] requires that the dealer broadcast a *proof of knowledge* of the discrete logarithm of C_0 to base g_{F} to ensure that a secret shared via F-Share can be efficiently reconstructed from any set of $t+1$ verifying shares. Therein the strong reconstruction property requires that the exact value α shared by the dealer is output by reconstruction. To recover α the intermediate value $\Delta \cdot \mathsf{f}(0)$ is interpolated and must be divisible by Δ^2 over $\mathbb{Z}$ for reconstruction to succeed.[9] If reconstruction fails then the PoK can be leveraged to prove that such a dealer can be used to break *ORD*, specifically because the difference between the witness that can be extracted from the verifying proof and the reconstructed secret will be a multiple of the group order.

In this work we eliminate the (potentially) costly PoK by proving a weaker reconstruction property that is ultimately sufficient for our goal of distributed key generation. The weak reconstruction property in Lemma 1 proves that the interpolated value $\overline{\alpha}$ directly satisfies $C_0^{\Delta^2} = g_{\mathsf{F}}^{\overline{\alpha}}$. We also use the Feldman Simulation Lemma 2 from [BDO23].

Lemma 1 (Weak Feldman Reconstruction). *If* $\gcd(\Delta, \mathrm{ord}(g_{\mathsf{F}})) = 1$ *then the following conditions are satisfied:*

(i) There exists an efficient algorithm which on input $C_0, \ldots, C_t \in \langle g_{\mathsf{F}} \rangle$ and $t+1$ values $y_{i_1}, \ldots, y_{i_{t+1}}$ satisfying Equation (6) outputs a value $\overline{\alpha} \in \mathbb{Z}$ such that $C_0^{\Delta^2} = g_{\mathsf{F}}^{\overline{\alpha}}$.[10]

(ii) Under the ORD assumption any collection of $t+1$ shares which satisfy Equation (6) can be used to recover the same value $\overline{\alpha} \in \mathbb{Z}$.

Proof. (i) The lemma follows from the proof of the "strong" variant appearing in [BDO23, Lemma 6 in the full version]. Let $\mathsf{h} \in \mathbb{Q}[X]$ be the unique polynomial of degree at most t such that $\mathsf{h}(i_j) = y_{i_j}$ for $j \in [t+1]$. Set $S := \{i_1, \ldots, i_{t+1}\}$ and let $L_j^S := L_j^S(0)$ be the integer Langrange interpolation coefficients multiplied by Δ that can be used to reconstruct $h(0)$ from its values in $i_1, \ldots, i_{t+1}$ as in Equation (5). We can efficiently compute the integer $\Delta \cdot \mathsf{h}(0) = \sum_{j=1}^{t+1} L_j^S \cdot y_{i_j}$ and output $\overline{\alpha} := \Delta \cdot h(0)$ directly.[11]

(ii) Follows from the same argument as in [BDO23]. □

Lemma 2 (Feldman Simulation [BDO23]). *Given $C_0 \in \langle g_{\mathsf{F}} \rangle$ and t points $(x_i, y_i) \in \mathbb{Z}^2$ for some $1 \leq x_1 < \cdots < x_t \leq N$, one can efficiently compute $C_1, \ldots, C_t \in \langle g_{\mathsf{F}} \rangle$ such that* F-Check$(x_i, y_i; C_0, \ldots, C_t, g_{\mathsf{F}})$ *is satisfied for all $i \in [t]$. Moreover, the values $C_1, \ldots, C_t \in \langle g_{\mathsf{F}} \rangle$ have the same distribution as produced by* F-Share *conditioned on C_0 and $(x_i, y_i)_{i \in [t]}$.*

3.3 Pedersen Commitments for the CL Framework

The standard Pedersen commitment [Ped92] is defined in a prime-order group and we define its adaptation to the CL framework. Previously Damgård et al. [DF02] constructed a similar commitment scheme for groups with unknown order, though in a slightly different setting. Couteau et al. [Cou+21] also considered Pedersen-like homomorphic integer commitments in class groups.

We first recall Definition 5 of a statistically hiding and computationally binding commitment scheme.

Definition 5 (Commitment Scheme). *A* commitment scheme *for the message space $\mathcal{M}$ is a pair of polynomial-time algorithms* $(\mathsf{Gen}, \mathsf{Com})$. *The former,* Gen, *is a probabilistic algorithm which takes as input the security parameter 1^λ and outputs a public key* pk. *The latter,* $\mathsf{Com}_{\mathsf{pk}}$ *parameterized by public key* pk,[12] *takes as input a message $m \in \mathcal{M}$ and randomness r sampled from some given distribution, and*

[9] The fact that $\mathsf{f}(0) = \alpha \cdot \Delta$ as in Share accounts for one factor of Δ while the integer Lagrange coefficients, scaled by Δ as in Equation (5), contribute the second factor.

[10] The respective (strong) lemma in [BDO23] has the additional precondition of a *proof of knowledge* for C_0, but provides the additional guarantee that $\alpha \in \mathbb{Z}$ is reconstructed such that $C_0 = g_{\mathsf{F}}^{\alpha}$.

[11] In contrast, [BDO23] proceed to show that $g_{\mathsf{F}}^{\Delta \cdot \mathsf{h}(0)} = C_0^{\Delta^2}$ and that $\Delta \cdot \mathsf{h}(0)$ is divisible by Δ^2 but the remainder of this portion of their proof is not required here.

[12] If the used public key pk is clear from the context, we write Com instead of $\mathsf{Com}_{\mathsf{pk}}$.

outputs a commitment $\mathsf{cm} \leftarrow \mathsf{Com}_{\mathsf{pk}}(m; r)$. *A commitment is opened by revealing* (m, r). *The opening can be verified by checking that* $\mathsf{Com}_{\mathsf{pk}}(m; r) \stackrel{?}{=} \mathsf{cm}$ *holds. A commitment scheme is* statistically hiding *if, for all* (m, m') *and independent* (r, r'), *the commitments* $\mathsf{Com}_{\mathsf{pk}}(m; r), \mathsf{Com}_{\mathsf{pk}}(m'; r')$ *are statistically indistinguishable. A commitment scheme is* computationally binding *if no PPT algorithm can produce* $(m, m'), (r, r')$ *with* $m \neq m'$ *such that* $\mathsf{Com}_{\mathsf{pk}}(m; r) = \mathsf{Com}_{\mathsf{pk}}(m'; r')$ *except with negligible probability.*

Definition 6 (Pedersen Commitments for CL). *Let* $\mathcal{M} = \mathbb{Z}^n$ *for some* $n \in \mathbb{N} \setminus \{0\}$ *be the message space and denote the commitment public parameters by* pp *which include CL parameters* $\mathsf{pp}_{\mathsf{cl}}$.

$\mathsf{Gen}(1^\lambda) \to \mathsf{pk}$. *Sample* $\mathsf{pp}_{\mathsf{cl}} \leftarrow \mathsf{CLGen}(1^\lambda, q)$ *and* $\omega_1, \ldots, \omega_n \leftarrow \mathcal{D}_q$, *compute* $\mathbf{h}_{\mathsf{P}} := (h_{\mathsf{P},1}, \ldots, h_{\mathsf{P},n}) = (g_q^{\omega_1}, \ldots, g_q^{\omega_n})$, *and output* $\mathsf{pk} := (\mathbf{h}_{\mathsf{P}}, \mathsf{pp}_{\mathsf{cl}})$.

$\mathsf{Com}_{\mathsf{pk}}(\mathbf{m}; r) \to \mathsf{cm}$. *On input a message* $\mathbf{m} = (m_1, \ldots, m_n) \in \mathbb{Z}^n$ *and randomness* $r \in_R \mathcal{D}_q$, *output* $\mathsf{cm} := g_q^r \cdot \prod_{i \in [n]} (h_{\mathsf{P},i})^{m_i}$.

An opening $(\mathbf{m}, r)$ *can be verified by checking that* $g_q^r \cdot \prod_{i \in [n]} (h_{\mathsf{P},i})^{m_i} \stackrel{?}{=} \mathsf{cm}$.

Pedersen commitments require a public commitment key $\mathbf{h}_{\mathsf{P}}$ generated at setup. Elements of the public key are supposed to be almost-uniformly random in G^q and therefore we encounter another instance of the same problem as in distributed key generation; an adversary can bias the public key. To generate an unbiased random group element Gennaro et al. [Gen+07] use Pedersen VSS which, in turn, requires a public key for Pedersen commitments. To break this circle we show that Pedersen commitments in the CL framework remain secure with a biased key.

$\underline{\mathsf{BiasedComGen}^{\mathcal{A}}(\mathsf{pp}, n)}$

1. Sample $\boldsymbol{\omega}' := (\omega'_1, \ldots, \omega'_n) \leftarrow (\mathcal{D}_q)^n$.
2. Set $\mathbf{h}^*_{\mathsf{P}} := \left(h^*_{\mathsf{P},1} = g_q^{\omega'_1}, \ldots, h^*_{\mathsf{P},n} = g_q^{\omega'_n}\right)$.
3. $\boldsymbol{\delta} \leftarrow \mathcal{A}(\mathsf{pp}, \mathbf{h}^*_{\mathsf{P}})$.
4. Define $\boldsymbol{\omega} := \boldsymbol{\omega}' + \boldsymbol{\delta}$.
5. Set $\mathbf{h}_{\mathsf{P}} := \left(h_{\mathsf{P},1} = g_q^{\omega_1}, \ldots, h_{\mathsf{P},n} = g_q^{\omega_n}\right)$.
6. Output $(\mathbf{h}_{\mathsf{P}}, \boldsymbol{\omega})$.

Fig. 2. Modified commitment key generation that permits limited adversarial bias.

The procedure $\mathsf{BiasedComGen}^{\mathcal{A}}$ samples n exponents from $\mathcal{D}_q$ and computes the corresponding n nearly uniform group elements $\mathbf{h}^*_{\mathsf{P}}$. The adversary, on input pp and $\mathbf{h}^*_{\mathsf{P}}$, outputs a vector of exponents $\boldsymbol{\delta}$ to bias the key.

Definition 7 (Pedersen Commitments for CL with Biased Public Key). *Pedersen Commitments as in Definition 6, but with the interactive procedure* $\mathsf{BiasedComGen}^{\mathcal{A}}$ *(Figure 2) instead of* Gen *to allow an adversary* $\mathcal{A}$ *to bias the public key* $\mathbf{h}_{\mathsf{P}}$.

Theorem 3. *Under the ORD assumption (Definition 1) Pedersen commitments for CL with biased public key are statistically hiding and computationally binding.*

Proof.

Hiding. The commitment randomness is sampled according to the distribution $r \leftarrow \mathcal{D}_q$ and so the value g_q^r is almost uniformly distributed in G^q. For all $i \in [n]$ the element $h_{\mathsf{P},i} \in \langle g_q \rangle$ so every commitment $\mathsf{cm} = g_q^r \cdot \prod_{i \in [n]} h_{\mathsf{P},i}^{m_i} \in \langle g_q \rangle$ is also almost uniformly distributed. For every message a commitment to the message is almost uniformly distributed in the subgroup and therefore the messages are statistically hidden.

Binding. Assume that there exists a PPT adversary $\mathcal{A}$ that outputs $(\mathbf{m}, \mathbf{m}', r, r')$ with $\mathbf{m} \neq \mathbf{m}'$ such that $g_q^r \cdot \prod_{i \in [n]} h_{\mathsf{P},i}^{m_i} = g_q^{r'} \cdot \prod_{i \in [n]} h_{\mathsf{P},i}^{m'_i}$, with non-negligible probability. We define the following reduction to the *ORD* problem.

On input $\mathsf{pp}_{\mathsf{cl}}$ sample $\omega_1, \ldots, \omega_n \in_R [\tilde{s} \cdot 2^\sigma]$ and run $\mathsf{BiasedComGen}^{\mathcal{A}}$ with $\mathcal{A}$ to obtain $\mathbf{h}_{\mathsf{P}}$ with $h_{\mathsf{P},i} = g_q^{\omega_i + \delta_i}$ for $i \in [n]$. Then run $\mathcal{A}$ on input $\mathsf{pk} := (\mathsf{pp}_{\mathsf{cl}}, \mathbf{h}_{\mathsf{P}})$. If $\mathcal{A}$ outputs $(\mathbf{m}, \mathbf{m}', r, r')$ as above then:

$$r + \sum_{i \in [n]} \omega_i \cdot m_i = r' + \sum_{i \in [n]} \omega_i \cdot m'_i \pmod{\mathrm{ord}(g_q)}$$
$$\iff 0 = (r - r') + \sum_{i \in [n]} \omega_i \cdot (m_i - m'_i) \pmod{\mathrm{ord}(g_q)}.$$

Let $M := (r - r') + \sum_{i \in [n]} \omega_i \cdot (m_i - m'_i) \in \mathbb{Z}$. By the above $g_q^M = 1$. Thus if M is non-zero then (g_q, M) is a solution to the *ORD* problem.

By the inequality $\mathbf{m} \neq \mathbf{m}'$ there exists $i^* \in [n]$ such that $m_{i^*} \neq m'_{i^*}$. Write $\omega_{i^*} = s \cdot \mathrm{ord}(g_q) + t + \delta_{i^*}$ where $t \in [0, \mathrm{ord}(g_q))$ and δ_{i^*} is the bias specified by $\mathcal{A}$ during $\mathsf{BiasedComGen}^{\mathcal{A}}$. Since both $\mathrm{ord}(g_q)$ and $m_{i^*} - m'_{i^*}$ are non-zero we can write, for $o = \mathrm{ord}(g_q)$:

$$M = 0 \iff$$
$$s \cdot o \cdot (m_{i^*} - m'_{i^*}) = (r' - r) + (t + \delta_{i^*}) \cdot (m'_{i^*} - m_{i^*}) + \sum_{i \in [n] \setminus \{i^*\}} \omega_i \cdot (m'_i - m_i)$$
$$\iff s = \frac{(r' - r) + (t + \delta_{i^*}) \cdot (m'_{i^*} - m_{i^*}) + \sum_{i \in [n] \setminus \{i^*\}} \omega_i \cdot (m'_i - m_i)}{o \cdot (m_{i^*} - m'_{i^*})}.$$

The values t and δ_{i^*} constitute the only information that $\mathcal{A}$ receives about ω_{i^*} so $\mathcal{A}$'s output, and the values on the right-hand side of the equation, are independent of s. The value s has at least σ bits of entropy. Hence, $M \neq 0$ and (g_q, M) is a solution to *ORD* except with negligible probability. We conclude that no adversary can break the binding of the commitment with non-negligible probability. □

3.4 Pedersen VSS

In this section we adapt Pedersen's verifiable secret sharing scheme [Ped92] to the CL framework. Previously, Pedersen VSS was considered in the RSA setting by Canetti et al. [Can+99] and Frankel et al. [FMY99]. Our Pedersen VSS satisfies a weak reconstruction property (Lemma 4) analogous to Lemma 1 for Feldman VSS, but also provides *statistical* privacy (Lemma 5).

Definition 8 (Pedersen VSS). *Assume that public parameters $\mathsf{pp}_{\mathsf{cl}}$ and h_{P} are given such that $h_{\mathsf{P}} \in_R \langle g_q \rangle$. Let $\ell \in \mathbb{N}$ denote the bit size of the secret and let $\ell_b \in \mathbb{N}$ an additional parameter. To share a secret $\alpha \in [0, 2^\ell)$, the dealer samples a value $\beta \in_R [0, 2^{\ell_b})$ and shares both values with $(y_1, \ldots, y_N) \leftarrow \mathsf{Share}^{\ell}(\alpha; \mathbf{r})$ and $(z_1, \ldots, z_N) \leftarrow \mathsf{Share}^{\ell_b}(\beta; \mathbf{s})$ (see Definition 3). The dealer also broadcasts values $C_0 := g_q^\beta \cdot h_{\mathsf{P}}^\alpha$ and $C_k := g_q^{s_k} \cdot h_{\mathsf{P}}^{r_k}$ for $k \in [t]$.*

Party P_i verifies the received share (y_i, z_i) by checking Equation (7) and, if verification fails, broadcasts a complaint.

$$g_q^{z_i} \cdot h_{\mathsf{P}}^{y_i} \stackrel{?}{=} C_0^\Delta \cdot \prod_{k=1}^{t} (C_k)^{(i^k)} \tag{7}$$

The same method can be used to verify shares during reconstruction.

We denote the sharing procedure by $((y_1, z_1), \ldots, (y_N, z_N); C_0, \ldots, C_t) \leftarrow \mathsf{P\text{-}Share}(\alpha; \beta, \mathbf{r}, \mathbf{s})$, writing the randomness $(\beta, \mathbf{r}, \mathbf{s})$ explicitly, where the output (y_i, z_i) is the share of P_i and the C_k are public. We denote the checking procedure by $\{\bot, \top\} \leftarrow \mathsf{P\text{-}Check}(i, y_i, z_i; C_0, \ldots, C_t)$.

Lemma 4 (Weak Pedersen Reconstruction).

(i) Given $C_0, \ldots, C_t \in \langle g_q \rangle$, and $t+1$ shares $(y_{i_1}, z_{i_1}), \ldots, (y_{i_{t+1}}, z_{i_{t+1}})$ such that Equation (7) holds there exists an efficient algorithm which parties can use to compute $\overline{\alpha}, \overline{\beta} \in \mathbb{Z}$ such that $C_0^{\Delta^2} = g_q^{\overline{\beta}} \cdot h_{\mathsf{P}}^{\overline{\alpha}}$.

(ii) Under the ORD assumption any collection of $t+1$ shares which satisfy Equation (7) can be used to recover the same pair $\overline{\alpha}, \overline{\beta} \in \mathbb{Z}$.

Proof. (i) Let $\omega \in \mathbb{Z}$ such that $h_{\mathsf{P}} = g_q^{\omega}$. By Lagrange interpolation there is are unique polynomials $\bar{\mathsf{f}}, \bar{\mathsf{g}} \in \mathbb{Q}[X]$ of degree at most t such that $\bar{\mathsf{f}}(i_j) = y_{i_j}$ and $\bar{\mathsf{g}}(i_j) = z_{i_j}$ for all $j \in [t+1]$. Using Lagrange interpolation over the integers, we can compute values $\overline{\alpha} := \Delta \cdot \bar{\mathsf{f}}(0), \overline{\beta} := \Delta \cdot \bar{\mathsf{g}}(0) \in \mathbb{Z}$ from the shares. Set $C'_0 := C_0^{\Delta}$ and $C'_k := C_k$ for $k \in [t]$. Write $C'_k := g_q^{\gamma_k}$ for some $\gamma_k \in \mathbb{Z}$ and $k \in [0,t]$, and define $\mathsf{h}'(X) := \sum_{k=0}^{t} \gamma_k \cdot X^k$. Then we have

$$g_q^{\bar{\mathsf{g}}(i_j)} \cdot h_{\mathsf{P}}^{\bar{\mathsf{f}}(i_j)} \overset{(7)}{=} C_0^{\Delta} \cdot \prod_{k=1}^{t} C_k^{(i_j^k)} = \prod_{k=0}^{t} C_k'^{(i_j^k)} = g_q^{\mathsf{h}'(i_j)} \qquad \text{for all } j \in [t+1].$$

By integer interpolation in the exponent, we get $g_q^{\Delta \cdot \bar{\mathsf{g}}(0)} \cdot h_{\mathsf{P}}^{\Delta \cdot \bar{\mathsf{f}}(0)} = g_q^{\Delta \cdot \mathsf{h}'(0)} = C_0'^{\Delta} = C_0^{\Delta^2}$. Hence, $\overline{\alpha}, \overline{\beta}$ satisfy the lemma.[13]

(ii) Suppose there is a PPT algorithm $\mathcal{A}$, that generates with non-negligible probability two collection of shares yielding $(\overline{\alpha}, \overline{\beta})$ and $(\overline{\alpha}', \overline{\beta}')$, respectively, so that $\overline{\alpha} \neq \overline{\alpha}'$. Then we can use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks the binding property of Pedersen commitments as follows: First we invoke $\mathcal{A}$ to obtain the shares, then run the reconstruction algorithm, and finally output $(\overline{\alpha}, \overline{\beta}), (\overline{\alpha}', \overline{\beta}')$. If $\overline{\alpha} \neq \overline{\alpha}'$, then both pairs are valid, but different openings of the Pedersen commitment $\mathsf{cm} := C_0^{\Delta^2} = g_q^{\overline{\beta}} \cdot h_{\mathsf{P}}^{\overline{\alpha}} = g_q^{\overline{\beta}'} \cdot h_{\mathsf{P}}^{\overline{\alpha}'}$. Hence, by Theorem 3, such an $\mathcal{A}$ cannot exist.

Similarly, if $\overline{\alpha} = \overline{\alpha}'$ but $\overline{\beta} \neq \overline{\beta}'$, then $\overline{\beta} - \overline{\beta}'$ must be a non-zero multiple of $\mathrm{ord}(g_q)$ since $g_q^{\overline{\beta} - \overline{\beta}'} = 1$. □

Lemma 5 (Privacy of Pedersen VSS). *The Pedersen VSS for CL (Definition 8) with $\ell \in \mathbb{N}$ and $\ell_b \geq \ell + \log_2(\bar{s}) + 2 \cdot \sigma + 1$ provides statistical privacy (Definition 8 from the full version of [BDO23]).*

Proof. Write $\ell_G = \log_2(\bar{s})$, and consider a Pedersen public key $h_{\mathsf{P}} = g_q^{\omega}$, where $\omega \in_R [0, 2^{\ell_G + \sigma})$. Let $\alpha, \alpha' \in [0, 2^{\ell})$ be arbitrary, and let $\mathcal{C} \subseteq [N]$ be an arbitrary set of $|\mathcal{C}| = t$ corrupted parties.

Suppose α is shares as $\mathsf{P\text{-}Share}(\alpha; \beta; \mathbf{r}, \mathbf{s})$, i.e., we have two polynomials $\mathsf{f}, \mathsf{g} \in \mathbb{Z}[X]_{\leq t}$ and group elements $C_0, \ldots, C_t \in G^q$ such that

$$\mathsf{f}(X) = \Delta \cdot \alpha + \sum_{k \in [t]} r_k \cdot X^k, \qquad \mathsf{g}(X) = \Delta \cdot \beta + \sum_{k \in [t]} s_k \cdot X^k,$$
$$C_0 = g_q^{\beta} \cdot h_{\mathsf{P}}^{\alpha}, \text{ and } C_k = g_q^{s_k} \cdot h_{\mathsf{P}}^{r_k} \text{ for } k \in [t].$$

First, we show that for α' there exists a matching value β': We set $\beta' = \beta + (\alpha - \alpha') \cdot \omega$ such that $\beta + \alpha \cdot \omega = \beta' + \alpha' \cdot \omega$ and therefore $C_0 = g_q^{\beta'} \cdot h_{\mathsf{P}}^{\alpha'}$. Note that $\beta' \in [-2^{\ell + \ell_G + \sigma}, 2^{\ell_b} + 2^{\ell + \ell_G + \sigma})$, but if $\beta \in [2^{\ell + \ell_G + \sigma}, 2^{\ell_b} - 2^{\ell + \ell_G + \sigma})$, then always $\beta' \in [0, 2^{\ell_b})$. By choice of ℓ_b, the probability of this is $1 - 2^{\ell + \ell_G + \sigma + 1 - \ell_b} \geq 1 - 2^{-\sigma}$.

Now we use privacy of Shamir's secret sharing to show that there exist corresponding sharing polynomials f', g' for α', β': Let $\mathsf{s}_{\mathcal{C}}(X) \in \mathbb{Z}[X]_{\leq t}$ be the sweeping polynomial corresponding to $\mathcal{C}$ (see Definition 8 in the full version of [BDO23]), and consider the following map $\varphi \colon \mathbb{Z}^2 \to (\mathbb{Z}[X]_{\leq t})^2$:

$$(\alpha^*, \beta^*) \mapsto \big(\mathsf{f}(X) + (\alpha^* - \alpha) \cdot \mathsf{s}_{\mathcal{C}}, \mathsf{g}(X) + (\beta^* - \beta) \cdot \mathsf{s}_{\mathcal{C}}\big).$$

By the privacy proof of Shamir's secret sharing over $\mathbb{Z}$ (proof of Theorem 16 in the full version of [BDO23]), we have that applying this map to any $\alpha' \in [0, 2^{\ell})$ and $\beta' \in [0, 2^{\ell_b})$ results in polynomials $\mathsf{f}'(X) = \Delta \cdot \alpha' + \sum_{k \in [t]} r'_k \cdot X^k$ and $\mathsf{g}'(X) = \Delta \cdot \beta' + \sum_{k \in [t]} s'_k \cdot X^k$ that lie in the range of Share^{ℓ} and Share^{ℓ_b} except with probability negligible in σ.

Finally, we verify that these sharing polynomials are consistent with the public values $C_1, \ldots, C_t$: Set $\mathsf{h}(X) := \mathsf{g}(X) + \mathsf{f}(X) \cdot \omega$, and note that we have $C_k = g_q^{\mathsf{h}_k}$. Similarly, let $\mathsf{h}'(X) := \mathsf{g}'(X) + \mathsf{f}'(X) \cdot \omega$. By choice of β', we have that $\mathsf{h}'(X) = \mathsf{h}(X)$:

$$\begin{aligned}
\mathsf{h}'(X) &= \mathsf{g}'(X) + \mathsf{f}'(X) \cdot \omega \\
&= \mathsf{g}(X) + (\beta' - \beta) \cdot \mathsf{s}_{\mathcal{C}}(X) + \big(\mathsf{f}(X) + (\alpha' - \alpha) \cdot \mathsf{s}_{\mathcal{C}}(X)\big) \cdot \omega \\
&= \mathsf{g}(X) + \mathsf{f}(X) \cdot \omega + \underbrace{\big((\beta' - \beta) + (\alpha' - \alpha) \cdot \omega\big)}_{=0} \cdot \mathsf{s}_{\mathcal{C}}(X) = \mathsf{h}(X).
\end{aligned}$$

[13] If we add a PoK here for C_0, we should get normal integer reconstruction of α as in [BDO23] for the Feldman VSS.

Hence, we have $C_k = g_q^{\mathsf{h}'_k}$ as well, so the public values are consistent with both sharings.

Overall, we have shown that the adversaries view of a Pedersen VSS of a value $\alpha \in [0, 2^\ell)$ can equally likely be explained with a VSS of any other value $\alpha' \in [0, 2^\ell)$ except with probability negligible in σ. Hence, we conclude that for any $\alpha, \alpha' \in [0, 2^\ell)$ the following distributions are statistically indistinguishable:

$$\left\{\mathsf{P\text{-}Share}_i(\alpha; \beta; \mathbf{r}, \mathbf{s})\right\}_{P_i \in \mathcal{C}} \approx_s \left\{\mathsf{P\text{-}Share}_i(\alpha'; \beta'; \mathbf{r}', \mathbf{s}')\right\}_{P_i \in \mathcal{C}}.$$

□

4 Zero-Knowledge Arguments for Unknown Order Groups

4.1 Using the Rough-Order Assumption

The work of [BDO23] claimed zero-knowledge proofs with *computational soundness* under the RO_C assumption. Instead, their proofs are *unconditionally sound* with soundness error $1/C$ in C-rough-order groups. Furthermore, it is unclear how computational soundness can be reduced to the RO_C assumption in the absence of efficiently verifiable winning conditions; a malicious prover $\mathcal{P}^*$ with non-negligible advantage in proving a false statement does not immediately imply a distinguisher for RO_C when this distinguisher does not have the ability to efficiently determine if a proof is honestly generated for a true statement or forged for a false statement. In the absence of an efficient method to detect forged proofs the reduction cannot proceed. While the original claim of computational soundness is not strictly correct (or rather, remains unproven) the zero-knowledge proofs therein can be used in a larger context as if the claim holds. The reason is as follows: when the zero-knowledge proofs are used within a higher level protocol it is often possible to determine if a proof was forged given the other messages sent by the prover $\mathcal{P}^*$, which solves the above problem.

Example 6. Consider the VSS presented in Definition 8 and suppose that $t+1$ parties are satisfied with the check in Equation (7). Lemma 4 tells us that if $C_0, \ldots, C_t \in \langle g_q \rangle$ then (weak) reconstruction is possible. Suppose the dealer has used zero-knowledge proofs to assert that the C_i are in $\langle g_q \rangle$. If the proofs verify but reconstruction fails then the dealer must have successfully forged the proof. If this happens with non-negligible probability then that dealer can be used to construct a distinguisher for RO_C by running the dealer on the $\mathsf{pp}_{\mathsf{cl}}$ as output by the RO_C challenger. If reconstruction succeeds or the proofs do not verify the distinguisher guesses randomly and otherwise guesses 0 (= normal).

This motivates the general theorem below for proving security under the RO_C assumption. Theorem 7 states that security proven when $\mathsf{pp}_{\mathsf{cl}}$ are sampled according to $\mathcal{D}_C^{\mathsf{rough}}$ implies the same when $\mathsf{pp}_{\mathsf{cl}}$ are sampled normally, provided security is defined with respect to an efficient game. We note that the oracle which samples $\mathsf{pp}_{\mathsf{cl}}$ is potentially inefficient, but the game is otherwise efficient. This captures our discussion above; efficiency requires that the adversary outputs enough information to verify the satisfaction of that game's winning conditions. In our case this theorem can be applied to generalize the proven unconditional soundness of zero-knowledge proofs in C-rough groups to the computational soundness of the same proofs under the RO_C assumption.

Theorem 7. *Consider a PPT adversary $\mathcal{A}$ against an efficient security game $\mathsf{Game}_{\mathcal{A}}^{\mathcal{O}}(\lambda)$ that generates CL public parameters via a single query to an oracle $\mathsf{pp}_{\mathsf{cl}} \leftarrow \mathcal{O}$ and let $\mathsf{Game}_{\mathcal{A}}^{\mathcal{O}}(\lambda) = 1$ denote that $\mathcal{A}$ wins. Define $\mathsf{Game}_{\mathcal{A}}^{\mathsf{normal}}(\lambda)$ and $\mathsf{Game}_{\mathcal{A}}^{\mathsf{rough}}(\lambda)$ to be $\mathsf{Game}_{\mathcal{A}}^{\mathcal{O}}(\lambda)$ when $\mathcal{O}$ samples $\mathsf{pp}_{\mathsf{cl}} \leftarrow \mathsf{CLGen}(1^\lambda, q; \mathcal{U}(\{0,1\}^\lambda))$ and $\mathsf{pp}_{\mathsf{cl}} \leftarrow \mathsf{CLGen}(1^\lambda, q; \mathcal{D}_C^{\mathsf{rough}}(\lambda, q))$, respectively. Under the RO_C assumption,*

$$\Pr[\mathsf{Game}_{\mathcal{A}}^{\mathsf{normal}}(\lambda) = 1] \leq \Pr[\mathsf{Game}_{\mathcal{A}}^{\mathsf{rough}}(\lambda) = 1] + \mathsf{negl}(\lambda).$$

In particular, if no adversary $\mathcal{A}$ wins $\mathsf{Game}_{\mathcal{A}}^{\mathsf{rough}}(\lambda)$ with non-negligible advantage then the same holds for $\mathsf{Game}_{\mathcal{A}}^{\mathsf{normal}}(\lambda)$.

Proof. Suppose towards contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\varepsilon(\lambda) := \Pr[\mathsf{Game}_{\mathcal{A}}^{\mathsf{normal}}(\lambda) = 1] - \Pr[\mathsf{Game}_{\mathcal{A}}^{\mathsf{rough}}(\lambda) = 1]$ is non-negligible. This adversary $\mathcal{A}$ can be used to construct a distinguisher $\mathcal{B}$ for the RO_C game as follows: Obtain $\rho \in \{0,1\}^\lambda$ from the RO_C challenger and invoke a security game

$\mathsf{Game}^{?}_{\mathcal{A}}(\lambda)$ with $\mathcal{A}$ with CL public parameters generated as $(q, \bar{s}, f, g_q, \widehat{G}, F; \rho) \leftarrow \mathsf{CLGen}(1^\lambda, q; \rho)$. Then, run the game. If $\mathcal{A}$ wins, i.e., $\mathsf{Game}^{?}_{\mathcal{A}}(\lambda) = 1$, then output 0, and otherwise output 1.

Note that if the RO_C challenger sampled $\rho \in_R \{0,1\}^\lambda$ then $\mathsf{mode} = \mathsf{normal}$ and the game is identically $\mathsf{Game}^{\mathsf{normal}}_{\mathcal{A}}(\lambda)$ but if the challenger had instead sampled $\rho \leftarrow \mathcal{D}^{\mathsf{rough}}_C(\lambda, q)$ then $\mathsf{mode} = \mathsf{rough}$ and the game is $\mathsf{Game}^{\mathsf{rough}}_{\mathcal{A}}(\lambda)$.

Therefore, we can write the advantage of $\mathcal{B}$ in the RO_C game as below, which is non-negligible.

$$\begin{aligned}
\mathsf{Adv}^{RO_C}_{\mathcal{B}}(\lambda) &= \Pr[\mathcal{B}(\rho_b) \to b \mid b \in_R \{0,1\}, \rho_0 \in_R \{0,1\}^\lambda, \rho_1 \leftarrow \mathcal{D}^{\mathsf{rough}}_C(\lambda, q)] - \frac{1}{2} \\
&= \frac{1}{2} \cdot \Pr[\mathcal{B}(\mathcal{U}(\{0,1\}^\lambda)) \to 0] + \frac{1}{2} \cdot \Pr[\mathcal{B}(\mathcal{D}^{\mathsf{rough}}_C(\lambda, q)) \to 1] - \frac{1}{2} \\
&= \frac{1}{2} \cdot \Pr[\mathsf{Game}^{\mathsf{normal}}_{\mathcal{A}}(\lambda) \to 1] + \frac{1}{2} \cdot \Pr[\mathsf{Game}^{\mathsf{rough}}_{\mathcal{A}}(\lambda) \to 0] - \frac{1}{2} \\
&= \frac{1}{2} \cdot \Big(\Pr[\mathsf{Game}^{\mathsf{normal}}_{\mathcal{A}}(\lambda) \to 1] - \Pr[\mathsf{Game}^{\mathsf{rough}}_{\mathcal{A}}(\lambda) \to 1]\Big) \\
&= \frac{1}{2} \cdot \varepsilon(\lambda),
\end{aligned}$$

Hence, such an adversary $\mathcal{A}$ cannot exist. □

4.2 Batched Zero-Knowledge Arguments

In this section we present a method to batch Σ-like protocols over unknown order groups with improved communication efficiency.

In the context of prime-order groups Gennaro et al. [Gen+04] give a method to batch Schnorr proofs of discrete logarithm to prove relations of the following form, where g is a group generator, x_i are elements of the prime field and $b \in \mathbb{N}$:

$$\{(g, h_1, \ldots, h_b); (x_1, \ldots, x_b) \mid g^{x_i} = h_i \text{ for } i \in [b]\}.$$

The prover sends $t := g^r$ for random r to the verifier, receives a random challenge k and responds with $u := r + \sum_{i=1}^{b} x_i \cdot k^i$. The verifier then checks that $g^u \stackrel{?}{=} t \cdot \prod_{i=1}^{b} h_i^{k^i}$. With this approach the prover sends a single group element t and a single field element u as opposed to b of each kind without batching. The idea essentially uses packed Shamir's secret sharing to (in this case) provide 1-privacy and $(b+1)$-reconstruction implying both zero-knowledge and $(b+1)$-special soundness.

Bartoli et al. [BC24] build proofs of knowledge for unknown-order groups from black-box secret sharing schemes, noting that "Shamir secret sharing 'over the integers' is not a threshold black box secret sharing scheme since it only allows to reconstruct a multiple of the secret instead of the secret itself." Indeed, the integer Lagrange interpolation (with integer coefficients) (Section 3.1) reconstructs a Δ-multiple of the secret. Nevertheless, we use this technique to build batched proofs that are *set-membership sound*–but not proofs of knowledge–under the RO_C assumption.

Similar to [BDO23] we consider proofs of general relations R which are the conjunction of n equality constraints ($Y_{k,i} = \prod_{j=1}^{m} X_{i,j}^{w_{k,j}}$ for $i \in [n]$) on m secret values ($w_{k,j}$ for $j \in [m]$) and batch b instances of R ($(Y_{k,i}, \mathbf{X}_i, \mathbf{w}_k)$ for $k \in [b]$):[14]

$$\mathsf{R}^b = \left\{ \begin{array}{r} (Y_{k,i}, \mathbf{X}_i)_{(k,i)\in[b]\times[n]}; \\ (\mathbf{w}_k)_{k\in[b]} \end{array} \middle| \begin{array}{c} \forall k \in [b] : (Y_{k,i}, \mathbf{X}_i, \mathbf{w}_k) \in \widehat{G} \times \widehat{G}^m \times \mathbb{Z}^m, \\ \bigwedge_{k=1}^{b} \bigwedge_{i=1}^{n} \left[Y_{k,i} = \prod_{j=1}^{m} X_{i,j}^{w_{k,j}} \right] \end{array} \right\} \tag{8}$$

Definition 9 (Batched Σ-like Arguments). *For $j \in [m]$ let $C, A_j, S_j \in \mathbb{N}$ and $V_j := S_j \cdot \sum_{k=1}^{b} C^k$. Consider $X = (Y_{k,i}, \mathbf{X}_i)_{(k,i)\in[b]\times[n]}$ and $\mathbf{w} := (\mathbf{w}_k)_{k\in[b]}$. Define $\mathsf{C\Sigma P}^b(\mathsf{R}^b)$ as the three-move protocol below.*

[14] This is essentially a conjunction of $b \cdot n$ terms, however, we allow that the same m secrets are used among the n constraints – important for example when we need to prove that two pairs of group elements have the same discrete logarithm – but the b instances use disjoint sets of secrets. Moreover, the group elements $X_{i,j}$ are the same in each of the b instances – we can for example prove b discrete logarithms with respect to the same base g.

– $\mathsf{Commit}(X, \mathbf{w})$: *Sample* $\mathsf{stt} := (r_1, \ldots, r_m) \in_R [A_1] \times \cdots \times [A_m]$. *For* $i \in [n]$ *compute* $t_i := \prod_{j=1}^{m} X_{i,j}^{r_j}$. *Output* $(\mathsf{stt}, \mathsf{com} := (t_i)_{i\in[n]})$.
– *A challenge is sampled* $\mathsf{chl} \in_R [C]$.
– $\mathsf{Respond}(X, \mathbf{w}, \mathsf{stt}, \mathsf{chl})$: *For* $j \in [m]$ *compute* $u_j := r_j + \sum_{k=1}^{b} w_{k,j} \cdot \mathsf{chl}^k$. *Output* $\mathsf{res} := (u_1, \ldots, u_m)$.
– $\mathsf{Verify}(X, \mathsf{com}, \mathsf{chl}, \mathsf{res})$: *If the following conditions are satisfied for* $k \in [b]$, $i \in [n]$, $j \in [m]$ *then output* $\top$ *and otherwise output* $\bot$.

$$Y_{k,i}, X_{i,j} \in \widehat{G} \quad \wedge \quad u_j \in [-V_j, V_j + A_j] \quad \wedge \quad t_i \cdot \prod_{k=1}^{b} Y_{k,i}^{\mathsf{chl}^k} = \prod_{j=1}^{m} X_{i,j}^{u_j}$$

– $\mathsf{Simulate}(X, \mathsf{chl})$: *For* $j \in [m]$, $i \in [n]$ *draw* $\hat{u}_j \in_R [-V_j, V_j + A_j]$ *and set* $t_i := \left(\prod_{k=1}^{b} Y_{k,i}^{-\mathsf{chl}^k}\right) \cdot \left(\prod_{j=1}^{m} X_{i,j}^{\hat{u}_j}\right)$. *Output* $\left(\mathsf{com} := (t_i)_{i\in[n]}, \mathsf{res} := (\hat{u}_j)_{j\in[m]}\right)$.

Theorem 8. *Let* R^b *be a batch-relation as described in Equation* (8).
(i) $\mathsf{C\Sigma P}^b(\mathsf{R}^b)$ *is sound for* R^b *with soundness error* b/C *in* C*-rough class groups.*[15]
(ii) $\mathsf{C\Sigma P}^b(\mathsf{R}^b)$ *is complete for* R^b *if* $w_{k,j} \in [-S_j, +S_j]$ *for all* $k \in [b]$ *and* $j \in [m]$.
(iii) $\mathsf{C\Sigma P}^b(\mathsf{R}^b)$ *is statistical special honest-verifier zero-knowledge if, for all* $k \in [b]$ *and* $j \in [m]$, V_j/A_j *is negligible in* σ *and* $w_{k,j} \in [-S_j, +S_j]$.

Proof. (i) Assume $\mathrm{ord}(\widehat{G})$ is C-rough. Towards contradiction suppose that there exists a malicious PPT prover $\mathcal{P}^*$ that makes a verifier $\mathcal{V}$ accept an invalid instance X (i.e., there is no $\mathbf{w}$ such that $(X; \mathsf{w}) \in \mathsf{R}^b$) with probability $> b/C$.
By averaging there must exist a first message com and challenges $\mathsf{chl}_{(\ell)}$ for which $\mathcal{P}^*$ can produce $\mathsf{res}_{(\ell)}$ such that $\mathcal{V}$ accepts all $(\mathsf{com}, \mathsf{chl}_{(\ell)}, \mathsf{res}_{(\ell)})_{\ell\in[0,b]}$. We denote $w_{0,j} := r_j$ for $j \in [m]$ and $Y_{0,i} := t_i$ for $i \in [n]$ and model responses as the evaluation of polynomials $u_j(X) = \sum_{k=0}^{b} w_{k,j} \cdot X^k$ on the challenges $u_j^{(\ell)} := u_j(\mathsf{chl}_{(\ell)})$.

At a high level, the proof is as follows: when the group order is C-rough then verifier acceptance implies the existence of a witness because (1) $C!$-multiples of the $u_j(X)$ coefficients–i.e., the witness–can be recovered via Lagrange interpolation over the integers and (2) $C!$ is invertible modulo the group order. This essentially defines a knowledge extractor, however extraction is inefficient as computing the inverse of $C!$ or performing any computation involving $C!$ without knowing the group order is infeasible for $C = \omega(\mathsf{poly}(\lambda))$. Thus we prove soundness–not knowledge soundness–below.

For $(\mathsf{chl}_{(0)}, \ldots, \mathsf{chl}_{(b)})$ let $V \in \mathbb{Z}^{(b+1)\times(b+1)}$ be the corresponding Vandermonde matrix and define $\overline{V} := C! \cdot V^{-1}$.

$$V = \begin{pmatrix} 1 & \mathsf{chl}_{(0)} & \mathsf{chl}_{(0)}^2 & \ldots & \mathsf{chl}_{(0)}^b \\ 1 & \mathsf{chl}_{(1)} & \mathsf{chl}_{(1)}^2 & \ldots & \mathsf{chl}_{(1)}^b \\ 1 & \mathsf{chl}_{(2)} & \mathsf{chl}_{(2)}^2 & \ldots & \mathsf{chl}_{(2)}^b \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \mathsf{chl}_{(b)} & \mathsf{chl}_{(b)}^2 & \ldots & \mathsf{chl}_{(b)}^b \end{pmatrix} \in \mathbb{Z}^{(b+1)\times(b+1)}. \tag{9}$$

Note that $\overline{V} \in \mathbb{Z}^{(b+1)\times(b+1)}$ [BDO23, Lemma 18 in the full version] and, for all $\alpha, \gamma \in [0, b]$, Equation (10) holds.

$$\sum_{\beta=0}^{b} (\overline{V})_{\alpha,\beta} \cdot \mathsf{chl}_\beta^\gamma = \sum_{\beta=0}^{b} (\overline{V})_{\alpha,\beta} \cdot V_{\beta,\gamma} = \begin{cases} C! & \text{if } \alpha = \gamma \\ 0 & \text{if } \alpha \neq \gamma \end{cases} \tag{10}$$

The transcripts are accepting and therefore Equation (11) holds.

$$\prod_{k=0}^{b} Y_{k,i}^{(\mathsf{chl}_\ell^k)} = \prod_{j=0}^{m} X_{i,j}^{u_j^{(\ell)}} = \qquad \text{for all } i \in [n] \text{ and } \ell \in [0, b]. \tag{11}$$

[15] Note that the soundness property does not require the existence of a witness $\mathbf{w}$ such that each w_j is within the range $[-S, +S]$. Moreover, while unconditional soundness is proved for C-rough class groups, the RO_C assumption allows to use these zero-knowledge proofs in higher-level protocols using normal class groups to obtain security against a computationally bounded adversary. See Section 4.1.

For $j \in [m]$ define $\left(w'_{0,j}, \ldots, w'_{b,j}\right)^\intercal := \overline{V} \cdot \left(u_j^{(0)}, \ldots, u_j^{(b)}\right)^\intercal$ and, equivalently,

$$w'_{k,j} = \sum_{\ell=0}^{b} \overline{V}_{k,\ell} \cdot u_j^{(\ell)} \qquad \text{for } j \in [m] \text{ and } k \in [0,b]. \tag{12}$$

Observe that then, for any $k \in [b]$ and $i \in [n]$:

$$\begin{aligned}
\prod_{j=0}^{m} (X_{i,j})^{w'_{k,j}} &\overset{(12)}{=} \prod_{j=0}^{m} (X_{i,j})^{\sum_{\ell=0}^{b}\left(\overline{V}_{k,\ell} \cdot u_j^{(\ell)}\right)} = \prod_{j=0}^{m}\prod_{\ell=0}^{b} \left(X_{i,j}^{u_j^{(\ell)}}\right)^{\overline{V}_{k,\ell}} \\
&= \prod_{\ell=0}^{b} \left(\prod_{j=0}^{m} X_{i,j}^{u_j^{(\ell)}}\right)^{\overline{V}_{k,\ell}} \overset{(11)}{=} \prod_{\ell=0}^{b} \left(\prod_{l=0}^{b} Y_{l,i}^{(\mathsf{chl}_\ell^l)}\right)^{\overline{V}_{k,\ell}} \\
&\overset{(9)}{=} \prod_{\ell=0}^{b}\prod_{l=0}^{b} Y_{l,i}^{(\overline{V}_{k,\ell} \cdot V_{\ell,l})} = \prod_{l=0}^{b} Y_{l,i}^{\sum_{\ell=0}^{b} \overline{V}_{k,\ell} \cdot V_{\ell,l}} \overset{(10)}{=} (Y_{k,i})^{C!}.
\end{aligned}$$

By assumption $\mathrm{ord}(\widehat{G})$ is C-rough so $C!$ is invertible modulo $\mathrm{ord}(\widehat{G})$. For $k \in [b]$ and $i \in [n]$ let $w_{k,j} := w'_{k,j} \cdot (C!)^{-1} \bmod \mathrm{ord}(\widehat{G})$. Then, by the above, we have a valid witness such that $(X; \mathbf{w}) \in \mathsf{R}^b$ in contradiction to our assumption.

$$Y_{k,i} = \prod_{j=0}^{m} X_{i,j}^{w_{k,j}} \qquad \text{for } k \in [b] \text{ and } i \in [n].$$

Thus, such a $\mathcal{P}^*$ cannot exist.

(ii) An honest prover on input $\mathbf{w}_k \in [-S_1, +S_1] \times \ldots, [-S_m, +S_m]$ outputs responses $u_j \in [-V_j, +V_j + A_j]$. Moreover, if $\mathbf{w}_k$ are valid witnesses and t_i are computed correctly then the verifier accepts.

$$\begin{aligned}
t_i \cdot \prod_{k=1}^{b} Y_{k,i}^{\mathsf{chl}^k} = \prod_{j=1}^{m} X_{i,j}^{u_j} &\iff \prod_{j=1}^{m} X_{i,j}^{r_j} \cdot \prod_{k=1}^{b} Y_{k,i}^{\mathsf{chl}^k} = \prod_{j=1}^{m} X_{i,j}^{r_j + \sum_{k=1}^{b} w_{k,j} \cdot \mathsf{chl}^k} \\
&\iff \prod_{k=1}^{b} (Y_{k,i})^{\mathsf{chl}^k} = \prod_{k=1}^{b} \left(\prod_{j=1}^{m} X_{i,j}^{w_{k,j}}\right)^{\mathsf{chl}^k}
\end{aligned}$$

(iii) Let $\mathbf{w}$ be a witness such that $w_{k,j} \in [-S_j, +S_j]$ for $(k,j) \in [b] \times [m]$, let $\mathsf{chl} \in [C]$ and define $v_j := \sum_{k=1}^{b} w_{k,j} \cdot \mathsf{chl}^k \in [-V_j, +V_j]$.

In the real protocol $r_j \in_R [A_j]$ is sampled and $u_j = r_j + v_j \in_R [v_j, v_j + A]$ is then computed.

$$\Pr_{r_j}[u_j = a \mid \mathsf{chl}] = \begin{cases} 0 & \text{if } -V_j < a < v_j \\ 1/A_j & \text{if } v_j < a < A_j + v_j \\ 0 & \text{if } A_j + v_j < a < A_j + V_j \end{cases}$$

In the simulation $\hat{u}_j \in_R [-V_j, A_j + V_j]$ is sampled uniformly and the unique t_i are selected such that the verifier accepts.

$$\Pr_{\hat{u}_j}[\hat{u}_j = a \mid \mathsf{chl}] = 1/(A_j + 2V_j) \text{ for } -V_j < a < A_j + V_j$$

In both cases the pair (u_j, chl) uniquely determine t_i. We compute the statistical distance between the response distributions:

$$\begin{aligned}
\Delta(u_j, \hat{u}_j) &= \max_I \left| \Pr[u_j \in I \mid \mathsf{chl}] - \Pr[\hat{u}_k \in I \mid \mathsf{chl}] \right| \\
&= 2 \cdot V_j \cdot \left(\frac{1}{A_j + 2 \cdot V_j} - 0\right) = \frac{2 \cdot V_j}{A_j + 2 \cdot V_j} < \frac{2 \cdot V_j}{A_j}.
\end{aligned} \tag{13}$$

As V_j/A_j is negligible in σ the responses are distributed statistically close.

□

Parameters and Communication Cost

Lemma 9. *Let $b, m, n \geq 1$ be as above. To achieve λ bit of computational and σ bit of statistical security, we can set $C := 2^{\lambda+\lceil\log_2(b)\rceil}$ and $A_j := 2^{\sigma+1} \cdot S_j \cdot \sum_{k=1}^{b} C^k \leq 2^{b\cdot\lambda+\sigma+\lceil\log_2(S_j)\rceil+b\cdot\lceil\log_2(b)\rceil+2}$ for $j \in [m]$. This results in a proof size of at most $\lambda + \lceil\log_2(b)\rceil + m \cdot (b \cdot \lambda + \sigma + \lceil\log_2(S)\rceil + b \cdot \lceil\log_2(b)\rceil + 2)$ bit with compressed Fiat-Shamir and $S := \max(S_1, \ldots, S_b)$.*

Proof.
- Soundness: By Theorem 8 (i) we have soundness error $b/C = b \cdot 2^{-\lambda-\lceil\log_2(b)\rceil} \leq 2^{-\lambda}$.
- Zero-Knowledge: Plugging A_j into Equation (13) yields a negligible $\Delta(u_j, \hat{u}_j) < 2^{-\sigma}$.
- Proof size: For simplicity, we assume that $S_1 = \cdots = S_m := S$ and therefore $A_1 = \cdots = A_m$. When using the compressed Fiat-Shamir transformation, the prover only needs to send chl and res:

$$\begin{aligned}|\mathsf{chl}| + |\mathsf{res}| &\leq (\lambda + \lceil\log_2(b)\rceil) + m \cdot \log_2(A) + 1 \\ &\leq (\lambda + \lceil\log_2(b)\rceil) + m \cdot (b \cdot (\lambda + \lceil\log_2(b)\rceil) + \sigma + \lceil\log_2(S)\rceil + 2).\end{aligned}$$

□

In comparison a proof of b instances without batching using the protocol from [BDO23] results has size:

$$|\mathsf{chl}'| + b \cdot |\mathsf{res}'| \leq \lambda + b \cdot m \cdot (\lambda + \sigma + \log_2(S) + 2).$$

In the batched variant the batch size b is multiplied by the challenge size $|\mathsf{chl}| = \lambda + \lceil\log_2(b)\rceil$, whereas in the non-batched variant the batch size is additionally multiplied by the statistical security parameter σ and the secret size $\lceil\log_2(S)\rceil$.

5 The Linearly Homomorphic Encryption Scheme

Let $\Pi_{\mathsf{hsm\text{-}cl}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be the original HSM-CL encryption scheme from [CLT18]. Following [BDO23], we specify a variant where an adversary $\mathcal{A}$ is allowed to bias the key in a limited way, and additionally multiply the original secret key by a constant factor a such that $\gcd(a, q) = 1$. We denote our variant as $\Pi^{*,a}_{\mathsf{hsm\text{-}cl}} := (\mathsf{Setup}, \mathsf{BiasedKeyGen}^{\mathcal{A}}_a, \mathsf{Enc}, \mathsf{Dec})$ and specify $\mathsf{BiasedKeyGen}^{\mathcal{A}}_a$ in Figure 3.By slightly tweaking the proof of [BDO23] we obtain Theorem 10.

$\underline{\mathsf{BiasedKeyGen}^{\mathcal{A}}_a(\mathsf{pp}_{\mathsf{cl}})}$

1. Sample $\alpha \leftarrow \mathcal{D}_q$.
2. Set $\mathsf{pk}^*_{\mathsf{cl}} := g_q^{\alpha}$.
3. $\delta \leftarrow \mathcal{A}(\mathsf{pp}_{\mathsf{cl}}, \mathsf{pk}^*_{\mathsf{cl}})$.
4. $\mathsf{sk}_{\mathsf{cl}} := a \cdot \alpha + \delta$ and $\mathsf{pk}_{\mathsf{cl}} := g_q^{\mathsf{sk}_{\mathsf{cl}}} = g_q^{a\cdot\alpha+\delta}$.
5. Output $(\mathsf{pk}_{\mathsf{cl}}, \mathsf{sk}_{\mathsf{cl}})$.

Fig. 3. Modified key generation algorithm for a constant $a \in \mathbb{Z}$ that allow an adversary $\mathcal{A}$ to influence the distribution of public keys in a limited way.

Theorem 10 (IND-CPA Security of $\Pi^{*,a}_{\mathsf{hsm\text{-}cl}}$). *If $\Pi_{\mathsf{hsm\text{-}cl}}$ is indistinguishable under chosen-plaintext attacks (IND-CPA), so is $\Pi^{*,a}_{\mathsf{hsm\text{-}cl}}$ for any a coprime to q.*

Proof. We prove the claim by showing that any adversary $\mathcal{B}$ that wins the IND-CPA game for $\Pi^{*,a}_{\mathsf{hsm\text{-}cl}}$ with advantage $\mathsf{Adv}^{\mathsf{hsm\text{-}cl},*,a}_{\mathcal{B}}$ can be transformed into an adversary $\mathcal{A}$ for $\Pi_{\mathsf{hsm\text{-}cl}}$ with the same advantage $\mathsf{Adv}^{\mathsf{hsm\text{-}cl}}_{\mathcal{A}} = \mathsf{Adv}^{\mathsf{hsm\text{-}cl},*,a}_{\mathcal{B}}$. Given an adversary $\mathcal{B}$, we create $\mathcal{A}$ as follows: Initially $\mathcal{A}$ receives public parameters $\mathsf{pp}_{\mathsf{cl}}$ and a public key $\mathsf{pk}_{\mathsf{cl}}$ from the challenger. It gives $\mathsf{pk}_{\mathsf{cl}}$ to $\mathcal{B}$, which responds with δ. Define the biased public key $\mathsf{pk}'_{\mathsf{cl}} := \mathsf{pk}^a_{\mathsf{cl}} \cdot g_q^{\delta}$. This phase corresponds to the $\mathsf{BiasedKeyGen}^{\mathcal{B}}_a(\mathsf{pp}_{\mathsf{cl}})$ procedure. Then $\mathcal{B}$ selects $m_0, m_1 \in \mathbb{F}_q$, which $\mathcal{A}$ forwards to the challenger as $m'_b := m_b \cdot a^{-1} \bmod q$ for $b \in \{0, 1\}$. The challenger samples $b \in_R \{0, 1\}$, encrypts m'_b and sends the resulting $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ to $\mathcal{A}$. $\mathcal{A}$ sends $\mathsf{ct}' = (\mathsf{ct}'_1, \mathsf{ct}'_2) = (\mathsf{ct}_1, \mathsf{ct}^a_2 \cdot \mathsf{ct}^{\delta}_1)$ to $\mathcal{B}$. Finally, $\mathcal{B}$ outputs a bit $b' \in \{0, 1\}$, and $\mathcal{A}$ forwards this output to the challenger.

The challenger creates the ciphertext $\mathsf{ct} = (g_q^r, \mathsf{pk}_{\mathsf{cl}}^r \cdot f^{m'_b})$, where r denotes the randomness used during the encryption. We have $\mathsf{pk}'_{\mathsf{cl}} = \mathsf{pk}_{\mathsf{cl}}^a \cdot g_q^\delta$ and

$$\mathsf{ct}' = \left(g_q^r, (\mathsf{pk}_{\mathsf{cl}}^r \cdot f^{m'_b})^a \cdot (g_q^r)^\delta\right) = \left(g_q^r, (\mathsf{pk}_{\mathsf{cl}}^a \cdot g_q^\delta)^r \cdot f^{m'_b \cdot a}\right) = \left(g_q^r, (\mathsf{pk}'_{\mathsf{cl}})^r \cdot f^{m_b}\right).$$

Therefore, ct' is a valid encryption of m_b under the biased public key $\mathsf{pk}'_{\mathsf{cl}}$ with the same distribution as it would normally have. Overall, $\mathcal{A}$ wins the game iff $\mathcal{B}$ answers correctly, which happens with probability $1/2 + \mathsf{Adv}_{\mathcal{B}}^{\mathsf{hsm\text{-}cl},*,a}$. Hence, the advantage in the case of biased keys is the same as in the standard scheme. □

Figure 4: Ideal Threshold Encryption Functionality $\mathcal{F}_{\mathsf{TE}}$

Init On input $(\mathsf{Init}, 1^\lambda, q)$ from all parties, $\mathcal{F}_{\mathsf{TE}}$ generates $\mathsf{pp}_{\mathsf{cl}} \leftarrow \mathsf{CLGen}(1^\lambda, q)$. It stores $\mathsf{pp}_{\mathsf{cl}}$ and outputs them to all parties. This method must be called exactly once, and before any other call.

KeyGen On input (KeyGen) from all parties, $\mathcal{F}_{\mathsf{TE}}$ runs $(\mathsf{pk}_{\mathsf{cl}}, \mathsf{sk}_{\mathsf{cl}}) \leftarrow \mathsf{BiasedKeyGen}_{\Delta^2}^{\mathcal{S}}(\mathsf{pp}_{\mathsf{cl}})$ with $\mathcal{S}$. When $\mathcal{S}$ responds with $\mathsf{continue}$, $\mathcal{F}_{\mathsf{TE}}$ sends $\mathsf{pk}_{\mathsf{cl}}$ to all parties. This method must be called exactly once, and before any call to $\mathsf{Decrypt}$.

Decrypt On input $(\mathsf{Decrypt}, \mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2) \in \widehat{G}^2)$ from all parties, $\mathcal{F}_{\mathsf{TE}}$ computes $M := \mathsf{ct}_2 \cdot \mathsf{ct}_1^{-\mathsf{sk}_{\mathsf{cl}}}$ and sends (ct, M) to $\mathcal{S}$.[a] If $\mathcal{S}$ responds with abort, $\mathcal{F}_{\mathsf{TE}}$ aborts. Otherwise, if it responds with $\mathsf{continue}$, $\mathcal{F}_{\mathsf{TE}}$ sends $m \leftarrow \mathsf{CLSolve}(\mathsf{pp}_{\mathsf{cl}}, M)$ to all parties.

[a] We leak M instead of m to also be able to simulate the decryption in case ct is not a valid ciphertext since if $M \notin F \implies \mathsf{CLSolve}(\mathsf{pp}_{\mathsf{cl}}, M) = \perp$.

6 Protocol with Static Security

In this section we present a statically secure threshold encryption protocol $\Pi_{\mathsf{TE}}^{\mathsf{static}}$ in Figure 5 realizing the ideal functionality $\mathcal{F}_{\mathsf{TE}}$ (Figure 4, adapted from [BDO23]).

6.1 The Static Protocol

The protocol $\Pi_{\mathsf{TE}}^{\mathsf{static}}$ uses $\mathcal{F}_{\mathsf{CL}}$ (Figure 1) to generate CL public parameters $\mathsf{pp}_{\mathsf{cl}}$. In the optimistic case – when no corrupted party interferes – distributed key generation proceeds in a single round. Each party P_i samples their contribution α_i to the secret key and shares it with Feldman VSS (Definition 4). The shares are verified and, if necessary, parties engage in a complaint phase to determine the set of qualified parties $\mathcal{Q}$ that distributed valid shares.

At this point the honest parties hold consistent shares of α_j for each $P_j \in \mathcal{Q}$ and can compute Shamir-style shares γ_i of $\sum_{P_j \in \mathcal{Q}} \alpha_j$. The secret key $\mathsf{sk}_{\mathsf{cl}}$, which is implicitly defined as $\mathsf{sk}_{\mathsf{cl}} := \Delta^2 \cdot \sum_{P_j \in \mathcal{Q}} \alpha_j$, can be reconstructed using integer Lagrange interpolation. Moreover, from the public Feldman values, every party can compute the public key $\mathsf{pk}_{\mathsf{cl}} = \prod_{P_j \in \mathcal{Q}} C_{j,0}^{\Delta^2}$. Finally, for each $P_j \in \mathcal{Q}$ a commitment $\Gamma_j = g_q^{\Delta^2 \cdot \gamma_j}$ can be computed and used to verify partial decryptions.

The DKG protocol is adapted from [BDO23] with the following differences:

- Because [BDO23] use their threshold cryptosystem to construct a multiparty computation protocol their DKG includes a second phase to generate a *special* public key which additionally includes an encryption of 1. We omit this second phase.[16]
- To eliminate the proofs of knowledge required in the protocol of [BDO23], we use the weak reconstruction property of Feldman VSS (Lemma 1). This results in the generated secret key being scaled by a factor Δ^2, which does *not* affect the security of the encryption scheme (Theorem 10).

[16] It is reasonable to expect that, due to the CPA security of the underlying scheme, the generation of special public keys which enable an indistinguishable switch to *lossy mode* and thereby facilitate reactive secure computation is possible as in [BDO23].

Figure 5: Protocol $\Pi^{\mathsf{static}}_{\mathsf{TE}}$

All parties maintain a set $\mathcal{Q}$, initialized to the entire set of parties, containing the qualified parties. A party is ignored once disqualified and removed from $\mathcal{Q}$.

Init. Send $(\mathsf{Gen}, 1^\lambda, q)$ to $\mathcal{F}_{\mathsf{CL}}$ and receive $\mathsf{pp}_{\mathsf{cl}} = (q, \bar{s}, f, g_q, \widehat{G}, F; \rho)$.

KeyGen. All P_i proceed in parallel:

1. Generation of the key pair $(\mathsf{pk}_{\mathsf{cl}}, \mathsf{sk}_{\mathsf{cl}})$.
   (a) Sample a contribution to the secret key $\alpha_i \leftarrow \mathcal{D}_q$.
   (b) Share α_i as $(y_{i,1}, \ldots, y_{i,N}; C_{i,0}, \ldots, C_{i,t}) \leftarrow \mathsf{F\text{-}Share}(\alpha_i; \mathbf{r}_i; g_q)$.
   (c) Compute $\pi_i \leftarrow \Pi^{\mathsf{ZK}}_{\mathsf{DLog}}.\mathsf{Prove}((g_q, (C_{i,0}, C_{i,1}^{\Delta}, \ldots, C_{i,t}^{\Delta})), (\alpha_i, r_{i,1}, \ldots, r_{i,t}))$.
   (d) Broadcast $(C_{i,k})_{k\in[0,t]}$ and π_i. Privately send $y_{i,j}$ to P_j.
2. Determining the set of qualified parties $\mathcal{Q}$.
   (a) Verify the shares received from $P_j \neq P_i$:
      i. $\mathsf{F\text{-}Check}(i, y_{j,i}; C_{j,0}, \ldots, C_{j,t}; g_q) \stackrel{?}{=} \top$, and
      ii. $\Pi^{\mathsf{ZK}}_{\mathsf{DLog}}.\mathsf{Verify}((g_q, (C_{j,0}, C_{j,1}^{\Delta}, \ldots, C_{j,t}^{\Delta})), \pi_j) \stackrel{?}{=} \top$.

   If Step 2(a)i failed, but Step 2(a)ii succeeded, then broadcast a complaint against P_j.
   (b) For every complaint received by $P_j \neq P_i$: broadcast $y_{i,j}$.
   (c) If, for P_j, Step 2(a)ii failed or a response $y_{j,l}$ to P_l does not satisfy Step 2(a)i: remove P_j from $\mathcal{Q}$.
3. Computing the public key $\mathsf{pk}_{\mathsf{cl}}$ and secret key share γ_i.
   (a) Compute public key $\mathsf{pk}_{\mathsf{cl}} := \prod_{P_j\in\mathcal{Q}} C_{j,0}^{\Delta^2}$.
   (b) Compute Shamir-share[a] $\gamma_i := \sum_{P_j\in\mathcal{Q}} y_{j,i}$ of $\mathsf{sk}_{\mathsf{cl}} = \Delta^2 \cdot \sum_{P_j\in\mathcal{Q}} \alpha_j$.
   (c) For each $P_j \in \mathcal{Q}$ compute[b] $\Gamma_j := \prod_{P_l\in\mathcal{Q}} \left(C_{l,0}^{\Delta^2} \cdot \prod_{k=1}^{t} C_{l,k}^{(j^k)} \right)^{\Delta}$.

Decrypt To jointly decrypt a $\Pi_{\mathsf{hsm\text{-}cl}}$ ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2) \in \widehat{G}^2$, all P_i proceed in parallel as follows:

1. Compute $w_i := \mathsf{ct}_1^{\gamma_i \cdot \Delta^2}$ and $\pi_i \leftarrow \Pi^{\mathsf{ZK}}_{\mathsf{EqDLog}}.\mathsf{Prove}((g_q^{\Delta^2}, \mathsf{ct}_1^{\Delta^2}, \Gamma_i, w_i), \gamma_i)$. Broadcast (w_i, π_i).
2. Define $S := \{P_j \mid \Pi^{\mathsf{ZK}}_{\mathsf{EqDLog}}.\mathsf{Verify}((g_q^{\Delta^2}, \mathsf{ct}_1^{\Delta^2}, \Gamma_j, w_j), \pi_j) = \top\}$.
3. Compute $W := \prod_{P_j\in S} w_j^{(L_j^S(0))}$ and $\overline{M} := \mathsf{ct}_2^{\Delta^2} \cdot W^{-1}$.
4. Output $m := \mathsf{CLSolve}(\mathsf{pp}_{\mathsf{cl}}, \overline{M}) \cdot \Delta^{-2} \bmod q$. If $\mathsf{CLSolve}(\mathsf{pp}_{\mathsf{cl}}, \overline{M}) \to \perp$ then output $\perp$.

[a] The private values γ_i are $\mathbb{Z}$-shares of $\mathsf{sk}_{\mathsf{cl}}$ as defined; efficient recovery of the secret key is done via interpolation with *integer* Lagrange coefficients $L = \Delta\ell$.

[b] The public values Γ_j have discrete logarithm $\Delta^2\gamma_j$. They can be more efficiently computed as $\Gamma_j := \left(\mathsf{pk}_{\mathsf{cl}} \cdot \prod_{k=1}^{t} \left(\prod_{P_l\in\mathcal{Q}} C_{l,k}\right)^{(j^k)}\right)^{\Delta}$ and applying Horner's rule in the exponent.

Threshold decryption is non-interactive – a single round is sufficient even if corrupted parties try to interfere. Each party P_i broadcasts their partial decryption $w_i = \mathsf{ct}_1^{\gamma_i \cdot \Delta^2}$ together with a proof of equal discrete logarithm with respect to the public commitment Γ_i. Given $t+1$ valid partial decryptions, the parties can use integer Lagrange interpolation in the exponent to obtain a group element $W = \mathsf{ct}_1^{\Delta^2 \cdot \mathsf{sk}_{\mathsf{cl}}} \in F$. Then the plaintext is computed as $\overline{M} := \mathsf{ct}_2^{\Delta^2} \cdot W^{-1}$ followed by computing the discrete logarithm and removing of the additional factors Δ^2 over $\mathbb{F}_q$: $m = \log_f(\overline{M}) \cdot \Delta^{-2} \bmod q$.

6.2 Static Security

Theorem 11 (UC Security of $\Pi^{\mathsf{static}}_{\mathsf{TE}}$ in Figure 5). *The protocol $\Pi^{\mathsf{static}}_{\mathsf{TE}}$ UC-realizes the ideal functionality $\mathcal{F}_{\mathsf{TE}}$ with computational and active security and guaranteed output delivery, tolerating up to $t < N/2$ static corruptions and providing guaranteed output delivery in the $\mathcal{F}_{\mathsf{CL}}$-hybrid model given secure channels and broadcast under the ORD and RO_C assumptions.*

Correctness of $\Pi^{\mathsf{static}}_{\mathsf{TE}}$

Qualified Parties. As disqualification is exclusively dependent upon broadcasted information all (honest) parties agree on the set of qualified parties $\mathcal{Q}$ by the end of Step 2c. Furthermore, as every honest party $P_i \in \mathcal{H}$ executes the protocol honestly, no honest P_i is disqualified.

Key Generation. Each honest party $P_i \in \mathcal{H}$ samples and shares α_i with F-Share, proving that $C_{i,k} \in \langle g_q \rangle$ for $k \in [0,t]$ with $\overline{\Pi}^{\mathsf{ZK}}_{\mathsf{DLog}}$. Step 2a verifies all received shares and proofs. By the end of Step 2c, each honest $P_j \in \mathcal{H}$ has received a verifying share $y_{i,j}$ from each qualified $P_i \in \mathcal{Q}$. By Lemma 1, under the assumption that Δ is co-prime to the order of g_q, the honest parties $\mathcal{H}$ collectively hold at least $|\mathcal{H}| \geq t+1$ consistent Feldman-shares $(y_{i,j})_{j \in \mathcal{H}}$ of the α_i shared by each qualified $P_i \in \mathcal{Q}$. Moreover, the weak reconstruction property allows reconstruction of $\overline{\alpha}_j$ from these shares such that $C_{j,0}^{\Delta^2} = g_q^{\overline{\alpha}_j}$.

In Step 3 each $P_i \in \mathcal{Q}$ computes $\mathsf{pk}_{\mathsf{cl}} := \prod_{P_j \in \mathcal{Q}} C_{j,0}^{\Delta^2}$ which implicitly defines the secret key $\mathsf{sk}_{\mathsf{cl}} := \Delta^2 \cdot \sum_{P_i \in \mathcal{Q}} \alpha_i$. Each P_i also computes their share $\gamma_i := \sum_{P_j \in \mathcal{Q}} y_{j,i}$ of the secret key.

To verify correctness of these shares, we observe the following: By Lemma 1, each $S \subseteq \mathcal{H}$ of size $|S| \geq t+1$ could use their shares to reconstruct a value $\overline{\alpha}_i = \sum_{P_j \in S} L_j^S(0) \cdot y_{i,j}$ such that $g_q^{\overline{\alpha}_i} = C_{i,0}^{\Delta^2}$. This results in

$$\mathsf{pk}_{\mathsf{cl}} = \prod_{P_i \in \mathcal{Q}} C_{i,0}^{\Delta^2} = \prod_{P_i \in \mathcal{Q}} g_q^{\sum_{P_j \in S} L_j^S(0) \cdot y_{i,j}} = g_q^{\sum_{P_j \in S} L_j^S(0) \cdot \sum_{P_i \in \mathcal{Q}} y_{i,j}} = g_q^{\sum_{P_j \in S} L_j^S(0) \cdot \gamma_j} = g_q^{\mathsf{sk}_{\mathsf{cl}}}. \tag{14}$$

Hence, we have $\sum_{P_j \in S} L_j^S(0) \cdot \gamma_j = \mathsf{sk}_{\mathsf{cl}} \pmod{\mathrm{ord}(g_q)}$.[17] The parties also compute public verification values Γ_j for each party P_j:

$$\Gamma_j := \prod_{P_l \in \mathcal{Q}} \left(C_{l,0}^{\Delta^2} \cdot \prod_{k=1}^{t} C_{l,k}^{(j^k)} \right)^{\Delta} \overset{(\star)}{=} \prod_{P_l \in \mathcal{Q}} g_q^{\Delta^2 \cdot y_{l,j}} = g_q^{\Delta^2 \cdot \sum_{P_l \in \mathcal{Q}} y_{l,j}} = g_q^{\Delta^2 \cdot \gamma_j}, \tag{15}$$

where $(\star)$ holds if the Feldman check Equation (6) is satisfied for the share that P_j has received from P_l (in particular for all $P_j \in \mathcal{H}$, since $P_l \in \mathcal{Q}$).

Distributed Decryption. For the ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2) = (g_q^r, \mathsf{pk}_{\mathsf{cl}}^r \cdot f^m)$ every P_i broadcasts $w_i = \mathsf{ct}_1^{\Delta^2 \cdot \gamma_i}$ and a proof of correctness π_i according to $\Pi^{\mathsf{ZK}}_{\mathsf{EqDLog}}$. This ensures that the correct share γ_i was used to compute the partial decryption by checking that $\log_{\mathsf{ct}_1}(w_i) = \log_{g_q}(\Gamma_i)$ (which is $\Delta^2 \cdot \gamma_i$ by Equation (15)).

The encrypted message can be recovered from any set of $t+1$ correct partial decryptions, where the set $S \subseteq \mathcal{Q}$ contains the parties that broadcast verifying proofs and thus correct w_i:

[17] The modulo stems from the fact that $\mathsf{sk}_{\mathsf{cl}}$ is defined as the discrete logarithm of $\mathsf{pk}_{\mathsf{cl}}$ to the basis g_q. In practice, there will only ever be a single representative of $\mathsf{sk}_{\mathsf{cl}}$ due to the ORD assumption.

$$W = \prod_{P_j \in S} w_j^{L_j^S(0)} = \mathsf{ct}_1^{\Delta^2 \cdot \sum_{P_j \in S} L_j^S(0) \cdot \gamma_j} \overset{(14)}{=} \mathsf{ct}_1^{\Delta^2 \cdot \mathsf{sk}_{\mathsf{cl}}} = \mathsf{pk}_{\mathsf{cl}}^{\Delta^2 \cdot r} \tag{16}$$

$$\overline{M} = \mathsf{ct}_2^{\Delta^2} \cdot W^{-1} = (\mathsf{pk}_{\mathsf{cl}}^r \cdot f^m)^{\Delta^2} \cdot \left(\mathsf{pk}_{\mathsf{cl}}^{\Delta^2 \cdot r}\right)^{-1} = f^{\Delta^2 \cdot m} \tag{17}$$

$$m = \mathsf{CLSolve}(\mathsf{pp}_{\mathsf{cl}}, \overline{M}) \cdot \Delta^{-2} \bmod q \tag{18}$$

Equation (16) demonstrates how parties compute the public key raised to $r \cdot \Delta^2$. This value W is removed from $\mathsf{ct}_2^{\Delta^2}$ to recover $m \cdot \Delta^2$ in the exponent of f as in Equation (17). Finally, we take the discrete logarithm and divide by Δ_2 in $\mathbb{F}_q$ to obtain the message m. In the case that CLSolve fails the honest parties output $\perp$.

Figure 6: Simulator for $\Pi_{\mathsf{TE}}^{\mathsf{static}}$

Let $\mathcal{C}$ and $\mathcal{H}$ denote the sets of corrupt and honest parties, respectively, and let $P_h \in \mathcal{H}$ be a fixed honest party.

Init. Upon receipt of $(\mathsf{Init}, 1^\lambda, q)$ from $P_i \in \mathcal{C}$ forward the query to $\mathcal{F}_{\mathsf{TE}}$. Output the received $\mathsf{pp}_{\mathsf{cl}}$.

KeyGen. Simulate the generation of $\mathsf{pk}_{\mathsf{cl}}$ by executing $\mathsf{BiasedKeyGen}_{\Delta^2}^{\mathcal{S}}$ with $\mathcal{F}_{\mathsf{TE}}$.

1. Query $\mathcal{F}_{\mathsf{TE}}$ with (KeyGen) on behalf of every $P_i \in \mathcal{C}$ and receive the intermediate public key $\mathsf{pk}_{\mathsf{cl}}^*$.
2. For all $P_i \in \mathcal{H} \setminus \{P_h\}$ simulate Steps 1a to 1c according to $\Pi_{\mathsf{TE}}^{\mathsf{static}}$.
3. For P_h:
   (a) Sample $\alpha_h \leftarrow \mathcal{D}_q$ and share via $(y_{h,1}, \ldots, y_{h,N}) \leftarrow \mathsf{Share}(\alpha_h; \mathbf{r}_h)$.
   (b) Define $C_{h,0} := \mathsf{pk}_{\mathsf{cl}}^* \cdot \prod_{P_i \in \mathcal{H} \setminus \{P_h\}} \left(C_{i,0}\right)^{(-1)}$.
   (c) Use Lemma 2 to simulate $C_{h,1}, \ldots, C_{h,t}$ given $C_{h,0}$ and $(y_{h,i})_{P_i \in \mathcal{C}}$.
   (d) Simulate $\pi_h \leftarrow \overline{\Pi}_{\mathsf{DLog}}^{\mathsf{ZK}}.\mathsf{SimProve}((g_q, (C_{h,k})_{k \in [0,t]}))$.
4. Upon receipt of the first round messages from all $P_i \in \mathcal{C}$ simulate verification and complaints honestly according to Step 2a of $\Pi_{\mathsf{TE}}^{\mathsf{static}}$.
5. Extract the adversaries contributions: From the shares $(y_{i,j})_{P_j \in \mathcal{H}}$ sent by $P_i \in \mathcal{Q} \cap \mathcal{C}$ compute:

$$\overline{\alpha}_i := \sum_{P_j \in \mathcal{H}} y_{i,j} \cdot L_j^{\mathcal{H}}(0) \quad \left[= \alpha_i \cdot \Delta^2 \pmod{\mathrm{ord}(g_q)}\right] \text{ for all } P_i \in \mathcal{C} \cap \mathcal{Q}$$

$$\overline{y}_{l,i} := \sum_{P_j \in \mathcal{H}} y_{l,j} \cdot L_j^{\mathcal{H}}(i) \quad [= y_{j,i} \cdot \Delta \pmod{\mathrm{ord}(g_q)}] \text{ for all } P_l \in \mathcal{Q}, P_i \in \mathcal{C} \cap \mathcal{Q}$$

$$\overline{\gamma}_i := \sum_{P_l \in \mathcal{Q}} \overline{y}_{l,i} \quad [= \gamma_i \cdot \Delta \pmod{\mathrm{ord}(g_q)}] \text{ for all } P_i \in \mathcal{C} \cap \mathcal{Q}$$

6. Define $\mathsf{pk}_{\mathsf{cl}}$ and Γ_i for all $P_i \in \mathcal{Q}$ as in Step 3 of $\Pi_{\mathsf{TE}}^{\mathsf{static}}$.
7. Send $\delta := \sum_{P_i \in \mathcal{C} \cap \mathcal{Q}} \overline{\alpha}_i$ to complete $\mathsf{BiasedKeyGen}_{\Delta^2}^{\mathcal{S}}(\mathsf{pp}_{\mathsf{cl}})$ and prompt $\mathcal{F}_{\mathsf{TE}}$ to $\mathsf{continue}$.

Decrypt. On input $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$

1. Query $\mathcal{F}_{\mathsf{TE}}$ on behalf of all corrupt parties, receive (ct, M) and define

$$\hat{w}_0 := \mathsf{ct}_2 \cdot M^{-1} \quad [= \mathsf{ct}_1^{\mathsf{sk}_{\mathsf{cl}}}]$$
$$\hat{w}_i := \mathsf{ct}_1^{\overline{\gamma_i}} \text{ for all } P_i \in \mathcal{C}$$

2. Let $R := \{0\} \cup \mathcal{C}$. For all $P_i \in \mathcal{H}$:
   (a) Define

$$w_i := \hat{w}_0^{L_0^R(i)} \cdot \prod_{P_j \in \mathcal{C}} \hat{w}_j^{L_j^R(i)} \quad [= \mathsf{ct}_1^{\Delta^2 \cdot \gamma_i}]$$

   (b) Simulate $\pi_i \leftarrow \Pi_{\mathsf{EqDLog}}^{\mathsf{ZK}}.\mathsf{Simulate}((\mathsf{ct}_1, \Gamma_i, \overline{w}_i), \mathsf{chl})$.
3. Prompt $\mathcal{F}_{\mathsf{TE}}$ to $\mathsf{continue}$.

Description of the Simulator in Figure 6

Key Generation. The simulator queries $\mathcal{F}_{\mathsf{TE}}$ with (KeyGen) on behalf of every corrupt party and executes $\mathsf{BiasedKeyGen}^{\mathcal{S}}_{\Delta^2}$ with $\mathcal{F}_{\mathsf{TE}}$ upon receipt of the intermediate public key $\mathsf{pk}^*_{\mathsf{cl}}$. $\mathcal{S}$ engages in $\Pi^{\mathsf{static}}_{\mathsf{TE}}$ on behalf of $\mathcal{H}$ simulating all $P_i \neq P_h$ honestly. The messages from P_h are simulated to ensure that in the protocol execution the contribution of the honest parties to the public key is $\mathsf{pk}^*_{\mathsf{cl}}$:

$$\mathsf{pk}^*_{\mathsf{cl}} = \prod_{P_i \in \mathcal{H}} C_{i,0}.$$

The simulator for $\overline{\Pi}^{\mathsf{ZK}}_{\mathsf{DLog}}$ is employed to guarantee that the proofs sent by the distinguished P_h verify. The complaint phase is simulated honestly for all $P_i \in \mathcal{H}$, verifying the shares received from any $P_j \in \mathcal{C}$.

Since $\mathcal{S}$ controls the honest parties, it has sufficient shares to reconstruct the contributions of the corrupted parties. Using the weak reconstruction properties (Lemma 1), $\mathcal{S}$ recovers $\overline{\alpha}_i \in \mathbb{Z}$ such that $g_q^{\overline{\alpha}_i} = C_{i,0}^{\Delta^2}$ for each $P_i \in \mathcal{Q} \cap \mathcal{C}$.

Additionally, $\mathcal{S}$ computes the values $\overline{y}_{l,i}$ and $\overline{\gamma}_i$ for each $P_i \in \mathcal{C}$. The former is a Δ-multiple of the share $y_{l,i}$ that a party $P_l \in \mathcal{Q}$ would have sent to P_i. While we can compute this directly for all honest $P_l \in \mathcal{H}$, we do not *know* what a corrupted $P_l \in \mathcal{C} \cap \mathcal{Q}$ has sent to another corrupted party P_i. However, we can employ interpolation to compute a share that is consistent with the shares received by the honest parties, and the Δ-factor stems from using the integer Lagrange coefficients Then $\overline{\gamma}_i = \sum_{P_l \in \mathcal{Q}} \overline{y}_{l,i}$ is P_i's Shamir-share of the secret key. Note that we can compute these values only for corrupted $P_i \in \mathcal{C}$, since we have only at most t shares of the secret key, which is unknown to $\mathcal{S}$.

Then the adversarial contribution to the secret key is computed as

$$\delta := \sum_{P_i \in \mathcal{C} \cap \mathcal{Q}} \overline{\alpha}_i$$

and forwarded to $\mathcal{F}_{\mathsf{TE}}$, after which $\mathcal{S}$ prompts $\mathcal{F}_{\mathsf{TE}}$ to continue.

Distributed Decryption. On input the ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ the simulator queries $\mathcal{F}_{\mathsf{TE}}$ on behalf of all corrupted parties and receives the decryption (ct, M) from the functionality. The leaked message is used to recover the partial decryptions to be sent by the honest parties in the protocol execution. First, we compute

$$w_0 := \left(\mathsf{ct}_2 \cdot M^{-1}\right)^{\Delta} \qquad [= g_q^{r \cdot \mathsf{sk}_{\mathsf{cl}} \cdot \Delta}].$$

Then we use w_0 and the corrupted parties' shares $\overline{\gamma}_i$ to compute the partial decryptions for the honest parties by interpolation in the exponent, and simulate the corresponding zero-knowledge proofs.

Guaranteed Output Delivery. Assuming an honest majority an adversary cannot prevent the successful completion of the protocol as the simulator $\mathcal{S}$ always instructs $\mathcal{F}_{\mathsf{TE}}$ to continue and deliver output to the honest parties.

Indistinguishability of the Simulation

Public Parameters $\mathsf{pp}_{\mathsf{cl}}$. The simulated public parameters are those output by $\mathcal{F}_{\mathsf{TE}}$ whereas the public parameters in the real protocol execution are those output by $\mathcal{F}_{\mathsf{CL}}$. As both functionalities compute them in the same way $\mathsf{pp}_{\mathsf{cl}} \leftarrow \mathsf{CLGen}(1^\lambda, q; \mathcal{U}(\{0,1\}^\lambda))$ from the queried inputs these values are identically distributed.

We prove security under the RO_C assumption. Hence, for the remainder of this proof, we assume the public parameters were instead generated as $\mathsf{pp}_{\mathsf{cl}} \leftarrow \mathsf{CLGen}(1^\lambda, q; \mathcal{D}_C^{\mathsf{rough}}(\lambda, q))$. This allow us to use the unconditional soundness of the zero-knowledge proofs $\overline{\Pi}^{\mathsf{ZK}}_{\mathsf{DLog}}$ and $\Pi^{\mathsf{ZK}}_{\mathsf{EqDLog}}$ in C-rough-order groups. By applying Theorem 7 we obtain security in the standard case.

Key Generation. For all honest parties $P_i \in \mathcal{H} \setminus \{P_h\}$ the messages sent by the simulated parties are identically distributed to those in the real protocol execution. The simulation deviates with the behavior of P_h in Step 3. P_h also shares a random value α_h, but instead of setting $C_{h,0} := g_q^{\alpha_h}$ as in the real protocol, it chooses the public commitment $C_{h,0}$ such that it is correctly distributed given the intermediate public key $\mathsf{pk}_{\mathsf{cl}}^*$ received from $\mathcal{F}_{\mathsf{TE}}$, but has an unknown discrete logarithm to base g_q. The remaining broadcast values $(C_{h,k})_{k\in[t]}$ are simulated with Lemma 2 to be consistent with $C_{h,0}$ and the shares (of α_h) sent to the corrupted parties $(y_{h,i})_{P_i\in\mathcal{C}}$. Since at most t shares are observed and the zero-knowledge proofs π_h sent by P_h are simulated with statistical zero-knowledge of $\overline{\Pi}^{\mathsf{ZK}}_{\mathsf{DLog}}$, this change is undetectable by any $\mathcal{Z}$.

Note that for every $P_i \in \mathcal{Q} \cap \mathcal{C}$, $\mathcal{S}$ received at least $t+1$ shares (of values α_i) that are consistent with F-Check and where the zero-knowledge proof π_i for $\mathsf{R}_{\mathsf{DLog}}$ asserts that $C_{i,k} \in \langle g_q \rangle$ for $k \in [0,t]$, unless $\mathcal{Z}$ has managed to forge a proof which happens with negligible probability. Therefore, a Δ^2-multiple $\overline{\alpha}$ of these values can be recovered from the received shares to compute the adversarial key contribution via interpolation with integer Lagrange coefficients (Lemma 1). Hence, we have

$$\delta := \sum_{P_j\in\mathcal{Q}\cap\mathcal{C}} \overline{\alpha}_j = \sum_{P_j\in\mathcal{Q}\cap\mathcal{C}} \sum_{P_i\in\mathcal{H}} y_{j,i} \cdot L_i^{\mathcal{H}}(0) \quad \text{such that} \quad \prod_{P_j\in\mathcal{Q}\cap\mathcal{C}} C_{j,0}^{\Delta^2} = g_q^{\delta}. \tag{19}$$

The bias δ input to $\mathcal{F}_{\mathsf{TE}}$ is then equal to the corrupt key contribution in the protocol execution and thus the keys output by $\mathcal{F}_{\mathsf{TE}}$ to the honest parties are identically distributed to those computed in the real protocol execution:

$$\mathsf{pk}_{\mathsf{cl}} = (\mathsf{pk}_{\mathsf{cl}}^*)^{\Delta^2} \cdot g_q^{\delta} = \left(\prod_{P_i\in\mathcal{H}} C_{i,0}\right)^{\Delta^2} \cdot \left(\prod_{P_i\in\mathcal{C}\cap\mathcal{Q}} C_{i,0}^{\Delta^2}\right) = \prod_{P_i\in\mathcal{Q}} C_{i,0}^{\Delta^2}.$$

Distributed Decryption. Note that $\mathcal{S}$ does not know the secret key and, therefore, cannot compute secret shares γ_i for the simulated $P_i \in \mathcal{H}$ that are consistent with both the secret key and the (at most t) shares that the corrupted parties have received. This is due to $\mathcal{S}$ having to simulate the contribution of P_h. Therefore, these values cannot be used to compute partial decryptions. Instead, the decrypted message $M = \mathsf{ct}_2 \cdot \mathsf{ct}_1^{-\mathsf{sk}_{\mathsf{cl}}}$ leaked to $\mathcal{S}$ by $\mathcal{F}_{\mathsf{TE}}$ and the shares $(\overline{\gamma}_i)_{P_i\in\mathcal{C}}$ of the corrupted parties are used to compute partial decryptions $(\overline{w}_i)_{P_i\in\mathcal{H}}$ that can correctly be aggregated to a decryption of message M.

Note that $\overline{\gamma}_i$ are Δ-multiples $\pmod{\mathrm{ord}(g_q)}$ of the shares γ_i that the P_i received. The original sharing polynomial has the constant term $\Delta \cdot \sum_{P_j\in\mathcal{Q}} \alpha_j$, but the secret key is actually defined as $\mathsf{sk}_{\mathsf{cl}} = \Delta^2 \cdot \sum_{P_j\in\mathcal{Q}} \alpha_j$. Thus, the $(\overline{\gamma}_i)_{P_i\in\mathcal{C}}$ are actually points of an (unknown) polynomial $\mathsf{f}(X) \in \mathbb{Z}[X]_{\leq t}$ that has constant term $\mathsf{sk}_{\mathsf{cl}}$:

$$\mathsf{f}(0) = \mathsf{sk}_{\mathsf{cl}} \quad \text{and} \quad \mathsf{f}(i) = \overline{\gamma}_i \text{ for } P_i \in \mathcal{C}.$$

Setting $R := \{0\} \cup \mathcal{C}$ and interpolating over the integers gives us

$$\Delta \cdot \mathsf{f}(X) = L_0^R(X) \cdot \mathsf{f}(0) + \sum_{P_i\in\mathcal{C}} L_i^R(X) \cdot \mathsf{f}(i) = L_0^R(X) \cdot \mathsf{sk}_{\mathsf{cl}} + \sum_{P_i\in\mathcal{C}} L_i^R(X) \cdot \overline{\gamma}_i.$$

In Step 1 of the decryption protocol in $\Pi^{\mathsf{static}}_{\mathsf{TE}}$, each party P_i is supposed to send

$$w_i := \mathsf{ct}_1^{\Delta^2 \cdot \gamma_i} = \mathsf{ct}_1^{\Delta \cdot \overline{\gamma}_i} = \mathsf{ct}_1^{\Delta \cdot \mathsf{f}(i)}.$$

By knowing $\overline{\gamma}_i$ for $P_i \in \mathcal{C}$, we can also compute

$$\hat{w}_i := \mathsf{ct}_1^{\overline{\gamma}_i} = \mathsf{ct}_1^{\mathsf{f}(i)} \quad \text{for } P_i \in \mathcal{C}.$$

We also have

$$\hat{w}_0 := \mathsf{ct}_2 \cdot M^{-1} = \mathsf{ct}_1^{\mathsf{sk}_{\mathsf{cl}}} = \mathsf{ct}_1^{\mathsf{f}(0)},$$

and can compute the matching w_i for $P_i \in \mathcal{H}$ by interpolation in the exponent:

$$w_i = \hat{w}_0^{L_0^R(i)} \cdot \prod_{P_j\in\mathcal{C}} \hat{w}_j^{L_j^R(i)} = \mathsf{ct}_1^{\sum_{j\in R} L_j^R(i) \cdot \mathsf{f}(j)} = \mathsf{ct}_1^{\Delta \cdot \mathsf{f}(i)}.$$

This is correct because for any set S of at least $t+1$ valid partial decryptions, we have

$$W := \prod_{P_j \in S} w_j^{L_j^S(0)} = \mathsf{ct}_1^{\sum_{P_j \in S} L_j^S(0) \cdot \Delta \cdot \mathsf{f}(i)} = \mathsf{ct}_1^{\Delta^2 \cdot \mathsf{f}(i)} = \mathsf{ct}_1^{\Delta^2 \cdot \mathsf{sk}_{\mathsf{cl}}}.$$

Hence, the w_i are exactly those values that $\mathcal{Z}$ expects to see given the plaintext M. Moreover, the simulated proofs $(\pi_i)_{P_i \in \mathcal{H}}$ are indistinguishable by the statistical zero-knowledge of $\Pi^{\mathsf{ZK}}_{\mathsf{EqDLog}}$. □

7 Protocol with Adaptive Security

7.1 Protocol

In Figure 7 we present $\Pi^{\mathsf{adaptive}}_{\mathsf{TE}}$, a threshold encryption protocol secure against *adaptive* corruption in the SIP model. In the SIP model corruption of the single inconsistent party P_h causes the simulation to fail. For this failure probability to be bounded by $\frac{t}{N} + \mathsf{negl}(\sigma)$ the messages of P_h must be statistically indistinguishable from those of the consistent honest parties. We modify the statically secure protocol $\Pi^{\mathsf{static}}_{\mathsf{TE}}$ to meet this requirement.[18] Following prior work in the settings of prime-order and RSA groups [Can+99; JL00; LP01; AF04] we combine (a variant of) Pedersen VSS with additive secret sharing. The protocol is in the $\mathcal{F}_{\mathsf{CL\text{-}Ped}}$-hybrid model; prior to key generation parties query $\mathcal{F}_{\mathsf{CL\text{-}Ped}}$ to receive CL parameters $\mathsf{pp}_{\mathsf{cl}}$ and an additional generator h_{P}. The functionality $\mathcal{F}_{\mathsf{CL\text{-}Ped}}$ is both defined and instantiated in Section 7.3.

Key Generation In $\Pi^{\mathsf{static}}_{\mathsf{TE}}$ parties share their contributions α_i to the secret key with Feldman VSS. This includes broadcasting commitments $(C_{i,k})_{k\in[0,t]}$ to the coefficients of the sharing polynomial, and in particular broadcasting $C_{i,0} = g_q^{\alpha_i}$ as a multiplicative contribution to the public key $\mathsf{pk}_{\mathsf{cl}} := \prod_{P_i \in \mathcal{Q}} C_{i,0}^{\Delta^2}$. The simulator must generate honest public key contributions such that the simulated protocol computes $\mathsf{pk}_{\mathsf{cl}}$ output by $\mathcal{F}_{\mathsf{TE}}$ but does not know the secret key $\mathsf{sk}_{\mathsf{cl}} = \Delta^2 \cdot \sum_{P_i \in \mathcal{Q}} \alpha_i$. Therefore, there must be (at least) one honest P_h whose contribution α_h remains unknown to the simulator . The simulator then must *fake the sharing* of α_h by simulating public values $(C_{h,k})_{k\in[t]}$ and corrupt shares $(y_{h,i})_{P_i \in \mathcal{C}}$ that are consistent with $C_{h,0} \in \langle g_q \rangle$ (with unknown discrete logarithm). The simulator for $\Pi^{\mathsf{static}}_{\mathsf{TE}}$ shares an arbitrary value and simulates $(C_{h,k})_{k\in[t]}$ that are consistent with $C_{h,0}$ and the corrupt shares. This is done by interpolation in the exponent of the points $\{0\} \cup \{i \mid P_i \in \mathcal{C}\}$.

In the *adaptive* case this simulation strategy no longer works because $\mathcal{Z}$ can adaptively corrupt parties after the simulated sharing and, because any fixed $C_{h,0}, \ldots, C_{h,t}$ is consistent with at most t faked shares, this strategy essentially requires the simulator to guess which t parties $\mathcal{Z}$ will corrupt. Instead we use a two-stage approach for distributed key generation. First, parties use the statistically hiding Pedersen VSS to share α_i and, in particular, broadcast a Pedersen commitment to their key contribution. In a second round each party broadcasts $D_i := g_q^{\alpha_i}$ with a zero-knowledge proof of knowledge that this value matches the committed α_i from their first round. The simulator still does not know α_h for the inconsistent party P_h so the proof sent by P_h is simulated with the statistical zero-knowledge simulator. The inconsistency is statistically hidden so the probability that $\mathcal{Z}$ corrupts P_h is only $\mathsf{negl}(\sigma)$ greater than the probability that $\mathcal{Z}$ corrupts any other party.

Decryption In $\Pi^{\mathsf{static}}_{\mathsf{TE}}$ during decryption parties partially decrypt ciphertexts with Shamir-style shares of the secret key. The partial decryptions commit the parties to their points on the sharing polynomial. Thus during simulation the simulator commits to $> t$ points, not all of which can be consistent with the unknown secret key. If $\mathcal{Z}$ corrupts an inconsistent party then the simulator cannot generate a view corresponding to that party that includes a share which explains all of their partial decryptions. This is clearly problematic in the case of adaptive corruption.

Instead our adaptive protocol uses the additive shares α_i and parties reconstruct any potentially missing shares during decryption. Using this additive sharing the simulator can generate consistent views for all parties except P_h. For P_h the simulator computes a partial decryption from the plaintext sent by $\mathcal{F}_{\mathsf{TE}}$ and simulates the corresponding zero-knowledge proof.

[18] We note that the simulator for $\Pi^{\mathsf{static}}_{\mathsf{TE}}$ also uses an inconsistent party P_h, but – as that protocol is secure against *static* corruption – the inconsistent party can be chosen after $\mathcal{Z}$ has selected the corrupted parties and the simulation never fails.

Figure 7: Protocol $\Pi_{\mathsf{TE}}^{\mathsf{adaptive}}$

The parties maintain a set $\mathcal{Q}$ of qualified parties initially containing all parties.

Init. Send $(\mathsf{Gen}, 1^\lambda, q)$ to $\mathcal{F}_{\mathsf{CL\text{-}Ped}}$ and receive $(\mathsf{pp}_{\mathsf{cl}} = (q, \bar{s}, f, g_q, \widehat{G}, F; \rho), h_{\mathsf{P}})$.

KeyGen. All P_i proceed in parallel:

1. Generation of a key pair $(\mathsf{pk}_{\mathsf{cl}}, \mathsf{sk}_{\mathsf{cl}})$.
   (a) Sample $\alpha_i \leftarrow \mathcal{D}_q$.
   (b) Share α_i as $(y_{i,1}, \ldots, y_{i,N}, z_{i,1}, \ldots, z_{i,N}; C_{i,0}, \ldots, C_{i,t}) \leftarrow \mathsf{P\text{-}Share}(\alpha_i; \beta_i, \mathbf{r}_i, \mathbf{s}_i; g_q, h_{\mathsf{P}})$.
   (c) Compute $\pi_i^{(1)} \leftarrow \overline{\Pi}_{\mathsf{Ped}}^{\mathsf{ZK}}.\mathsf{Prove}((g_q, h_{\mathsf{P}}, (C_{i,k})_{j\in[0,t]}), ((\alpha_i, \beta_i), (r_{i,1}, s_{i,1}), \ldots, (r_{i,t}, s_{i,t}))$.
   (d) Broadcast $(C_{i,k})_{k\in[0,t]}$ and $\pi_i^{(1)}$. Privately send $(y_{i,j}, z_{i,j})$ to each P_j.
   (e) Verify the shares received from $P_j \neq P_i$: Check if
      i. $\mathsf{P\text{-}Check}(i, y_{j,i}, z_{j,i}; C_{j,0}, \ldots, C_{j,t}; g_q) \stackrel{?}{=} \top$, and
      ii. $\overline{\Pi}_{\mathsf{Ped}}^{\mathsf{ZK}}.\mathsf{Verify}((g_q, h_{\mathsf{P}}, (C_{j,k})_{k\in[0,t]}), \pi_j^{(1)}) \stackrel{?}{=} \top$.

   If Step 1(e)i failed, but Step 1(e)ii succeeded, then broadcast a complaint against P_j.
   (f) For every complaint received by $P_j \neq P_i$, broadcast the values $(y_{i,j}, z_{i,j})$.
   (g) If Step 1(e)ii failed for P_j, or if P_j broadcasted values $(y_{j,l}, z_{j,l})$ in response to P_l that do not satisfy Step 1(e)i, remove P_j from $\mathcal{Q}$.
2. Extracting the public key contributions.
   (a) Set $D_i := g_q^{\Delta^2 \cdot \alpha_i}$ and $\pi_i^{(2)} \leftarrow \Pi_{\mathsf{Ped\text{-}DLog}}^{\mathsf{PoK}}.\mathsf{Prove}((g_q, h_{\mathsf{P}}, C_{i,0}, D_i), (\alpha_i, \beta_i))$.
   (b) Broadcast $(D_i, \pi_i^{(2)})$.
   (c) Verify the proofs received from all $P_j \in \mathcal{Q} \setminus \{P_i\}$: $\Pi_{\mathsf{Ped\text{-}DLog}}^{\mathsf{PoK}}.\mathsf{Verify}((g_q, h_{\mathsf{P}}, C_{j,0}, D_j), \pi_j^{(2)}) \stackrel{?}{=} \top$.
   (d) If $\pi_j^{(2)}$ does not verify, broadcast $z_{j,i}$ and apply Lemma 4 to extract $\overline{\alpha}_j$ and compute $D_j := g_q^{\overline{\alpha}_j}$.
3. Compute the public key $\mathsf{pk}_{\mathsf{cl}}$ for the (implicitly defined) secret key $\mathsf{sk}_{\mathsf{cl}}$:

$$\mathsf{pk}_{\mathsf{cl}} := \prod_{P_j \in \mathcal{Q}} D_j \quad \text{with} \quad \mathsf{sk}_{\mathsf{cl}} := \Delta^2 \cdot \sum_{P_j \in \mathcal{Q}} \alpha_j.$$

Decrypt To jointly decrypt a $\Pi_{\mathsf{hsm\text{-}cl}}$ ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2) \in \widehat{G}^2$, all P_i proceed in parallel as follows:

1. Compute $w_i := \mathsf{ct}_1^{\Delta^2 \cdot \alpha_i}$ and $\pi_i \leftarrow \Pi_{\mathsf{EqDLog}}^{\mathsf{ZK}}.\mathsf{Prove}((g_q, \mathsf{ct}_1, D_i, w_i), \Delta^2 \cdot \alpha_i)$. Broadcast w_i and π_i.
2. Define $S := \{P_j \in \mathcal{Q} \mid \Pi_{\mathsf{EqDLog}}^{\mathsf{ZK}}.\mathsf{Verify}((g_q, \mathsf{ct}_1, D_j, w_j), \pi_j) = \top\}$.
3. For all $P_j \in \mathcal{Q} \setminus S$: apply Lemma 4 to extract $\overline{\alpha}_j$ and compute $w_j := \mathsf{ct}_1^{\overline{\alpha}_j}$.
4. Compute $W := \prod_{P_j \in \mathcal{Q}} w_j$ and $M := \mathsf{ct}_2 \cdot W^{-1}$.
5. Output $m := \mathsf{CLSolve}(\mathsf{pp}_{\mathsf{cl}}, M)$.

7.2 Security Proof

Theorem 12 (UC-SIP Security of $\Pi_{\mathsf{TE}}^{\mathsf{adaptive}}$ in Figure 7). *The protocol $\Pi_{TE}^{adaptive}$ UC-SIP-realizes the ideal functionality $\mathcal{F}_{TE}$ with computational and active security, tolerating up to $t < N/2$ adaptive corruptions and providing guaranteed output delivery in the $\mathcal{F}_{\mathsf{CL\text{-}Ped}}$-hybrid model given secure channels and broadcast under the ORD and RO_C assumptions.*

Proof. First, we show correctness. Second, we present a simulator for the protocol and prove indistinguishability of the simulation.

Correctness Disqualification is dependent upon broadcast information and thus all honest parties agree on the set of qualified parties $\mathcal{Q}$ by the end of Step 1g. Moreover, every honest party executes the protocol honestly and therefore is not disqualified.

During the first round of key generation, Step 1, every honest party $P_i \in \mathcal{H}$ samples and shares a value α_i with P-Share. By the end of Step 1g each honest P_i has received a verifying share $(y_{i,j}, z_{i,j})$ from each qualified $P_j \in \mathcal{Q}$ and thus the honest parties collectively hold at least $|\mathcal{H}| \geq t+1$ consistent shares of the (α_j, β_j) shared by each qualified $P_j \in \mathcal{Q}$. By the weak reconstruction property of the VSS (Lemma 4) the shares collectively held by $\mathcal{H}$ permit the efficient recovery of the values

$$(\overline{\alpha}_j, \overline{\beta}_j) \quad \text{such that} \quad C_{0,j}^{\Delta^2} = g_q^{\overline{\beta}_j} \cdot h_{\mathsf{P}}^{\overline{\alpha}_j} \quad \text{for all } P_j \in \mathcal{Q}.$$

During the second round of key generation, Step 2, every qualified $P_i \in \mathcal{Q}$ either reveals $D_i = g_q^{\overline{\alpha}_i}$ or such a D_i is extracted from the shares distributed in the first round. At least the $t+1$ parties in $\mathcal{H}$ participate to extract each D_i in the case that P_i did not participate. By the end of Step 2d the parties hold the values

$$D_j := g_q^{\overline{\alpha}_j} \quad \text{for all } P_j \in \mathcal{Q}.$$

Parties hold their additive share of the secret key and compute the public key as follows:

$$\mathsf{pk}_{\mathsf{cl}} := \prod_{P_i \in \mathcal{Q}} D_i = \prod_{P_i \in \mathcal{Q}} g_q^{\overline{\alpha}_i} = g_q^{\sum_{P_i \in \mathcal{Q}} \overline{\alpha}_i} = g_q^{\mathsf{sk}_{\mathsf{cl}}}.$$

Moreover, party P_i holds a threshold share $y_{i,j}$ of the value $\overline{\alpha}_j$ for each $P_j \in \mathcal{Q}$.

For the ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2) = (g_q^r, \mathsf{pk}_{\mathsf{cl}}^r \cdot f^m)$ every P_i broadcasts $w_i := \mathsf{ct}_1^{\Delta^2 \cdot \alpha_i}$ and a proof π_i. By the end of Step 3 every party $P_j \in \mathcal{Q}$ has either broadcast w_j or the value $w_j = \mathsf{ct}_1^{\overline{\alpha}_j}$ has been computed from the threshold shares of $\overline{\alpha}_j$. At least the $t+1$ parties in $\mathcal{H}$ send the consistent shares received during key generation, which can be verified against the public values broadcast during key generation. Therefore, the value $\mathsf{ct}_1^{\mathsf{sk}_{\mathsf{cl}}}$ can always be computed. Parties output $\mathsf{CLSolve}(\mathsf{pp}_{\mathsf{cl}}, M)$, for

$$M := \mathsf{ct}_2 \cdot \left(\mathsf{ct}_1^{\mathsf{sk}_{\mathsf{cl}}}\right)^{-1} \quad [= f^m].$$

Indistinguishability of the Simulation The simulator $\mathcal{S}$ is defined in Figure 8. The simulator $\mathcal{S}$ simulates all honest parties $P_i \in \mathcal{H} \setminus \{P_h\}$ according to the real protocol. Consequently, these simulated messages are always distributed correctly and, upon adaptive corruption by $\mathcal{Z}$, $\mathcal{S}$ can present a valid view with the correct distribution. We focus on the simulated inconsistent party P_h, proving that the messages sent by the simulator are distributed statistically close to what P_h would send in the real protocol. Since P_h was chosen randomly in the beginning, no environment corrupts it (causing the simulation to fail) with probability greater than $\frac{t}{N} + \mathsf{negl}(\sigma)$.

Public Parameters $(\mathsf{pp}_{\mathsf{cl}}, h_{\mathsf{P}})$. In contrast to the statically secure protocol $\Pi_{\mathsf{TE}}^{\mathsf{static}}$, we need in addition to the CL public parameters $\mathsf{pp}_{\mathsf{cl}}$ also a Pedersen public key h_{P}. In the protocol $\Pi_{\mathsf{TE}}^{\mathsf{adaptive}}$ both are generated by $\mathcal{F}_{\mathsf{CL\text{-}Ped}}$, which allows the adversary to bias h_{P}. In the simulation both are generated in the same way: $\mathsf{pp}_{\mathsf{cl}}$ is chosen by $\mathcal{F}_{\mathsf{TE}}$ and h_{P} is sampled by $\mathcal{S}$ running the $\mathsf{BiasedComGen}^{\mathcal{Z}}$ with $\mathcal{Z}$ such that it knows the corresponding trapdoor. Hence, both are distributed the same way.

As in the proof of Theorem 11 we again use the RO_C assumption and apply Theorem 7: By assuming the public parameters were generated as $\mathsf{pp}_{\mathsf{cl}} \leftarrow \mathsf{CLGen}(1^\lambda, q; \mathcal{D}_C^{\mathsf{rough}}(\lambda, q))$, we can use the unconditional soundness of $\overline{\Pi}_{\mathsf{Ped}}^{\mathsf{ZK}}$(in KeyGen and $\Pi_{\mathsf{EqDLog}}^{\mathsf{ZK}}$ in Decrypt. The knowledge soundness of $\Pi_{\mathsf{Ped\text{-}DLog}}^{\mathsf{PoK}}$ is independent of this since we instantiate the proof with binary challenges and repetitions.

Figure 8: Simulator for $\Pi^{\mathsf{adaptive}}_{\mathsf{TE}}$

Let $\mathcal{C}$ and $\mathcal{H}$ denote, at any time, the sets of currently corrupt and honest parties, respectively. Let $P_h \in \mathcal{H}$ be the single inconsistent party that is randomly chosen upon initialization.

Init. Simulate the call to $\mathcal{F}_{\mathsf{CL\text{-}Ped}}$.

1. Forward each query $(\mathsf{Gen}, 1^\lambda, q)$ from $P_c \in \mathcal{C}$ as an Init query to $\mathcal{F}_{\mathsf{TE}}$ and receive $\mathsf{pp}_{\mathsf{cl}}$.
2. Execute the Pedersen setup with $\mathcal{Z}$: Send $h^*_{\mathsf{P}} := g_q^\omega$ for $\omega \leftarrow \mathcal{D}_q$ to $\mathcal{Z}$ and receive δ. Define $h_{\mathsf{P}} := g_q^{\omega+\delta}$.
3. Simulate $\mathcal{F}_{\mathsf{CL\text{-}Ped}}$ and output $(\mathsf{pp}_{\mathsf{cl}}, h_{\mathsf{P}})$ to each corrupt party.

KeyGen. Simulate the generation of $\mathsf{pk}_{\mathsf{cl}}$ by executing $\mathsf{BiasedKeyGen}^{\mathcal{S}}_{\Delta^2}$ with $\mathcal{F}_{\mathsf{TE}}$.

1. Send (KeyGen) to $\mathcal{F}_{\mathsf{TE}}$ and receive the intermediate public key $\mathsf{pk}^*_{\mathsf{cl}}$.
2. For all $P_i \in \mathcal{H}$: simulate Step 1 according to $\Pi^{\mathsf{adaptive}}_{\mathsf{TE}}$. Let $\mathcal{Q} \supseteq \mathcal{H}$ be the set of qualified parties.
3. For each $P_i \in \mathcal{Q} \setminus \{P_h\}$: apply Lemma 4 to the shares $\{(y_{i,j}, z_{i,j})\}_{P_j \in \mathcal{H}}$ and values $C_{i,0}, \ldots, C_{i,t}$ to extract $\overline{\alpha}_i, \overline{\beta}_i \in \mathbb{Z}$ such that $C_{i,0}^{\Delta^2} = g_q^{\overline{\beta}_i} \cdot h_{\mathsf{P}}^{\overline{\alpha}_i}$.
4. For all $P_i \in \mathcal{H} \setminus \{P_h\}$: simulate Step 2a according to $\Pi^{\mathsf{adaptive}}_{\mathsf{TE}}$.
5. For P_h: compute $D_h := (\mathsf{pk}^*_{\mathsf{cl}})^{\Delta^2} \cdot \prod_{P_i \in \mathcal{Q} \setminus \{P_h\}} g_q^{-\overline{\alpha}_i}$ and $\pi_h^2 \leftarrow \Pi^{\mathsf{PoK}}_{\mathsf{Ped\text{-}DLog}}.\mathsf{SimProve}((g_q, h_{\mathsf{P}}, C_{i,0}, D_i))$.
6. For all $P_i \in \mathcal{H}$: simulate the broadcast of $(D_i, \pi_i^{(2)})$.
7. Send $\delta := 0$ to $\mathcal{F}_{\mathsf{TE}}$ and prompt the functionality to $\mathsf{continue}$.

Decrypt. On input $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$:

1. Query $\mathcal{F}_{\mathsf{TE}}$ on behalf of all corrupt parties and receive (ct, M) from $\mathcal{F}_{\mathsf{TE}}$.
2. Compute $W := \mathsf{ct}_2 \cdot M^{-1}$.
3. For all $P_i \in \mathcal{Q} \setminus \{P_h\}$: compute $w_i := \mathsf{ct}_1^{\overline{\alpha}_i}$.
4. For $P_i \in \mathcal{H} \setminus \{P_h\}$: compute $\pi_i \leftarrow \Pi^{\mathsf{ZK}}_{\mathsf{EqDLog}}.\mathsf{Prove}((g_q, \mathsf{ct}_1, D_i, w_i), \Delta^2 \cdot \alpha_i)$.
5. For P_h: instead, compute $w_h := W \cdot \prod_{P_i \in \mathcal{Q} \setminus \{P_h\}} w_i^{-1}$ and $\pi_h \leftarrow \Pi^{\mathsf{ZK}}_{\mathsf{EqDLog}}.\mathsf{SimProve}((g_q, \mathsf{ct}_1, D_h, w_h))$.
6. For all $P_i \in \mathcal{H}$: simulate the broadcast of (w_i, π_i).
7. Continue as in $\Pi^{\mathsf{adaptive}}_{\mathsf{TE}}$.
8. Prompt $\mathcal{F}_{\mathsf{TE}}$ to $\mathsf{continue}$.

Key Generation. As discussed above, $\mathcal{S}$ lets all honest parties $P_i \neq P_h$ behave as in $\Pi^{\mathsf{adaptive}}_{\mathsf{TE}}$. For Step 1 of $\Pi^{\mathsf{adaptive}}_{\mathsf{TE}}$ it also lets P_h follow the protocol by sharing a random value α_h using the Pedersen VSS. Then in Step 3 of the simulation, $\mathcal{S}$ uses Lemma 4 to extract the qualified parties' contributions to the secret key from the honest parties' VSS shares. This is well-defined unless $\mathcal{Z}$ managed to forge a proof for $\overline{\Pi}^{\mathsf{ZK}}_{\mathsf{Ped}}$ which happens with negligible probability.

In the second part of the simulation, $\mathcal{S}$ chooses P_h's contribution to the public key D_h independently of the shared value α_h such that the simulated protocol outputs the public key generated by $\mathcal{F}_{\mathsf{TE}}$. By statistical privacy of the Pedersen VSS (Lemma 5) and statistical zero-knowledge of $\Pi^{\mathsf{PoK}}_{\mathsf{Ped\text{-}DLog}}$, the adversary's view in the simulation is statistically indistinguishable from its view in the real protocol execution.

The output of distributed key generation is

$$\mathsf{pk}_{\mathsf{cl}} = \prod_{P_i \in \mathcal{Q}} D_i,$$

where each D_i is either broadcast by P_i with a proof of knowledge w.r.t. $\mathsf{R}_{\mathsf{Ped\text{-}DLog}}$, or recomputed as $D_i := g_q^{\overline{\alpha}_i}$ where $\overline{\alpha}_i$ is extracted from the Pedersen VSS via Lemma 4. In the simulation $\mathcal{F}_{\mathsf{TE}}$ outputs $\mathsf{pk}_{\mathsf{cl}} := (\mathsf{pk}^*_{\mathsf{cl}})^{\Delta^2}$ as $\mathcal{S}$ defines the bias as $\delta := 0$ and computes the key contribution D_h so that the simulated protocol outputs

$$\mathsf{pk}'_{\mathsf{cl}} = (\mathsf{pk}^*_{\mathsf{cl}})^{\Delta^2} \cdot \left(\prod_{P_i \in \mathcal{Q} \setminus \{P_h\}} g_q^{-\overline{\alpha}_i} \right) \cdot \left(\prod_{P_i \in \mathcal{Q} \setminus \{P_h\}} D_i \right),$$

where the $\overline{\alpha}_i$ are reconstructed by $\mathcal{S}$ from the shares received by the honest parties during the VSS.

Claim. Assuming *ORD*, no party P_i can produce, except with negligible probability,

a) valid shares $((y_{i,j})_{j \in \mathcal{H}}, (C_{i,k})_{k \in [0,t]})$ and proof $\pi_i^{(1)}$ in Step 1b of $\Pi^{\mathsf{adaptive}}_{\mathsf{TE}}$, and

b) $D_i \in \widehat{G}$ with a valid proof $\pi_i^{(2)}$ in Step 2a of $\Pi_{\mathsf{TE}}^{\mathsf{adaptive}}$,
such that $D_i \neq g_q^{\overline{\alpha}_i}$ where $\overline{\alpha}_i, \overline{\beta}_i$ can be extracted from the shares.

Proof of Claim. Assume towards contradiction that an environment $\mathcal{Z}$ can make P_i produce the values stated in the claim with non-negligible probability. Then we can construct an algorithm $\mathcal{B}$ that breaks the binding property of the Pedersen commitments (Theorem 3) as follows: It runs the simulation of the key generation, applies Lemma 4 to extract $\overline{\alpha}_i, \overline{\beta}_i \in \mathbb{Z}$ from the VSS (possible due to condition a)), and uses the knowledge soundness property of $\Pi_{\mathsf{Ped\text{-}DLog}}^{\mathsf{PoK}}$ to extract $\alpha_i, \beta_i \in \mathbb{Z}$ from $\pi_i^{(2)}$ (e.g., by rewinding and reprogramming the random oracle if a Fiat-Shamir-transformed Σ protocol is used). It outputs $(\Delta^2 \cdot \alpha_i, \overline{\alpha}_i, \Delta^2 \cdot \beta_i, \overline{\beta}_i))$.

Note that we have $C_{i,0}^{\Delta^2} = g_q^{\overline{\beta}_i} \cdot h_{\mathsf{P}}^{\overline{\alpha}_i}$ from the VSS, as well as $C_{i,0} = g_q^{\beta_i} \cdot h_{\mathsf{P}}^{\alpha_i}$ and $D_i = g_q^{\Delta^2 \cdot \alpha_i}$ due to $\Pi_{\mathsf{Ped\text{-}DLog}}^{\mathsf{PoK}}$. Moreover $\Delta^2 \cdot \alpha_i \neq \overline{\alpha}_i$ by our assumption of inconsistency. Therefore, $\mathcal{B}$ outputs the two openings for the commitment $C_{i,0}^{\Delta^2} = \mathsf{Com}(\Delta^2 \cdot \alpha_i; \Delta^2 \cdot \beta_i) = \mathsf{Com}(\overline{\alpha}_i; \overline{\beta}_i)$. ■

Note that the output of $\mathcal{F}_{\mathsf{TE}}$ and the simulated execution of $\Pi_{\mathsf{TE}}^{\mathsf{adaptive}}$ are equal if $D_i = g_q^{\overline{\alpha}_i}$ for all $P_i \in \mathcal{Q} \setminus \{P_h\}$.Hence the messages sent by P_h are exactly what $\mathcal{Z}$ expects given the public key $\mathsf{pk}_{\mathsf{cl}}$ output by $\mathcal{F}_{\mathsf{TE}}$. Moreover, by the statistical privacy of the Pedersen VSS (Lemma 5) and the statistical zero-knowledge of the proofs, the messages sent by P_h during simulation are statistically indistinguishable from those sent during a real protocol execution.

Distributed Decryption. When decrypting a ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ the simulator receives $M = \mathsf{ct}_2 \cdot \mathsf{ct}_1^{-\mathsf{sk}_{\mathsf{cl}}}$ from $\mathcal{F}_{\mathsf{TE}}$. In $\Pi_{\mathsf{TE}}^{\mathsf{adaptive}}$, parties P_i broadcast their partial decryptions $w_i = \mathsf{ct}_1^{\Delta^2 \cdot \alpha_i}$ such that $W := \mathsf{ct}_1^{\mathsf{sk}_{\mathsf{cl}}} = \prod_{P_i \in \mathcal{Q}} w_i$. In the simulation, the partial decryption w_h broadcast by the inconsistent party P_h must be faked. First, from the leaked M the simulator computes $W := \mathsf{ct}_2 \cdot M^{-1}$. For all parties $P_i \neq P_h$, the values $\overline{\alpha}_i$ that were extracted during key generation can be used by $\mathcal{S}$ to compute the values w_i that should be sent in Step 1 (or recovered in Step 3 if P_i does not participate in decryption). By the claim above $D_i = g_q^{\overline{\alpha}_i}$ and by the soundness of $\Pi_{\mathsf{EqDLog}}^{\mathsf{ZK}}$ these partial decryptions are such that $w_i = \mathsf{ct}_1^{\overline{\alpha}_i}$. Now $\mathcal{S}$ can compute P_h's partial decryption as $w_h := W \cdot \prod_{P_i \in \mathcal{Q} \setminus \{P_h\}} w_i^{-1}$. Hence, P_h sends exactly what $\mathcal{Z}$ would expect given that ct decrypts to the leaked M. The corresponding proof can, again, be simulated.

Since the messages sent by P_h are distributed statistically close to what it would send in the real protocol and P_h was chosen randomly in the beginning, no environment corrupts it (causing the simulation to fail) with probability greater than $\frac{1}{2} + \mathsf{negl}(\sigma)$.

Guaranteed Output Delivery. Assuming an honest majority an adversary cannot prevent the successful completion of the protocol as the simulator $\mathcal{S}$ always instructs $\mathcal{F}_{\mathsf{TE}}$ to $\mathsf{continue}$ and deliver output to the honest parties. □

7.3 Pedersen Setup Protocol

In contrast to the statically secure protocol of Section 6 the protocol $\Pi_{\mathsf{TE}}^{\mathsf{adaptive}}$ requires additional setup, namely the generation of a Pedersen public key h_{P}. Ideally h_{P} would be almost-uniformly sampled from G^q, but this suffers the same difficulties as distributed key generation. However, as discussed in Section 3.4, Pedersen commitments retain their hiding and binding properties when $h_{\mathsf{P}} \in \widehat{G}^q$ is sampled from an adversarially-biased distribution. Therefore our ideal functionality $\mathcal{F}_{\mathsf{CL\text{-}Ped}}$ (Figure 9) can generate h_{P} and allow the adversary to induce this bias. The functionality outputs this biased Pedersen public key along with the usual public parameters $(\mathsf{pp}_{\mathsf{cl}}, h_{\mathsf{P}})$.

Theorem 13 states that the ideal functionality is realized via the protocol $\Pi_{\mathsf{CL\text{-}Ped}}$ (Figure 10). In this protocol each party P_i samples an ω_i and broadcasts $h_i = g_q^{\omega_i}$ as their contribution to h_{P}. The party P_i additionally proves knowledge of $\omega_i = \mathsf{log}_{g_q}(h_i)$ with a straight-line extractable zero-knowledge proof π_i. Then, for $\mathcal{Q}$ the set of parties that broadcast verifying proofs, the Pedersen base is computed as

$$h_{\mathsf{P}} := \prod_{P_i \in \mathcal{Q}} h_i.$$

Figure 9: Extended Setup Functionality $\mathcal{F}_{\mathsf{CL\text{-}Ped}}$

On input $(\mathsf{Gen}, 1^\lambda, q)$ from all parties, $\mathcal{F}_{\mathsf{CL\text{-}Ped}}$ does:

1. Run the CL setup algorithm: $\mathsf{pp}_{\mathsf{cl}} = (q, \bar{s}, f, g_q, \widehat{G}, F; \rho) \leftarrow \mathsf{CLGen}(1^\lambda, q)$ and send $\mathsf{pp}_{\mathsf{cl}}$ to $\mathcal{S}$.
2. Samples $\omega \leftarrow \mathcal{D}_q$, and set $h_{\mathsf{P}}^* := g_q^\omega$.
3. Send h_{P}^* to $\mathcal{S}$ and receive $\delta \in \mathbb{Z}$ from $\mathcal{S}$.
4. Set $h_{\mathsf{P}} := h_{\mathsf{P}}^* \cdot g_q^\delta = g_q^{\omega+\delta}$.
5. Record and output $(\mathsf{pp}_{\mathsf{cl}}, h_{\mathsf{P}})$ to all parties.

Figure 10: Protocol $\Pi_{\mathsf{CL\text{-}Ped}}$

On input $(\mathsf{Gen}, 1^\lambda, q)$ do:

1. Send $(\mathsf{Gen}, 1^\lambda, q)$ to $\mathcal{F}_{\mathsf{CL}}$ which returns $\mathsf{pp}_{\mathsf{cl}} = (q, \bar{s}, f, g_q, \widehat{G}, F; \rho)$.
2. Generation of a Pedersen public key h_{P}.
   All P_i proceed in parallel:
   (a) Sample the contribution $\omega_i \leftarrow \mathcal{D}_q$ and set $h_i := g_q^{\omega_i}$.
   (b) Compute $\pi_i \leftarrow \Pi_{\mathsf{DLog}}^{\mathsf{PoK,sle}}.\mathsf{Prove}(h_i, \omega_i)$.
   (c) Broadcast (h_i, π_i).
   (d) Let $\mathcal{Q} := \{P_i \mid \Pi^{\mathsf{PoK,sle}}.\mathsf{Verify}(h_i, \pi_i) = \top\}$.
   (e) Set $h_{\mathsf{P}} := \prod_{P_i \in \mathcal{Q}} h_i$.
   (f) Output $(\mathsf{pp}_{\mathsf{cl}}, h_{\mathsf{P}})$.

Theorem 13 (Security of $\Pi_{\mathsf{CL\text{-}Ped}}$). *The protocol $\Pi_{\mathsf{CL\text{-}Ped}}$ securely realizes the functionality $\mathcal{F}_{\mathsf{CL\text{-}Ped}}$ in the UC SIP model, in presence of active and adaptive corruptions of up to $t \leq N-1$ parties, given that $\Pi_{\mathsf{DLog}}^{\mathsf{PoK,sle}}$ is a straight-line extractable zero-knowledge proof of knowledge for the relation $\mathsf{R}_{\mathsf{DLog}}$.*

Proof. We give the simulator in Figure 11.

Correctness of the Simulation. Let P_h be the single inconsistent party. By the following equation the protocol output h_{P} matches the output of the ideal functionality. $\mathcal{F}_{\mathsf{CL\text{-}Ped}}$:

$$\begin{aligned} h_{\mathsf{P}} := \prod_{P_i \in \mathcal{Q}} h_i &= h_h' \cdot \left(\prod_{P_i \in \mathcal{Q} \setminus \{P_h\}} h_i \right) \\ &= \left(h_{\mathsf{P}}^* \cdot \prod_{P_i \neq P_h} g_q^{-\omega_i'} \right) \cdot \left(\prod_{P_i \in \mathcal{Q} \setminus \{P_h\}} g_q^{\omega_i} \right) \cdot \underbrace{\left(\prod_{P_i \notin \mathcal{Q}} g_q^{\omega_i} \right)}_{=1} \\ &= h_{\mathsf{P}}^* \cdot g_q^{\sum_{P_i \neq P_h} (\omega_i' - \omega_i)} = h_{\mathsf{P}}^* \cdot g_q^\delta \end{aligned}$$

Indistinguishability of the Simulation. All parties except P_h follow the protocol honestly and therefore their messages have the same distribution as in a real protocol execution. The message (h_h, π_h) sent by P_h is statistically indistinguishable from the real protocol. The first component, h_h, is distributed identically conditioned on the output h_{P}. The second component, the simulated proof π_h, is statistically indistinguishable from a real proof. □

8 Evaluation

We implemented our statically secure secure distributed key generation and decryption protocols of Section 6 using the BICYCL library [Bou+23] for class group arithmetic and evaluated it on a M1 Macbook Pro using 8 cores. Table 1 reports the computation time for $\Pi_{\mathsf{TE}}^{\mathsf{static}}$ of a single party during KeyGen (P_N),

Figure 11: Simulator for $\Pi_{\mathsf{CL\text{-}Ped}}$

Let P_h be the (randomly chosen and thus far uncorrupted) inconsistent party. If $\mathcal{Z}$ corrupts some $P_i \in \mathcal{H} \setminus \{P_h\}$ then $\mathcal{S}$ reveals the state of P_i terminates the simulation of P_i and removes P_i from $\mathcal{H}$. If $\mathcal{Z}$ corrupts P_h then the simulation fails.

On input $(\mathsf{Gen}, 1^\lambda, q)$:

1. Receive $\mathsf{pp}_{\mathsf{cl}}$ from $\mathcal{F}_{\mathsf{CL\text{-}Ped}}$. Simulate call to $\mathcal{F}_{\mathsf{CL}}$.
2. Simulate generation of h_{P}:
   (a) Receive h_{P}^* from $\mathcal{F}_{\mathsf{CL\text{-}Ped}}$.
   (b) For every $P_i \neq P_h$: sample $\omega_i' \leftarrow \mathcal{D}_q$, set $h_i' := g_q^{\omega_i'}$, compute $\pi_i' \leftarrow \Pi_{\mathsf{DLog}}^{\mathsf{PoK,sle}}.\mathsf{Prove}((g_q, h_i'), \omega_i')$.
   (c) For P_h: instead, set
   $$h_h' := h_{\mathsf{P}}^* \cdot \Big(\prod_{i \neq h} h_i'\Big)^{-1}$$
   and simulate $\pi_h' \leftarrow \Pi_{\mathsf{DLog}}^{\mathsf{PoK,sle}}.\mathsf{SimProve}((g_q, h_h'))$.
   (d) For each P_i:
      i. If P_i is still honest: simulate the broadcast of $(h_i, \pi_i) := (h_i', \pi_i')$.
      ii. If P_i is corrupt: receive (h_i, π_i).
   (e) Let $\mathcal{Q} := \{P_i \mid \Pi_{\mathsf{DLog}}^{\mathsf{PoK,sle}}.\mathsf{Verify}(h_i, \pi_i) = \top\}$. (Note: if $P_i \in \mathcal{H}$ then $P_i \in \mathcal{Q}$.)
   (f) For each $P_i \in \mathcal{Q} \setminus \{P_h\}$: compute $\omega_i := \Pi_{\mathsf{DLog}}^{\mathsf{PoK,sle}}.\mathsf{Extract}(h_i, \pi_i)$. (Note: if $P_i \in \mathcal{H}$ then $\omega_i = \omega_i'$.) For each $P_j \notin \mathcal{Q}$: set $\omega_j := 0$.
   (g) Send δ to $\mathcal{F}_{\mathsf{CL\text{-}Ped}}$, for
   $$\delta := \sum_{P_i \neq P_h} (\omega_i - \omega_i') \in \mathbb{Z}.$$

assuming that there are no complaints, and Decrypt (P_t), assuming a set of $t+1$ contributing parties, as in $\Pi_{\mathsf{TE}}^{\mathsf{static}}$. We split computation costs for the KeyGen protocol into the Dealing, Check, and Extract phases and the Decrypt protocol into Partial Decrypt, Verify, and Combine steps.

1. KeyGen. Dealing Phase (Step 1 of $\Pi_{\mathsf{TE}}^{\mathsf{static}}$): The commitments $C_{i,0}, \ldots, C_{i,t}$ are generated in parallel during F-Share and the batching method of Section 4.2 is used for $\Pi_{\mathsf{DLog}}^{\mathsf{ZK}}$.
2. KeyGen. Check Phase (Step 2a of $\Pi_{\mathsf{TE}}^{\mathsf{static}}$): The $N-1$ shares dealt to each $P_j \neq P_i$ are verified in parallel. The computation of (6) in F-Check uses Horner's method "in the exponent."
3. KeyGen. Extract Phase (Step 3 of $\Pi_{\mathsf{TE}}^{\mathsf{static}}$): The N party public keys Γ_j are computed in parallel (again, with Horner "in the exponent").
4. Decrypt. Partial Decryption (Step 1 of $\Pi_{\mathsf{TE}}^{\mathsf{static}}$): Computation of the partial decryption w_i and the commitment of $\Pi_{\mathsf{EqDLog}}^{\mathsf{ZK}}$ involve 3 exponentiations that are done in parallel.
5. Decrypt. Verify (Step 2 of $\Pi_{\mathsf{TE}}^{\mathsf{static}}$): The small exponents test ([BGR98]) is used for batch verification as in [Fri+23]. Note that this does not allow identification of the cheating player. If verification fails one has to verify all of the $N-1$ proofs to determine the set S. The two verification equations involve 2 exponentiations and 4 multiexponentiations which are done in parallel.
6. Decrypt. Combine (Steps 3 and 4 of $\Pi_{\mathsf{TE}}^{\mathsf{static}}$):

Table 2 reports the communication costs for both of our protocols.

Comparison Since the Paillier cryptosystem [Pai99] is the most commonly used linearly homomorphic encryption scheme, we report timings of Tiresias [Fri+23], a threshold version of Paillier[19], for comparison. Their library implements decryption with a 2048 bit modulus only (for $\lambda = 112$ bit security), so we can neither compare KeyGen nor Decrypt at other security levels. Additionally, Tiresias Decrypt was measured with Steps 5 and 6 combined, as reflected in Table 1. Threshold decryption of a ciphertext involving $t+1 = N/2$ parties with our scheme instead of Tiresias is about 6.5× faster with 7× and 5.5× less communication for $N = 10, 100$ and about 2.5× faster with 2× less communication for $N = 1000$.

[19] https://github.com/odsy-network/tiresias

The key generation for Tiresias involves the multiparty generation of an RSA modulus with the Diogenes protocol [Che+21]. This paper reports running times (including communication) of 445 s for 10 parties, 486 s for 100 parties, 1306 s for 1000 parties (with malicious security). These are much higher than our measurements for KeyGen (albeit we have only included the computation cost). This is to be expected however since the generation of an RSA modulus is a more complex operation than generating a key pair for a DLog-based cryptosystem.

Recently, [WMC24] also published a threshold version of HSM-CL. There is no public implementation available, so we take the numbers from their paper. For KeyGen with $N = 10$ and $t = N - 1$, they report a computation time of 42 s and ≈ 35 kB of communication. For Decrypt they report 1797 ms and 0.8 kB. These numbers are significantly higher than ours.

Table 1. Running times in ms for a single party with $\lambda = \sigma = 112$ and $t = N/2 - 1$ for this work and Tiresias [Fri+23]. The numbers for [WMC24] are taken from their paper with $\lambda = 128$ and $t = N - 1$.

N	10	100	1000	10	100	1000	10
KeyGen	$\Pi_{\mathsf{TE}}^{\mathsf{static}}$						[WMC24]
Dealing	24	241	17 325				
Check	63	3 609	351 540				
Extract	7	467	46 338				
total	93	4 318	415 203				42 000
Decrypt	$\Pi_{\mathsf{TE}}^{\mathsf{static}}$			Tiresias [Fri+23]			[WMC24]
Partial Decrypt	13	32	352	122	153	639	
Verify	14	28	253	67	609	18 661	
Combine	2	59	6 922				
total	29	119	7 527	189	762	19 300	1 797

Table 2. Communication costs per party in KiB with $\lambda = \sigma = 112$ and $t = N/2 - 1$. For our protocol, we distinguish between the communication via peer-to-peer channels (P2P) and via the broadcast channel. The numbers for [WMC24] are taken from their paper with $\lambda = 128$ and $t = N - 1$.

	KeyGen (P2P / Broadcast)						Decrypt (Broadcast)		
N	10		100		1000		10	100	1000
$\Pi_{\mathsf{TE}}^{\mathsf{static}}$	1.0	1.0	21.3	9.1	1 758.6	90.8	0.6	0.7	2.3
$\Pi_{\mathsf{TE}}^{\mathsf{adaptive}}$	3.1	38.4	53.5	47.1	3 627.0	135.5	0.6	0.6	0.6
Tiresias [Fri+23]							2.1	2.2	4.0
[WMC24]	35.5						0.8		

Acknowledgements This research was supported by the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC), the Danish Independent Research Council under Grant-ID DFF-3103-00077B (CryptoDigi), the French ANR Project ANR-21-CE39-0006 SANGRIA, the France 2030 ANR Project ANR-22-PECY-003 SecureCompute, and the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0085. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA). Distribution Statement "A" (Approved for Public Release, Distribution Unlimited). Kelsey Melissaris was supported by a Protocol Labs Postdoctoral Fellowship.

References

[AF04] Masayuki Abe and Serge Fehr. *Adaptively Secure Feldman VSS and Applications to Universally-Composable Threshold Cryptography*. In: *Advances in Cryptology – CRYPTO 2004*. Aug. 2004. DOI: 10.1007/978-3-540-28628-8_20.

[BC24] Claudia Bartoli and Ignacio Cascudo. *On Sigma Protocols and (packed) Black-Box Secret Sharing Schemes*. In: *PKC 2024: 27th International Conference on Theory and Practice of Public Key Cryptography, Part II*. Apr. 2024. DOI: 10.1007/978-3-031-57722-2_14.

[BDO23] Lennart Braun, Ivan Damgård, and Claudio Orlandi. *Secure Multiparty Computation from Threshold Encryption Based on Class Groups*. In: *Advances in Cryptology – CRYPTO 2023, Part I*. Aug. 2023. DOI: 10.1007/978-3-031-38557-5_20.

[Bou+23] Cyril Bouvier, Guilhem Castagnos, Laurent Imbert, and Fabien Laguillaumie. *I Want to Ride My BICYCL : BICYCL Implements CryptographY in CLass Groups*. In: *Journal of Cryptology* 3 (July 2023). DOI: 10.1007/s00145-023-09459-1.

[Can+99] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. *Adaptive Security for Threshold Cryptosystems*. In: *Advances in Cryptology – CRYPTO'99*. Aug. 1999. DOI: 10.1007/3-540-48405-1_7.

[Can01] Ran Canetti. *Universally Composable Security: A New Paradigm for Cryptographic Protocols*. In: *42nd Annual Symposium on Foundations of Computer Science*. Oct. 2001. DOI: 10.1109/SFCS.2001.959888.

[Che+21] Megan Chen, Carmit Hazay, Yuval Ishai, Yuriy Kashnikov, Daniele Micciancio, Tarik Riviere, abhi shelat, Muthu Venkitasubramaniam, and Ruihan Wang. *Diogenes: Lightweight Scalable RSA Modulus Generation with a Dishonest Majority*. In: *2021 IEEE Symposium on Security and Privacy*. May 2021. DOI: 10.1109/SP40001.2021.00025.

[CL15] Guilhem Castagnos and Fabien Laguillaumie. *Linearly Homomorphic Encryption from* DDH. In: *Topics in Cryptology – CT-RSA 2015*. Apr. 2015. DOI: 10.1007/978-3-319-16715-2_26.

[CLT18] Guilhem Castagnos, Fabien Laguillaumie, and Ida Tucker. *Practical Fully Secure Unrestricted Inner Product Functional Encryption Modulo p*. In: *Advances in Cryptology – ASIACRYPT 2018, Part II*. Dec. 2018. DOI: 10.1007/978-3-030-03329-3_25.

[CLT22] Guilhem Castagnos, Fabien Laguillaumie, and Ida Tucker. *Threshold Linearly Homomorphic Encryption on* $\mathbf{Z}/2^k\mathbf{Z}$. In: *Advances in Cryptology – ASIACRYPT 2022, Part II*. Dec. 2022. DOI: 10.1007/978-3-031-22966-4_4.

[Cou+21] Geoffroy Couteau, Michael Klooß, Huang Lin, and Michael Reichle. *Efficient Range Proofs with Transparent Setup from Bounded Integer Commitments*. In: *Advances in Cryptology – EUROCRYPT 2021, Part III*. Oct. 2021. DOI: 10.1007/978-3-030-77883-5_9.

[DF02] Ivan Damgård and Eiichiro Fujisaki. *A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order*. In: *Advances in Cryptology – ASIACRYPT 2002*. Dec. 2002. DOI: 10.1007/3-540-36178-2_8.

[DN03] Ivan Damgård and Jesper Buus Nielsen. *Universally Composable Efficient Multiparty Computation from Threshold Homomorphic Encryption*. In: *Advances in Cryptology – CRYPTO 2003*. Aug. 2003. DOI: 10.1007/978-3-540-45146-4_15.

[Fis05] Marc Fischlin. *Communication-Efficient Non-interactive Proofs of Knowledge with Online Extractors*. In: *Advances in Cryptology – CRYPTO 2005*. Aug. 2005. DOI: 10.1007/11535218_10.

[FMY99] Yair Frankel, Philip D. MacKenzie, and Moti Yung. *Adaptively-Secure Optimal-Resilience Proactive RSA*. In: *Advances in Cryptology – ASIACRYPT'99*. Nov. 1999. DOI: 10.1007/978-3-540-48000-6_15.

[Fri+23] Offir Friedman, Avichai Marmor, Dolev Mutzari, Yehonatan C. Scaly, Yuval Spiizer, and Avishay Yanai. *Tiresias: Large Scale, Maliciously Secure Threshold Paillier*. Cryptology ePrint Archive, Report 2023/998. https://eprint.iacr.org/2023/998. 2023.

[FS87] Amos Fiat and Adi Shamir. *How to Prove Yourself: Practical Solutions to Identification and Signature Problems*. In: *Advances in Cryptology – CRYPTO'86*. Aug. 1987. DOI: 10.1007/3-540-47721-7_12.

[Gen+04] Rosario Gennaro, Darren Leigh, R. Sundaram, and William S. Yerazunis. *Batching Schnorr Identification Scheme with Applications to Privacy-Preserving Authorization and Low-Bandwidth*

Communication Devices. In: *Advances in Cryptology – ASIACRYPT 2004*. Dec. 2004. DOI: 10.1007/978-3-540-30539-2_20.

[Gen+07] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. *Secure Distributed Key Generation for Discrete-Log Based Cryptosystems*. In: *Journal of Cryptology* 1 (Jan. 2007). DOI: 10.1007/s00145-006-0347-3.

[JL00] Stanislaw Jarecki and Anna Lysyanskaya. *Adaptively Secure Threshold Cryptography: Introducing Concurrency, Removing Erasures*. In: *Advances in Cryptology – EUROCRYPT 2000*. May 2000. DOI: 10.1007/3-540-45539-6_16.

[LP01] Anna Lysyanskaya and Chris Peikert. *Adaptive Security in the Threshold Setting: From Cryptosystems to Signature Schemes*. In: *Advances in Cryptology – ASIACRYPT 2001*. Dec. 2001. DOI: 10.1007/3-540-45682-1_20.

[Pai99] Pascal Paillier. *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*. In: *Advances in Cryptology – EUROCRYPT'99*. May 1999. DOI: 10.1007/3-540-48910-X_16.

[Ped92] Torben P. Pedersen. *Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing*. In: *Advances in Cryptology – CRYPTO'91*. Aug. 1992. DOI: 10.1007/3-540-46766-1_9.

[WMC24] Harry W. H. Wong, Jack P. K. Ma, and Sherman S. M. Chow. *Secure Multiparty Computation of Threshold Signatures Made More Efficient*. In: *ISOC Network and Distributed System Security Symposium – NDSS 2024*. Feb. 2024.