# Secure Multiparty Computation of Threshold Signatures Made More Efficient

Harry W. H. Wong
Department of Information Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong

Jack P. K. Ma
Department of Information Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong

Sherman S. M. Chow*
Department of Information Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong

***Abstract*—Threshold signatures, notably ECDSA, are fundamental for securing decentralized applications. Their non-linear structure poses challenges in distributed signing, often tackled by pairwise multiplicative-to-additive share conversion, leading to $O(n)$ communication and $O(n^2)$ verification costs for each of $n$ signers. Moreover, most schemes lack robustness, necessitating a complete restart upon fault. A pioneering work by Wong et al. (NDSS '23) still requires rolling back to the preceding round to resume signing after another round to convince all other signers.**

**We revisit secure multiparty computation from threshold linearly homomorphic encryption (LHE). Realizing its public verifiability and fault recovery, we encompass two technical contributions to Castagnos–Laguillaumie LHE (CT-RSA '15): a 2-round robust distributed key generation (DKG) protocol in the dishonest majority setting and an accompanying zero-knowledge proof allowing extraction in an unknown-order group. We extend the DKG with dual-code-based verification (ACNS '17), upgrading its $O(tn^2)$-cost private verifiability to an $O(n^2)$ public one.**

**Built on our DKG, we present the first threshold ECDSA protocol with $O(1)$ communication and $O(n)$ verification per-party costs while matching the lowest round complexity of non-robust schemes (CCS '20). Empirically, we halve the computation and communication costs of the signing phase compared to state-of-the-art robust threshold ECDSA (NDSS '23). We also illustrate the versatility of our techniques with an improved threshold extension (IEEE S&P '23) of BBS+ signatures (IEEE Syst. J. '13).**

## I. Introduction

Securing multifaceted critical services over the Internet and ensuring trustworthy information dissemination is paramount. A key concern is establishing trust. Infrastructure, including public-key infrastructure and DNS servers, crucially relies on the information authenticity and integrity. Any compromises expose users to risks like phishing, exploiting misplaced trust.

Mitigating such risks could use help from decentralization techniques, which eliminate a single point of compromise. Recently, there has been a surge of interest in decentralized services, such as decentralized social networks, driven by the demand for higher availability and resiliency. The growing popularity of blockchain has further fueled this trend.

Cryptography plays a pivotal role in achieving these goals. We see a wave of advancements in thresholdizing cryptosystems, *e.g.*, ECDSA, EdDSA [47], and the recently thresholdized BBS+ signature [26]. (Threshold) ECDSA finds applications in many domains, *e.g.*, DNS [24] and cryptocurrency [43], while BBS+ signature [4] serves as a crucial primitive in many anonymous credential designs [20], [42]. $(t, n)$-threshold signatures allow any size-$t$ subset of $n$ participants to collaboratively create a signature.

### A. Shortcomings of Existing Threshold ECDSA

Existing schemes suffer from several significant drawbacks.

(I) Roll Call Necessity: Many schemes [11], [15], [16], [28], [31], [41] need an often unaccounted communication round for a roll call [8] to form a fixed group of $t$ participating signers, which needs to stay the same during the entire signing process from initiation, hampering decentralized applications where parties might dynamically join or suddenly go offline.

(II) Non-robust Threshold Signing: Some schemes [11], [16] prioritize best-case performance (in communication complexity and rounds) by using $t$-out-of-$t$ (additive) secret sharing for minimal-participant signing. If any participant is faulty or absent, their contribution irrecoverably lost, rendering previous computations useless. This results in a critical denial-of-service vulnerability, especially in time-sensitive missions [48].

(III) Lack of Public Verifiability or Fault Recovery: Some schemes focus on private verifiability. To exclude a faulty participant from subsequent protocol steps, the identifying party is responsible for generating and sending evidence to others [16], requiring an additional round of communication. Public verifiability, even if supported, is rarely accompanied by fault recovery and needs a restart for any minor fault.

(IV) Inefficiency: Prior schemes commonly exhibit inefficient signing due to two-party interactions between each participants pair. A popular approach of using multiplicative-to-additive (MtA) protocol [11], [16], [35], [48] requires computation and communication of $\mathcal{O}(t)$ messages for each party. For fault identification, participants can show the correctness of

*Sherman Chow (corresponding author) is supported by the General Research Fund (CUHK 14210621) from Research Grant Council, Hong Kong. The authors acknowledge the anonymous reviewers for their valuable input. Special appreciation is expressed to Rosario Gennaro for providing insightful comments on a related PhD thesis.

Network and Distributed System Security (NDSS) Symposium 2024
26 February - 1 March 2024, San Diego, CA, USA
ISBN 1-891562-93-2
https://dx.doi.org/10.14722/ndss.2024.24601
www.ndss-symposium.org

their broadcast messages [11]. Nevertheless, each party needs to verify $\mathcal{O}(t^2)$ messages from $\mathcal{O}(t)$ signers to ensure correct computation, contributing to the quadratic verification cost.

We are intrigued to ask: *Can we build threshold signature schemes (ECDSA, BBS+, and possibly more) featuring*

1) *security against a dishonest majority,*
2) *materialization of the final signature whenever more than $t$ honest parties are present (even if they initially abstained),*
3) *constant communication overhead, and the least number of rounds as the state of the art, without roll call?*

### B. Homomorphic Computation of Threshold ECDSA

We depart from the trend [8], [11], [15], [16], [28], [31], [41], [48] and revisit the Cramer–Damgård–Nielsen (CDN) paradigm [22] of secure multi-party computation (SMC) from threshold linearly homomorphic encryption ((T)LHE). TLHE enables two functionalities: i) homomorphic operations allow anyone to operate on encrypted messages without decryption; and ii) threshold decryption can be done by any $t$ participants. In the context of threshold signature, participants can encrypt signature components and compute the final signature homomorphically without revealing the underlying values.

When instantiated appropriately with "robust" building blocks allowing public verification (a technical part of our contributions), this approach offers a strong robustness guarantee, naturally addressing the identified shortcomings.[1] Specifically, even in the worst-case scenario with fewer than $t$ participants remaining, the signing process just pauses. Once a sufficient number of participants rejoin, regardless of their past absence, any intermediate values or the final signature can be recovered by decrypting broadcasted and verifiably encrypted results. In short, throughout various phases of signature generation, they can seamlessly resume any previously disrupted signing.

"Reviving" this idea *efficiently (in time and space)*, with new *robust* machinery (while still maintaining efficiency), requires dedicated efforts. To illustrate, we provide an overview of our homomorphic signing proceeds for deriving an ECDSA signature $(R, \sigma)$, where each party holds a threshold decryption key of TLHE and a share of the ECDSA signing key $x$.

1) Distributively generate randomness $k$ in ciphertext;
2) Distributively generate $g^\gamma$ for a random $\gamma$, and $(k\gamma, kx)$ in ciphertext via homomorphic evaluation on encrypted $k$;
3) Threshold-decrypt $k\gamma, g^\gamma$ and compute $R{:=}(g^\gamma)^{1/k\gamma}$;
4) Threshold-decrypt $\sigma := km + rkx$, which is homomorphically generated from encrypted $(k, kx)$ and plain $(r, m)$.

Each party $\mathsf{P}_i$ only takes $\mathcal{O}(1)$ messages and 3 rounds to compute $k\gamma$ (and similarly $kx$) instead of $\mathcal{O}(t)$ for MtA:

1) Broadcast the encrypted $k_i$;
2) Compute (homomorphically) the encrypted $k := \sum_i k_i$ and then encrypted $\gamma_i k$, broadcast the latter;
3) Compute (homomorphically) the encrypted $k\gamma := \sum_i \gamma_i k$, then threshold-decrypt the encrypted $k\gamma$.

Applying zero-knowledge proof (ZKP) over the computation also only takes $\mathcal{O}(1)$ message (*cf.*, $\mathcal{O}(t)$ for MtA). *Public* verifiability allows immediate and non-interactive fault attribution (with no extra round of ZKP of secrets needed by MtA). Fault attribution is "for free," asymptotically.

While enjoying all advantages (no roll call, robust, publicly verifiable, and computationally efficient), our scheme matches the least number of rounds [11] among prior threshold ECDSA schemes with identifiable abort or robustness, as in Table I. Despite the higher computational cost of class-group computations compared to RSA modulus operations, we substantiate the efficiency of our approach through experimental results. Communication cost refers to the number of messages each party sends/broadcasts.[2] The '$+$' part indicates the extra cost for fault identification. For per-party computational cost, we distinguish between the cost of computing the signature and verifying intermediate values, which may only needed for fault identification.[3] The cost of the SMC-based scheme [35] is provided as a lower bound due to unspecified MtA instantiation.

This approach relies on two LHE extensions: i) distributed key generation (DKG) for LHE and ii) ZKP of knowledge for public verifiability that allows witness extraction, which is crucial for security proof in the *dishonest majority setting* expecting the weakest possible assumption about colluding parties. We build upon Castagnos–Laguillaumie (CL) encryption [17] for its decentralized setup and its message space (a group $\mathbb{Z}_q$ of prime order $q$) aligned with the modulus used by ECDSA, avoiding additional ZKP needed if other LHE was used (*e.g.*, range proof with threshold Paillier encryption [25], [29], which operates on integers modulo an RSA modulus [16], [19]). The simplification of ZKP is not just a matter of efficiency but also security since its complexity makes it error-prone (several bugs have been found in previous protocols). However, while benefiting from the prime order of the message space, the class group itself operates with an unknown order. Apart from the more expensive atomic operation (than Paillier), this hinders the design space since division in exponent is not feasible. Witness extraction also becomes tricky due to the same reason.

### C. Extending Distributed Key Generation to the Class Groups

We propose two DKG protocols, one for CL encryption [17] and one for discrete-logarithm-based cryptosystems.

*1) Outline of Our DKG:* Each party sets up its own CL encryption key pair to aid the distributed generation of key pairs for either CL encryption or ECDSA. They first distribute CL-encrypted shares (and committed shares) with ZKP. After verifying all shares, excluding invalid ones, each party broadcasts their share in exponent and uses ZKP to show decryption of the (homomorphically-evaluated) ciphertext returns the correct share. The public key (as shared secret in exponent) is uniformly distributed, and can be constructed from the broadcasted share in exponent without further interaction.

[1] Alternatively, one could use a universal thresholdizer [6], but it requires threshold fully homomorphic encryption, which is heavyweight. Prior threshold ECDSA schemes often use LHE for specific tasks, *e.g.*, share preparation. Non-threshold "microscopic" usages of LHE do not readily enjoy the benefits.

[2] Public verification requires all messages to be broadcasted, so we assume point-to-point messages are encrypted and broadcasted.

[3] Verification is done with respect to received messages, making the verification cost at least the communication cost multiplied by $\mathcal{O}(n)$, except for the Paillier-based construction, which requires verifier-specific proof. $\mathcal{O}(t)$ and $\mathcal{O}(n)$ are of the same order and asymptotically interchangeable. Our $\mathcal{O}(n^2)$ dual-code-based verification is in distributed key generation and is not shown.

TABLE I: Comparison of $(n, n)$ or $(t, n)$ Threshold ECDSA Schemes with Identifiable Abort or Robustness

| Schemes | Communication (Rounds) | | Protection | Comm. Cost | Comp. Cost | Threshold Type |
|---|---|---|---|---|---|---|
| Paillier-based [11] | Offline: $3+1$ | Online: $1+1$ | Identifiable Abort | $\mathcal{O}(n)+\mathcal{O}(n^2)$ | $\mathcal{O}(n)+\mathcal{O}(n^2)$ | Fixed size-$n$ group |
| | Offline: $6+1$ | Online: $1$ | Identifiable Abort | $\mathcal{O}(n)+\mathcal{O}(n)$ | $\mathcal{O}(n)+\mathcal{O}(n^2)$ | Fixed size-$n$ group |
| SMC-based [35] | Offline:10 | Online: 3 | Identifiable Abort<br>Robust for Online only | $\mathcal{O}(n)+\mathcal{O}(n^2)$ | $\mathcal{O}(n)+\mathcal{O}(n^2)$ | Fixed size-$n$ group for Offline<br>Any size-$\geq t$ subgroup for Online |
| CL-based [16] | Offline: $6+1$ | Online: $1$ | Identifiable Abort | $\mathcal{O}(n)+\mathcal{O}(n)$ | $\mathcal{O}(n)+\mathcal{O}(n^2)$ | Fixed size-$t$ group |
| CL-based [48] | Offline: $4+2$ | Online: $1$ | Identifiable + Robust | $\mathcal{O}(n)+\mathcal{O}(n)$ | $\mathcal{O}(n)+\mathcal{O}(tn^2)$ | Any size-$\geq t$ subgroup for Both |
| Ours | Offline: 3 | Online: 1 | Identifiable Abort<br>Identifiable + Robust | $\mathcal{O}(1)$ | $\mathcal{O}(n)+\mathcal{O}(n)$ | Any size-$\geq t$ group for Both |

Note that shares are distributed in exponent or encrypted form with ZKP, both of which are publicly verifiable. This avoids another round for resolving conflict. In short, our 2-round robust DKG protocol removes culprits non-interactively.

*2) Dual-Code-based Verification:* ZKP alone does not safeguard against the distribution of inconsistent shares to different parties, leading to inconsistency. The traditional remedy is generating a committed polynomial corresponding to the shares as a basis for share verification. However, verification requires the knowledge of shares, and verifying them w.r.t. the committed shares requires $\mathcal{O}(tn)$ computation (the $t$ factor comes from the evaluation of degree-$t$ polynomials). Thus, complaining requires an extra round when private verification does not pass.

In lieu of this, we extend dual-code-based verification [12], initially designed for public attestation of uniform randomness beacons, assuming an honest majority. We adapt it for our ECDSA usage. In essence, it leverages the existence of a dual vector such that its inner product with a vector of shares would be non-zero upon inconsistent shares. This allows public verification of all shares (in exponent) with $\mathcal{O}(n^2)$ computation.

We further extend this approach to commitments of shares. Our extension eliminates any information leakage during the first communication round, thwarting rushing adversaries (who observe submissions of others before providing their own contribution) from gaining any advantage or introducing bias by intentional dropout. Consequently, our protocol maintains its robustness even in the presence of faults, ensuring a uniform distribution of the key and eliminating any concerns related to non-uniform key distribution that requires a protocol abort.

### D. Recovery of Shares for Dishonest Majority via Extraction

A major hurdle in DKG under a dishonest majority lies in extracting shares of the secret key, usually needed by simulation-based security. In the simpler case where a simulator controls *an honest majority*, possessing at least $t$ shares allows easy recovery of the secret and then the shares held by other parties — an option unavailable to a dishonest majority. The beauty of our approach is that all shares are encrypted and broadcasted. The simulator can simply extract the shares by decryption, and the decryption key is the only element to be extracted through the ZKP of knowledge. This also enhances the extendibility to the concurrent setting.

However, the (share of) secret key in the class group of CL encryption resides in integers over an unknown modulus. Without help from any computational or model assumption, the standard strategies of using $\Sigma$-protocols in unknown order groups [5], [46] have been shown to have a $1/2$ soundness lower bound. Rather than idealized models like the generic group model (GGM) [49], we consider the 2-fractional-root assumption [21], a computational one taking care of challenges posed by division in the class group needed by $\Sigma$-protocols.

### E. Miscellaneous Improvements for Threshold CL Encryption

We further enrich our understanding and the performance of the threshold CL encryption scheme in a few aspects.

*1) Reduction in Exponentiation (Section V-A):* We streamline threshold CL encryption by reducing the number of exponentiations of the factorial $n!$ from 3 to 2 in decryption. This significantly improves the efficiency since we operate in an unknown-order group, dealing with large exponents with no modular reduction. The order of magnitude for $n!$ can go up to 128-bit (*e.g.*, when $n = 32$). In threshold ECDSA, decryption is often needed, including recovery and final signature output.

*2) Property for Simple Security Proof (Sections V-C and VI-B):* In complement to its universally composable (UC) security, we show the security of threshold CL encryption in the game-based definition for accessibility. Additionally, we derive a property from two fundamental security requirements of TLHE that greatly simplifies the security proof when used in the threshold signature scheme. By incorporating this property, the security argument in applying various encryption schemes (*e.g.*, threshold lifted ElGamal encryption and threshold Paillier encryption) is simplified and made more accessible.

### F. Assessing Related and Concurrent Works

*a) Threshold ECDSA:* Most proposals perform conceptually similar computations using MtA but with different optimizations for communication overhead. We refer to representative works [11], [16], [35], [48] for surveys of developments [38]. Abram *et al.* [2] propose a non-robust scheme with 1-round pre-signing via a pseudorandom correlation generator.

The protocol efficiency is mainly affected by the primitive to instantiate MtA. While constructions based on oblivious transfer (OT) [28] are communication-intensive, their computational efficiency is high for using symmetric-key primitives. Our approach prioritizes communication over computation, driven by the recognition that computation speed can often be enhanced through optimization (of class-group operations) or hardware advancements, whereas communication faces physical barriers such as network latency or instability.

As mentioned, most schemes assume a fixed roll call for signing, which requires a restart from scratch in case of any fault, even if it is detectable. Gągol *et al.* [35] achieve robust online signing under the assumption that the offline signature is computed faithfully by all $n$ parties. A notable exception is the recent scheme of Wong *et al.* [48], which achieves fault attribution against a dishonest majority and self-healing under an honest majority. However, in the worst case with faulty contributions spanning multiple rounds, it still takes two extra rounds. These are reported under "Threshold Type" in Table I.

Concurrently, Bouez and Singh [8] formulate the requirement of a fixed group as "with roll call." They achieve 1-round online signing with a 7-round offline pre-signing phase on top of a 1-round scheme of Gennaro and Goldfeder [32], but fault identification [32] is left as future work.

*b) Threshold BBS+:* Doerner *et al.* [26] adapt SMC techniques for threshold ECDSA [27], [28] to threshold BBS+ with selective abort. Using oblivious transfer to multiply two secrets, their scheme is computationally lighter than the LHE approach. In contrast, our threshold-signing protocol takes 4-round communication (optimizable to 3 with extra recovery computation) with $\mathcal{O}(1)$ messages per party. More importantly, ours is robust: (1) cheaters are identified and removed non-interactively, and (2) any party can help (even if they did not participate in the current protocol execution) to continue.

*c) Threshold LHE:* Castagnos *et al.* [19] proposed a threshold LHE scheme on $\mathbb{Z}_{2^k}$, integers modulo a power-of-2. They first proposed a homomorphic encryption scheme on $\mathbb{Z}_{2^k}$ and then applied linear integer secret sharing to share the decryption key on a bounded integer. Thanks to the ElGamal-like (linear) structure, the threshold scheme immediately has a 1-round threshold decryption. However, as shown by Cramer and Fehr [23], each party holds at least $\mathcal{O}(\log n)$ shares on average. Using integer secret sharing results in a heavy overhead under a malicious setting; each party at least proves $\mathcal{O}(\log n)$ elements for decryption correctness. In our work, we build robust DKG for threshold LHE over $\mathbb{Z}_q$ [10].

*d) Concurrent Work:* Braun *et al.* [10] revisit the CDN paradigm of SMC. Their work focuses on the "You-Only-Speak-Once" (YOSO) setting, which considers an honest majority. Thus, they do not consider witness extraction from ZKP under a dishonest majority, or adopt dual-code-based verification. They did use ZKP to ensure the recovery of the shared secret, which is unnecessary, as correct reconstruction over the public key space alone suffices for threshold decryption.

That said, their work suggests a very promising approach to general SMC with only a transparent setup, albeit it does not come with our new techniques specifically for pushing the frontier of threshold signatures or contributing a set of gadgets for more discrete-logarithm/class-group-based cryptosystems.

*e) DKG for Class Groups:* DKG of Braun *et al.* [10] yields a biased key without weakening security in the YOSO setting. They aim for statistical ciphertext indistinguishability using a lossy public key, taking two sequential executions of verifiable secret sharing. In contrast, our DKG yields a uniformly distributed public key, impervious to disruption by rushing adversaries. DKG for hidden-order groups, initiated by Rabin [44], remains rather limited. Castagnos *et al.* [19] did not propose any DKG but assumed a trusted dealer. Our DKG follows the paradigm of Gennaro *et al.* [34], also employed in the "real" threshold ECDSA scheme of Wong *et al.* [48].

*f) DKG for Prime-order Cyclic Groups:* Recent DKG developments pursue diverse objectives, *e.g.*, asynchronous DKG for both low-/high-threshold scenarios with security in the standard model [50] and batched asynchronous DKG [37]. The work by Groth [36] also incorporates share verification through publicly verifiable ZKP and stands out as a 1-round solution, but the public key it produces can be biased due to its 1-round nature [40]. Kate *et al.* [39] adhere to this ZKP structure but employ CL encryption for share distribution instead of ElGamal encryption. Alongside this ZKP, Cascudo and David [14] propose a new DKG paradigm, which is more efficient for using ZKP for dual-code-based verification. ALBATROSS [13] proposes compact ZKP for share validity, specialized for large batches of shared secrets. Atapoor *et al.* [3] suggest a post-quantum secure DKG. However, its ZKP only ensures share consistency and lacks public verifiability.

### G. Noteworthy Contributions

Summarizing, our contribution manifests at different levels.

(I) We propose a robust DKG for CL encryption under the dishonest majority setting, which generates a uniformly-distributed public key and extends the security of threshold CL encryption originally tailored for an honest majority [10]. Our public verification machinery enables a 2-round construction. We extend the dual-code-based verification [12] to class groups for our $(t, n)$-threshold setting, which reduces the $\mathcal{O}(tn^2)$ cost of the usual "pairwise" approach from one-to-one primitives.

(II) Our proposed ECDSA outperforms the state-of-the-art (see Table I) at a slight running-time overhead for (local) computation over CL encryption. Notably, it

- features robustness (parties can leave at any time),
- takes 3 rounds of communication (avoiding the extra communication round for conflict resolution/recovery by operating on (a different form of) publicly verifiable shares only),
- ensures $\mathcal{O}(1)$[4] communication overhead for verification of all other parties in threshold signing.

(III) We provide an alternative design and realization of threshold BBS+ [26]. The only existing attempt [28] mostly follows the blueprint of MtA-based ECDSA counterpart, subject to its relative weakness w.r.t. latest threshold ECDSA, *e.g.*, costly public verifiability and the lack of robustness.

(IV) Our technical contributions own independent interest and apply to other cryptosystems to be thresholdized with public verifiability, inheriting the compatibility of CL encryption, or general SMC tasks [10] in the dishonest majority setting.

## II. Preliminary

### A. Security and Communication Models

Let $\mathcal{P} = [1, n] = \{1, 2, \ldots, n\}$ be a party set. Each party $\mathsf{P}_i$ is indexed by $i \in \mathcal{P}$. Let $\mathsf{negl}(\lambda)$ be negligible functions in security parameter $\lambda$ and $\lambda_d$ be the parameter for statistical distance. A probabilistic polynomial-time

[4] More precisely, our ZKP involves integers (*i.e.*, *not* cryptographic group elements) of size $\mathcal{O}(n \log n)$ bits.

(PPT) algorithm $\mathsf{Alg}(\mathsf{in}) \to \mathsf{out}$, on input in, returns an output out. A protocol $\mathsf{Protocol}\langle \mathsf{P}_1(\mathsf{in}, \mathsf{in}_1); \ldots; \mathsf{P}_n(\mathsf{in}, \mathsf{in}_n)\rangle \to \langle(\mathsf{out}, \mathsf{out}_1); \ldots; (\mathsf{out}, \mathsf{out}_n)\rangle$, or $\mathsf{Protocol}\langle \mathsf{in}; \mathsf{in}_i\rangle_{i\in[1,n]} \to \langle \mathsf{out}; \mathsf{out}_i\rangle_{i\in[1,n]}$ for brevity. Each party $\mathsf{P}_i$ inputs a common input in and a private input $\mathsf{in}_i$. Protocol returns a common output out to all $n$ parties and a private output $\mathsf{out}_i$ to each $\mathsf{P}_i$.

The $(t, n)$ threshold access structure $\mathbb{A}$ is a collection of $\mathcal{P}' \subseteq \mathcal{P}$ such that $|\mathcal{P}'| \geq t$, denoted by $\mathcal{P}' \in \mathbb{A}$. We consider a PPT adversary that corrupts at most $t-1$ parties at the beginning (static corruption). Any corrupted party can deviate from the protocol specification at any time (malicious) and submit messages after viewing others (rushing). We use authenticated and synchronized broadcast channels, which hold irresponsive signers accountable and ensure bounded communication delay.

### B. (Multi-Party) Zero-Knowledge Proof

For a relation $(x, w) \in \mathcal{R}$, $x \in \mathcal{L}$ is a (public) statement for an NP language $\mathcal{L}$, and $w$ is the private witness. The prover with $w$ can use a zero-knowledge proof of knowledge (ZKPoK) to convince the verifier for $(x, w) \in \mathcal{R}$ without leaking anything beyond the statement is true. The $\Sigma_{\mathcal{R}}$-protocol can be used as a 3-move zero-knowledge proof of knowledge for the relation $\mathcal{R}$, which satisfies the security requirements: completeness, special soundness, and honest-verifier zero-knowledge (Appendix A-B). We will use the below two styles of proof systems derived from $\Sigma_{\mathcal{R}}$-protocol.

*a) Non-interactive zero-knowledge proof of knowledge (NIZK) $\mathsf{Z}_{\mathcal{R}}$ from the Fiat–Shamir transform of $\Sigma_{\mathcal{R}}$:*

- $\mathsf{P}(x; w) \to \pi$: On input of statement $x$ and the (secret) witness $w$, the probabilistic algorithm returns a proof $\pi$.
- $\mathsf{Vf}(\pi, x) \to 0/1$: On input the proof $\pi$ and the statement $x$, the deterministic algorithm returns a truth value $0/1$.

*b) Multi-party $\Sigma$-protocol $\Sigma^*_{\mathcal{R}}$:* We use a *multi-party* $\Sigma$-protocol $\Sigma^*_{\mathcal{R}}\langle\{x_j\}_{j\in\mathcal{P}}; w_i\rangle_{i\in\mathcal{P}}$, where each $\mathsf{P}_i$ can prove the knowledge of $w_i$ w.r.t. $x_i$ to all others in $\mathcal{P}$. It may not inherit the special soundness (allowing knowledge extraction) from the $\Sigma_{\mathcal{R}}$-protocol because simultaneous rewinding for multiple challenges in the multi-party setting is inefficient.[5] $\Sigma^*_{\mathcal{R}}$ [22] can be obtained by distributively generating a common random challenge (the second step of the $\Sigma$-protocol) and committing the first round message (the first step of the $\Sigma$-protocol) via an equivocable commitment.[6] This variant only requires changing the resultant challenge for knowledge extraction to avoid inefficient rewinding for multiple challenges.

### C. Class Groups and CL Encryption

Let $\mathbb{G} = \langle g\rangle$ be a group of prime order $q$ and $\mathbb{F}_q$ (or $\mathbb{Z}_q$) be a finite field with $q$ elements. The public elements in the group are denoted by uppercase letters $(X, \Gamma, K, \ldots)$, and their secret elements, the logarithm, are denoted by lowercase letters $(x := \log_g X, \gamma, k, \ldots)$. We denote the ring of polynomials in one variable $z$ with coefficients in $\mathbb{F}_q$ as $\mathbb{F}_q[z]$.

We outline some background and notation on class groups. On input of the security parameter $1^\lambda$ and two primes $q, q'$, the deterministic algorithm $\mathsf{Setup}_{\mathsf{CL}}$ [17], [18] outputs $\mathsf{pp}_{\mathsf{CL}}$, which defines an unknown-order class group, including $(\tilde{s}, \mathfrak{fg}_q, \mathfrak{f}, \mathfrak{g}_q, \hat{\mathfrak{G}}, \mathfrak{G}, \mathfrak{F}, \mathfrak{G}^q)$ such that $\hat{\mathfrak{G}} \supset \mathfrak{G} = \mathfrak{F} \times \mathfrak{G}^q$. Let $\hat{\mathfrak{G}}$ be the finite abelian subgroup of squares of the class group[7] $C(\Delta_q)$ with order $q\hat{s}$ and $\gcd(q, \hat{s}) = 1$, but only the upper bound $\tilde{s}$ of $\hat{s}$ is given. From $\hat{\mathfrak{G}}$, three subgroups are defined by the group-order-generator tuples: $(\mathfrak{G}, qs, \mathfrak{fg}_q)$, $(\mathfrak{F}, q, \mathfrak{f})$, and $(\mathfrak{G}^q, s, \mathfrak{g}_q)$. $\mathfrak{G}$ is a subgroup of $\hat{\mathfrak{G}}$ (implies that $s$ divides $\hat{s}$). $\mathfrak{F}$ is a subgroup with a deterministic *polynomial-time* algorithm $\mathsf{Dlog}(\cdot)$ that returns $x$ on input $\mathfrak{f}^x \in \mathfrak{F}$. $\mathfrak{G}^q$ is the subgroup containing all $q$-th powers in $\mathfrak{G}$. In practice, $[0, B = 2^{\lambda_d} q\tilde{s}]$ is treated as $\mathcal{D}_q$ for sampling elements in $\mathbb{Z}_s$ uniformly.

*a) CL Encryption ($\mathsf{CLE}$):* $\mathsf{CLE}$ is for $\mathbb{F}_q$. It is indistinguishable under chosen plaintext attacks (IND-CPA-secure) under the hard subgroup membership assumption [17].

- $\mathsf{Setup}(q, q', 1^\lambda) \to \mathsf{pp}$: Output $\mathsf{pp}_{\mathsf{CL}} \leftarrow \mathsf{Setup}_{\mathsf{CL}}(q, q', 1^\lambda)$.
- $\mathsf{KGen}(\mathsf{pp}) \to (\mathsf{ek}, \mathsf{dk})$: Output $\mathsf{dk} \leftarrow\$ \mathcal{D}_q$ and $\mathsf{ek} := \mathfrak{g}_q^{\mathsf{dk}}$.
- $\mathsf{Enc}(\mathsf{ek}, m; r) \to c_m$: Pick $r \leftarrow\$ \mathcal{D}_q$, output $(\mathfrak{g}_q^r, \mathfrak{f}^m \mathsf{ek}^r)$.
- $\mathsf{Dec}(\mathsf{dk}, c_m = (c_0, c_1)) \to m/\perp$: Output $\mathsf{Dlog}(c_1/c_0^{\mathsf{dk}})$.
- $\mathsf{Eval}(c_m, c_{m'}, +) \to c_{m+m'}$: Parse $c_m$ as $(c_{m,0}, c_{m,1})$ and $c_{m'}$ as $(c_{m',0}, c_{m',1})$, output $c_{m+m'} := (c_{m,0} \cdot c_{m',0}, c_{m,1} \cdot c_{m',1})$. We denote $\mathsf{Eval}(c_m, c_{m'}, +)$ as $c_m \boxplus c_{m'}$.
- $\mathsf{Eval}(a, c_m, \cdot) \to c_{am}$: Parse $c_m$ as $(c_{m,0}, c_{m,1})$, output $c_{am} := (c_{m,0}^a, c_{m,1}^a)$. We denote $\mathsf{Eval}(a, c_m, \cdot)$ as $a \boxdot c_m$.

Subtraction can be realized by computing the inverse element. For $c_m := (c_0, c_1)$, its inverse is $c_{-m} := (1/c_0, 1/c_1)$.

*b) CL Encryption for Group Elements:* We also define encryption and decryption algorithms $(\mathsf{GEnc}, \mathsf{GDec})$ for plaintext $\mathfrak{M} \in \mathfrak{G}$. The IND-CPA-security of this variant can be easily shown like ElGamal encryption under DDH in $\mathfrak{G}$ [17].

- $\mathsf{GEnc}(\mathsf{ek}, \mathfrak{M}; r) \to c_{\mathfrak{M}}$: Pick $r \leftarrow\$ \mathcal{D}_q$, output $(\mathfrak{g}_q^r, \mathfrak{M}\mathsf{ek}^r)$.
- $\mathsf{GDec}(\mathsf{dk}, c_{\mathfrak{M}}) \to \mathfrak{M}/\perp$: Parse $c_{\mathfrak{M}} = (c_0, c_1)$, output $c_1/c_0^{\mathsf{dk}}$.

## III. Secret Sharing and Dual-Code Verification

We present $\mathbb{Z}$-SS – Shamir secret sharing [10] over integers. We then extend the dual-code technique [12] to verify the commitments of shares to ensure their consistency, so any $t$-size subset $\mathcal{P}'$ will yield the same correct polynomial.

### A. Shamir Secret Sharing over Integers

$(t, n)$-Shamir secret sharing for secret $x \in \mathbb{F}_q$ is defined by:

- $\mathsf{Share}(x, \mathcal{P}) \to \{x_i\}_{i\in\mathcal{P}}$: Pick a $(t-1)$-degree polynomial $F(z) := \sum_{d=0}^{t-1} a_d z^d \in \mathbb{F}_q[z]$ s.t. $a_0 = x$, return $\{F(i)\}_{i\in\mathcal{P}}$.
- $\mathsf{Reconst}(\{x_i\}_{i\in\mathcal{P}'}) \to x$: If $\mathcal{P}' \in \mathbb{A}$, return $x := \sum_{i\in\mathcal{P}'} L_{i,\mathcal{P}'} x_i$, where $L_{i,\mathcal{P}'} = \prod_{j\in\mathcal{P}'\setminus\{i\}} \frac{j}{j-i}$.

$(t, n)$-Shamir secret sharing SS satisfies the following:

- Correctness: Given any subset $\mathcal{P}' \subseteq \mathcal{P}$ of correct shares $\{x_i\}_{i\in\mathcal{P}'}$, where $|\mathcal{P}'| \geq t$, *i.e.*, it satisfies the $(t, n)$ threshold access structure $\mathbb{A}$ ($\mathcal{P}' \in \mathbb{A}$), Reconst uniquely recovers $x$.
- Unconditional secrecy: The view of any adversary, who corrupts up to a set $\mathcal{C} \notin \mathbb{A}$ of parties, is independent of $x$.

[5] Changes in a challenge may nullify the simulation for other challenges due to adversary-imposed depdendices, requiring an exponential-time simulation.

[6] Concatenation of random challenges (in bits) is used as the only challenge of the proof system. The equivocable commitment allows a trapdoor owner to open the commitment $\mathsf{H}_{\mathsf{Com}}(m, r)$ of the message $m$ and the randomness $r$ as $(m', r')$. A hash-based commitment is efficiently equivocable if $\mathsf{H}_{\mathsf{Com}}$ is modeled by a programmable random oracle.

[7] The group elements are classes of ideals of the order of discriminant $\Delta_q$. We refer to [17] for more details on the parameter.

*a) Secret sharing for Integers:* $\mathsf{SS.Reconst}$ involves division in the Lagrange coefficient $L_{i,\mathcal{P}'}$, which is unsupported by unknown-order groups. $\mathsf{Reconst}$ in $\mathbb{Z}$-$\mathsf{SS}$ thus returns $\Delta^2 s$ instead of the secret $s$, where $\Delta := n!$.

$\mathbb{Z}$-$\mathsf{SS}$ := $(\mathsf{Share}, \mathsf{Reconst})$ for $x \in [0, B]$ is defined as follows. Let $l := \lceil \log_2 B \rceil$ and $l^* := (l+1) + 2\log t + n \log n$.

- $\mathsf{Share}(x, \mathcal{P}) \to \{x_i\}_{i\in\mathcal{P}}$: Pick $a_1, \ldots, a_{t-1} \leftarrow\$ [0, 2^{l^*+\lambda_d}]$, set $F(z) := \Delta x + \sum_{d=1}^{t-1} a_d z^d$, return $x_i := F(i)$ for $i \in \mathcal{P}$.
- $\mathsf{Reconst}(\{x_i\}_{i\in\mathcal{P}'}) \to x'$: If $\mathcal{P}'(\subseteq \mathcal{P})$ is of size $t$ or larger, return $x' := \sum_{i\in\mathcal{P}'} (\Delta L_{i,\mathcal{P}'}) x_i$.

*b) Security:* For any set $\mathcal{P}' \in \mathbb{A}$, Lagrange interpolation gives $F(0) = \sum_{i\in\mathcal{P}'} L_{i,\mathcal{P}'} F(i)$ for $(t-1)$-degree polynomial $F(z)$. Thus, for $F(0) = \Delta x$ in $\mathbb{Z}$-$\mathsf{SS}$, $\mathsf{Reconst}(\{x_i\}_{i\in\mathcal{P}'})$ returns $\sum_{i\in\mathcal{P}'} (\Delta L_{i,\mathcal{P}'}) x_i = \Delta F(0) = \Delta^2 x$. This scheme [10, Theorem 16] is statistically private: no information about secret $x$ is leaked to any adversary corrupting $\mathcal{C} \notin \mathbb{A}$ if $l^* \geq (l+1) + \lceil \log(t \cdot 2h_{\max}) \rceil$, where we suggest[8] $h_{\max} \leq (t-1)n^n$.

The value of share is upper bounded by $(n^t) \cdot 2^{l^*+\lambda_d}$, where $2^{l^*+\lambda_d}$ and $n^t$ are contributed by the coefficient term of $F(z)$ and the sum of the geometric series ($n^t < \sum_{d=0}^{t-1} z^i$ for $z \in [1, n]$), respectively. The share is thus of $(l^* + \lambda_d + t \log n)$-bit.

*c) Simulating Lifted Shares:* Shares are often lifted to the exponent to allow verifiability. Security proof requires the simulation of these "public shares" consistent with shares from corrupted parties. Meanwhile, Feldman's simulation does not work since division is unavailable here. We present Lemma 1, which recovers the public shares directly.[9]

**Lemma 1.** *Let $F(z) = \Delta x + \sum_{d=1}^{t-1} a_d z^d$ be the $(t-1)$-degree polynomial in $\mathbb{Z}$-$\mathsf{SS}$. Given the lifted secret $\mathfrak{g}_q^{F(0)}$ and $(t-1)$ shares $\{F(i)\}_i$, one can efficiently compute $\mathfrak{g}_q^{\Delta F(\ell)}$ for $\ell \in \mathbb{Z}$.*

$F(\ell) = \prod_{i\in\mathcal{P}'} L_{i,\mathcal{P}'}(\ell) F(i)$ by interpolation, where $L_{i,\mathcal{P}'}(\ell) = \prod_{j\in\mathcal{P}'\setminus\{i\}} \frac{j-\ell}{j-i}$. Given $t-1$ shares $\{F(i)\}_i$ and the lifted secret $\mathfrak{g}_q^{F(0)}$, we can simulate the value in exponent as $\mathfrak{g}_q^{\Delta F(\ell)} = (\mathfrak{g}_q^{F(0)})^{\Delta L_{0,\mathcal{P}}(\ell)} \cdot \prod_{i\in\mathcal{P}'} \mathfrak{g}_q^{\Delta L_{i,\mathcal{P}}(\ell) F(i)}$. Following the existing argument [45] (detailed in Appendix D-C), $\Delta$ alone suffices to cancel the denominators in the Lagrange coefficient.

### B. Dual-Code Verification for $\mathbb{Z}$-$\mathsf{SS}$

Consistency of shares can be verified via dual codes [12].

**Lemma 2.** *[12, Lemma 1] Let $C$ and its dual $C^\perp$ be $C := \{(F(1), F(2), \ldots, F(n)) : F(z) \in \mathbb{F}_q[z], \deg F(z) \leq t-1\}$ and $C^\perp := \{(v_1 P(1), v_2 P(2), \ldots, v_n P(n)) : P(z) \in \mathbb{F}_q[z], \deg P \leq n-t-1\}$, where $v_i := \prod_{j=1, j\neq i}^{n} \frac{1}{i-j}$. If $(x_1, \ldots, x_n) \in \mathbb{F}_q^n \setminus C$, and $(x_1^\perp, \ldots, x_n^\perp)$ is chosen uniformly at random in $C^\perp$, the probability that $\sum_{i=1}^n x_i x_i^\perp = 0$ is $1/q$.*

At a high level, Lemma 2 says that for any dual code $(x_1^\perp, x_2^\perp, \ldots, x_n^\perp)$, if the set of shares $(x_1, x_2, \ldots, x_n)$ is inconsistent, *i.e.*, the shares are generated from a polynomial with degree $\geq t$, then this set of shares passes the verification $\sum_{i=1}^n x_i x_i^\perp = 0$ with negligible probability. We define an algorithm $\mathsf{Dual}$ that returns dual elements $\{x_i^\perp\}_{i\in\mathcal{P}}$ in $C^\perp$, where $\mathcal{P}$ is the set of evaluation points of the polynomial.

- $\mathsf{SS.Dual}(\mathcal{P}) \to \{x_i^\perp\}_{i\in\mathcal{P}}$: Pick a polynomial $P(z) \leftarrow \mathbb{F}_q[z]$ such that $\deg P = n-t-1$, return $x_i^\perp := v_i P(i)$ for $i \in \mathcal{P}$, where $v_i = \prod_{j\in\mathcal{P}\setminus\{i\}} \frac{1}{i-j}$.

*a) Extension to Commitment:* The dual code can verify the lifted shares $\{g^{F(i)}\}_{i\in\mathcal{P}}$. We further extend this verification to Pedersen commitments $\{\mathsf{PC}_{F(i)} := g^{F(i)} h^{F'(i)}\}_{i\in\mathcal{P}}$ of share $F(i)$, where $F(z), F'(z)$ are $(t-1)$-degree polynomials.

**Lemma 3.** *Let $\mathsf{PC}_{F(i)} = g^{F(i)} h^{F'(i)}$ be a Pedersen commitment of share $F(i)$ using randomness $F'(i)$. For $\{x_i^\perp\}_{i\in\mathcal{P}} \leftarrow \mathsf{SS.Dual}(\mathcal{P})$, if $\prod_{i\in\mathcal{P}} \mathsf{PC}_{F(i)}^{x_i^\perp} = 1$, the probability that $\deg F > t-1$ is negligible if the commitment is binding.*

Intuitively, passing verification implies $g^a h^b = 1$, which break the binding property of the commitment when $a, b \neq 0$. We defer the proof to Appendix D-A.

*b) Extension to Commitment in Class Groups:* We use the Pedersen commitments in class groups [1], *i.e.*, $\mathfrak{h}^m \mathfrak{g}_q^r$ is a commitment of $m$ with randomness $r$ over $\mathfrak{G}$. $\mathbb{Z}$-$\mathsf{SS.Dual}(\mathcal{P})$ verifies consistency among committed shares.

- $\mathbb{Z}$-$\mathsf{SS.Dual}(\mathcal{P}) \to \{x_i^\perp\}_{i\in\mathcal{P}}$: Pick $b_d \leftarrow\$ [0, B]$ for $d \in [0, n-t-1]$, set $P(z) := \sum_{d=0}^{n-t-1} b_d z^d$ and $v_i := \Delta \prod_{j\in\mathcal{P}\setminus\{i\}} \frac{1}{i-j} \in \mathbb{Z}$, return $x_i^\perp := v_i P(i)$ for $i \in \mathcal{P}$.

**Lemma 4.** *Let $\mathfrak{h}^{F'(i)} \mathfrak{g}_q^{F(i)}$ be a commitment of share $F(i)$. For $\{x_i^\perp\}_{i\in\mathcal{P}} \leftarrow \mathbb{Z}$-$\mathsf{SS.Dual}(\mathcal{P})$, if $\prod_{i\in\mathcal{P}} (\mathfrak{h}^{F'(i)} \mathfrak{g}_q^{F(i)})^{x_i^\perp} = 1$, the probability that $\deg F > t-1$ is negligible.*

Lemma 4 is analogous to Lemma 3, but in $\mathfrak{G}$ instead of $\mathbb{G}$. The proof is deferred to Appendix D-B.

## IV. Rushing-resilient Distributed Key Generation

We propose a 2-round distributed key generation protocol for discrete-logarithm-based cryptosystems over a prime-order cyclic group $\mathbb{G}$, *e.g.*, an elliptic curve. We then extend it to support unknown-order class groups (with integer secret keys).

We use a publicly verifiable linearly homomorphic encryption scheme to distribute the encrypted shares [30], so no additional communication round is needed to resolve the private complaints. However, 1-round construction is impossible [40] to defend against rushing adversaries, who bias the distribution of the key. We thus use the 2-stage paradigm [34]: the first stage distributes shares of an initial secret (so the initial secret is fixed without leaking); the second stage reveals all the (lifted) initial secrets for computing the resultant key.

Our protocol follows the distributed randomness generation (DRG) construction of Wong *et al.* [48], except using dual-code-based verification to verify the consistency of broadcasted committed shares for reconstruction. Their correspondence with the encrypted shares is maintained via NIZK.

### A. Security Definition of $\mathsf{DKG}$

Recall that $\mathsf{DKG}$ correctness requires three conditions [34]:

[8] We bound the value $h_{\max} < (t-1)n^n$ that it is not provided [10, Theorem 16] and suggest the value to be $l^* := l + 1 + 2\log t + 2n \log n$.

[9] Alternatively, [10, Lemma 8] unnecessarily recovers the polynomial $F$.

- (C1.) All sufficiently-large subsets ($\mathcal{P}' \in \mathbb{A}$) of shares from the honest parties define the same unique secret key $x$.
- (C2.) All honest parties have the same value of the public key $X = g^x \bmod q$, where $x$ is the unique secret (as (C1)).
- (C3.) $x$ is uniformly distributed in $\mathbb{F}_q$ (so does $X$ in $\langle g \rangle$).

Robustness requires that the protocol always completes successfully if a set $\mathcal{P} \in \mathbb{A}$ of honest parties participates. Secrecy mandates a simulator that can simulate on input $X$ a view indistinguishable from that of any PPT adversary $\mathcal{A}$ corrupting a set $\mathcal{C} \notin \mathbb{A}$ of parties in a real protocol run that outputs $X$.

### B. DKG *for Discrete-logarithm-based Cryptosystems*

Figure 1 presents our DKG-DL protocol built upon Pedersen commitment, Shamir secret sharing SS, and linearly homomorphic encryption CLE. For brevity, all algorithms implicitly take in the encryption keys $\{\mathsf{ek}_j\}_{j\in\mathcal{P}}$ of all parties.

The protocol is run by an *updating* set of parties $\mathcal{P}$. At the end of the generation phase, the distributively-generated secret is fixed by the contribution from and hence the shares held by the honest parties. We call this set the *qualified set* $\mathcal{Q}$. $\mathcal{P}$ may be updated to remove cheaters identified during GenVf, and $\mathcal{Q} := \mathcal{P}$ is fixed afterward. More cheaters may be removed from $\mathcal{P}$ during RevealVf, but it will not affect $\mathcal{Q}$.

*1) Generation Phase (*Gen $\leftrightarrow$ GenVf*):* Suppose $|\mathcal{P}| = n$; we aim to $(t, n)$-threshold-share the initial secret $\chi_i$ sampled by the dealer $\mathsf{P}_i$ with all other parties.

$\mathsf{P}_i$ runs $\mathsf{Share}(\chi_i, \mathcal{P})$ to generate $\{\chi_{ij}\}_{j\in\mathcal{P}}$ of initial secret $\chi_i$. $\forall j \in \mathcal{P}$ (Line 3-6), each share $\chi_{ij}$ is then encrypted using the receiving party's encryption key $\mathsf{ek}_j$, and its well-formedness is ensured by $\mathsf{Z}_{\mathsf{Enc\text{-}PC}}$, which binds to Pedersen commitments $\mathsf{PC}_{\chi_{ij}}$ of $\chi_{ij}$ too. Commitments, ciphertexts, and proofs are broadcasted to all and can be publicly verified.

GenVf verifies $\mathsf{Z}_{\mathsf{Enc\text{-}PC}}$ and, via dual code in the form of commitment, whether all delivered shares constitute $(t, n)$-threshold shares of a secret. Parties supplying incorrect shares will be removed. At the end of this phase, the combined secret (key) $x := \sum_{j\in\mathcal{Q}} \chi_j$ is fixed for the qualified set $\mathcal{Q} := \mathcal{P}$. Public verification ensures that the shares are correct, so any problem detected afterward is not imputed to the dealer.

*2) Revelation Phase (*Reveal $\leftrightarrow$ RevealVf*):* We aim to generate $(t, n)$ threshold shares $\{x_i\}_{i\in\mathcal{P}}$ of the final secret $x = \sum_{j\in\mathcal{Q}} \chi_j$ via linear homomorphism of the shares. This phase equips $\mathsf{P}_i$ with secret share $x_i$ and public key $X = g^x$.

After verification (of ciphertexts) of the generation phase, $\mathsf{P}_i \in \mathcal{Q}$ runs Reveal to decrypt the homomorphically-added ciphertext $c_{x_i}$ and obtains the share $x_i$ of the secret key $x$. $\mathsf{P}_i$ asserts the correctness of the lifted share $X_i$ against the ciphertext $c_{x_i}$ by $\mathsf{Z}_{\mathsf{Dec\text{-}DL}}$ for correct decryption. $\mathsf{P}_i$ then broadcasts the lifted shares $X_i$ and its proof $\pi_{X_i}$. At the end, after the verification (of $\mathsf{Z}_{\mathsf{Dec\text{-}DL}}$) done by RevealVf, the public key $X$ is computed via interpolating in exponent $\prod_{i\in\mathcal{P}} X_i^{L_{i,\mathcal{P}}}$.

We defer the correctness and security arguments (Lemmas 8 and 9) to Appendix B.

### C. DKG *for Class Groups*

Figure 2 presents DKG-CL for class groups. Similar to DKG-DL, publicly verifiable linearly homomorphic encryption

**DKG-DL**

$\mathsf{Gen}(\mathsf{pp}) \to \{\mathsf{PC}_{\chi_{ij}}, c_{\chi_{ij}}, \pi_{\chi_{ij}}\}_{j\in\mathcal{P}}$

1 : $\chi_i \leftarrow\!\$\ \mathbb{F}_q, \{\chi_{ij}\}_{j\in\mathcal{P}} \leftarrow \mathsf{Share}(\chi_i, \mathcal{P})$
2 : $\chi'_i \leftarrow\!\$\ \mathbb{F}_q, \{\chi'_{ij}\}_{j\in\mathcal{P}} \leftarrow \mathsf{Share}(\chi'_i, \mathcal{P})$
3 : **foreach** $j \in \mathcal{P}$ **do**
4 : $\quad \mathsf{PC}_{\chi_{ij}} := g^{\chi_{ij}} h^{\chi'_{ij}},\ c_{\chi_{ij}} \leftarrow \mathsf{Enc}(\mathsf{ek}_j, \chi_{ij}; r_{\chi_{ij}})$
5 : $\quad \pi_{\chi_{ij}} \leftarrow \mathsf{Z}_{\mathsf{Enc\text{-}PC}}.\mathsf{P}((\mathsf{PC}_{\chi_{ij}}, c_{\chi_{ij}}, \mathsf{ek}_j);$
6 : $\qquad\qquad (\chi_{ij}, \chi'_{ij}, r_{\chi_{ij}}))$
7 : **return** $\{\mathsf{PC}_{\chi_{ij}}, c_{\chi_{ij}}, \pi_{\chi_{ij}}\}_{j\in\mathcal{P}}$

$\mathsf{GenVf}(\{\{\mathsf{PC}_{\chi_{ij}}, c_{\chi_{ij}}, \pi_{\chi_{ij}}\}_{j\in\mathcal{P}}\}_{i\in\mathcal{P}})$

1 : $\{x_j^{\perp}\}_{j\in\mathcal{P}} \leftarrow \mathsf{SS.Dual}(\mathcal{P})$
2 : **foreach** $i \in \mathcal{P}$ **do**
3 : $\quad$ **if** $\prod_{j\in\mathcal{P}} \mathsf{PC}_{\chi_{ij}}^{x_j^{\perp}} \neq 1$ **then** $\mathcal{P} := \mathcal{P} \setminus \{i\}$
4 : $\quad$ **else foreach** $j \in \mathcal{P}$ **do**
5 : $\qquad$ **if** $\neg \mathsf{Z}_{\mathsf{Enc\text{-}PC}}.\mathsf{Vf}(\pi_{\chi_{ij}}, (\mathsf{PC}_{\chi_{ij}}, c_{\chi_{ij}}, \mathsf{ek}_j))$
6 : $\qquad\quad \mathcal{P} := \mathcal{P} \setminus \{i\},$ **break**
7 : $\mathcal{Q} := \mathcal{P}$

$\mathsf{Reveal}(\{c_{\chi_{ji}}\}_{j\in\mathcal{Q}}, \mathsf{dk}_i) \to (x_i, X_i, \pi_{X_i})$

1 : $c_{x_i} := \boxplus_{j\in\mathcal{Q}} c_{\chi_{ji}},\ x_i \leftarrow \mathsf{Dec}(\mathsf{dk}_i, c_{x_i}),\ X_i := g^{x_i}$
2 : $\pi_{X_i} \leftarrow \mathsf{Z}_{\mathsf{Dec\text{-}DL}}.\mathsf{P}((X_i, c_{x_i}, \mathsf{ek}_i); (x_i, \mathsf{dk}_i))$
3 : **return** $(x_i, X_i, \pi_{X_i})$

$\mathsf{RevealVf}(\{\{c_{\chi_{ji}}\}_{j\in\mathcal{Q}}, X_i, \pi_{X_i}\}_{i\in\mathcal{P}}) \to X$

1 : **foreach** $i \in \mathcal{P}$ **do**
2 : $\quad c_{x_i} := \boxplus_{j\in\mathcal{Q}} c_{\chi_{ji}}$
3 : $\quad$ **if** $\neg \mathsf{Z}_{\mathsf{Dec\text{-}DL}}.\mathsf{Vf}(\pi_{X_i}, (X_i, c_{x_i}, \mathsf{ek}_i))$
4 : $\qquad \mathcal{P} := \mathcal{P} \setminus \{i\}$
5 : **return** $X := \prod_{i\in\mathcal{P}} X_i^{L_{i,\mathcal{P}}}$

Fig. 1: Distributed Key Generation for Discrete-Log Keys

is used to deliver the shares. Pedersen commitments and NIZK are used to verify the consistency among the shares. After delivery of correct shares, DKG-CL fixes the secret key. The parties then give the shares of the public key (lifted secret key) for reconstruction. For secrets lying in $\mathbb{Z}_s$ (integers modulo an unknown order $s$), we make the changes below to DKG-DL.

First, the initial secret $\chi_i$ is a large integer in $[0, B]$ without knowing the modulus $s$, so it requires Shamir secret sharing over integer $\mathbb{Z}$-SS (Line 1-2 of Gen) to distribute it.

Second, the protocol prepares a ciphertext $c_{\mathfrak{X}_{ij}}$ (Line 5 of Gen), which helps to show the correctness of the public key (lifted secret key)'s share $\mathfrak{X}_i = \mathfrak{g}_q^{\chi_i}$ to be Reveal-ed. Recall that DKG-DL verifies the public key's shares $X_i$ against the (homomorphically-evaluated) decrypted ciphertext $c_{x_i}$. This fails for DKG-CL because the plaintext space size is only $q$ but $x_i \in \mathbb{Z}$. Specifically, homomorphism gives $c_{x_i} = \boxplus_{j\in\mathcal{Q}} c_{\chi_{ji}}$, where $c_{\chi_{ji}} = \boxplus_{\ell=0}^{\mathsf{len}-1}(q^{\ell} \boxdot c_{\chi_{ji\ell}})$. However, the plaintext space is $\mathbb{F}_q$, we have $c_{\chi_{ji}} = c_{\chi_{ji0}}$ that only $x_i \bmod q$ is verified.

With plaintext space $\mathfrak{G}$, the extra ciphertext $c_{\mathfrak{X}_{ij}}$ can be used to verify $x_i \in \mathbb{Z}$. One could generate a proof of the

DKG-CL

$\mathsf{Gen}(\mathsf{pp}) \to \{\mathsf{PC}_{\chi_{ij}}, \{c_{\chi_{ij\ell}}\}_\ell, c_{\mathfrak{x}_{ij}}, \pi_{\chi_{ij}}\}_{j\in\mathcal{P}}$

1 : $\chi_i \leftarrow\$ [0, B], \{\chi_{ij}\}_{j\in\mathcal{P}} \leftarrow \mathbb{Z}\text{-SS.Share}(\chi_i, \mathcal{P})$
2 : $\chi'_i \leftarrow\$ [0, B], \{\chi'_{ij}\}_{j\in\mathcal{P}} \leftarrow \mathbb{Z}\text{-SS.Share}(\chi'_i, \mathcal{P})$
3 : **foreach** $j \in \mathcal{P}$ **do**
4 : $\quad \mathsf{PC}_{\chi_{ij}} := \mathfrak{h}^{\chi_{ij}} \mathfrak{g}_q^{\chi'_{ij}}$
5 : $\quad \mathfrak{X}_{ij} := \mathfrak{g}_q^{\chi_{ij}\Delta},\ c_{\mathfrak{x}_{ij}} \leftarrow \mathsf{GEnc}(\mathsf{ek}_j, \mathfrak{X}_{ij}; r_{\mathfrak{x}_{ij}})$
6 : $\quad$ **write** $\chi_{ij}$ in $q$-ary $= \sum_{\ell=0}^{\mathsf{len}-1} q^\ell \chi_{ij\ell}$
7 : $\quad$ **for** $\ell \in [0, \mathsf{len}-1]$ **do**
8 : $\quad\quad c_{\chi_{ij\ell}} \leftarrow \mathsf{Enc}(\mathsf{ek}_j, \chi_{ij\ell}; r_{\chi_{ij\ell}})$
9 : $\quad \pi_{\chi_{ij}} \leftarrow \mathsf{Z}_{\mathsf{BInt}}.\mathsf{P}((\mathsf{PC}_{\chi_{ij}}, \{c_{\chi_{ij\ell}}\}_\ell, c_{\mathfrak{x}_{ij}}, \mathsf{ek}_j);$
10 : $\quad\quad (\{\chi_{ij\ell}\}_\ell, \chi'_{ij}, \{r_{\chi_{ij\ell}}\}_\ell, r_{\mathfrak{x}_{ij}}))$
11 : **return** $\{\mathsf{PC}_{\chi_{ij}}, \{c_{\chi_{ij\ell}}\}_\ell, c_{\mathfrak{x}_{ij}}, \pi_{\chi_{ij}}\}_{j\in\mathcal{P}}$

$\mathsf{GenVf}(\{\{\mathsf{PC}_{\chi_{ij}}, \{c_{\chi_{ij\ell}}\}_\ell, c_{\mathfrak{x}_{ij}}, \pi_{\chi_{ij}}\}_{j\in\mathcal{P}}\}_{i\in\mathcal{P}})$

5 : **if** $\neg \mathsf{Z}_{\mathsf{BInt}}.\mathsf{Vf}(\pi_{\chi_{ij}}, (\mathsf{PC}_{\chi_{ij}}, \{c_{\chi_{ij\ell}}\}_\ell, c_{\mathfrak{x}_{ij}}, \mathsf{ek}_j))$
6 : $\quad$ / all other lines are the same as GenVf in Figure 1, except $\mathbb{Z}$-SS

$\mathsf{Reveal}(\{\{c_{\chi_{ji\ell}}\}_\ell, c_{\mathfrak{x}_{ji}}\}_{j\in\mathcal{Q}}, \mathsf{dk}_i) \to (x_i, \mathfrak{X}_i, \pi_{\mathfrak{x}_i})$

1 : **foreach** $j \in \mathcal{P}$ **do** $\chi_{ji} := \sum_{\ell=0}^{\mathsf{len}-1} q^\ell \mathsf{Dec}(\mathsf{dk}_i, c_{\chi_{ji\ell}})$
2 : $\quad x_i := \sum_{j\in\mathcal{Q}} \chi_{ji},\ c_{\mathfrak{x}_i} := \boxplus_{j\in\mathcal{Q}} c_{\mathfrak{x}_{ji}},\ \mathfrak{X}_i := \mathfrak{g}_q^{\Delta x_i}$
3 : $\quad \pi_{\mathfrak{x}_i} \leftarrow \mathsf{Z}_{\mathsf{GDec\text{-}CL}}.\mathsf{P}((\mathfrak{X}_i, c_{\mathfrak{x}_i}, \mathsf{ek}_i, \mathfrak{g}_q^\Delta); (\mathsf{dk}_i, x_i))$
4 : **return** $(x_i, \mathfrak{X}_i, \pi_{\mathfrak{x}_i})$

$\mathsf{RevealVf}(\{\{c_{\mathfrak{x}_{ji}}\}_{j\in\mathcal{Q}}, \mathfrak{X}_i, \pi_{\mathfrak{x}_i}\}_{i\in\mathcal{P}}) \to \mathfrak{X}$

1 : **foreach** $i \in \mathcal{P}$ **do**
2 : $\quad c_{\mathfrak{x}_i} := \boxplus_{j\in\mathcal{Q}} c_{\mathfrak{x}_{ji}}$
3 : $\quad$ **if** $\neg \mathsf{Z}_{\mathsf{GDec\text{-}CL}}.\mathsf{Vf}(\pi_{\mathfrak{x}_i}, (\mathfrak{X}_i, c_{\mathfrak{x}_i}, \mathsf{ek}_i, \mathfrak{g}_q^\Delta))$
4 : $\quad\quad \mathcal{P} := \mathcal{P} \setminus \{i\}$
5 : **return** $\mathfrak{X} := \prod_{i\in\mathcal{P}} \mathfrak{X}_i^{\Delta L_{i,\mathcal{P}}}$

Fig. 2: Distributed Key Generation for Class Groups

share $x_i$ w.r.t. the ciphertexts $\{\chi_{ji\ell}\}_{j,\ell}$, yet it takes $\mathcal{O}(n \cdot \mathsf{len})$-size statement and witness. One could also verify against the commitment $\mathsf{PC}_{\chi_{ij}}$, but it requires extra ciphertexts for sending the randomness $\chi'_{ij}$.

Third, the share $\chi_{ij}$ is a big integer in $[0, (n^t) \cdot 2^{l^*+\lambda_d}]$, but CLE only supports plaintext in $\mathbb{F}_q$. Thus, $\chi_{ij}$ is decomposed in base $q$ (Line 6-8 of Gen) such that $\chi_{ij} = \sum_{\ell=0}^{\mathsf{len}-1} q^\ell \chi_{ij\ell}$; and $\mathsf{P}_i$ delivers ciphertexts $\{c_{\chi_{ij\ell}}\}_\ell$, with $\mathsf{len} := \lceil \frac{\log((n^t)\cdot 2^{l^*+\lambda_d})}{\log(q)} \rceil$.

We defer the correctness and security argument (Lemmas 10 and 11) to Appendix B.The argument is analogous to the argument for DKG-DL but with modifications for class groups, *i.e.*, using $\mathbb{Z}$-SS and multiple ciphertexts with ZKP to transmit shares, which does not affect the flow of argument.

### D. Communication Round vs. Efficiency

Although DKG-CL does not need to resolve complaints, Gen and GenVf in the generation phase involve $O(n\mathsf{len})$ and $O(n^2\mathsf{len})$ operations, respectively. Appendix C presents a 3-round construction from hybrid encryption to remove the $O(\mathsf{len})$ factor. As a tradeoff, the correspondence between the hybrid encryption and commitment becomes difficult to prove by ZKPs. Thus, only private verification is provided, requiring one extra round to resolve complaints, if any.

## V. Threshold Homomorphic Encryption from DKG

We introduce a threshold variant of CLE, t-CL $=$ (Setup, KGen, Enc, PartDec, FinDec), using our DKG in Section IV-C to produce a uniformly distributed key. (Eval is the same as CLE.Eval.) We optimize the decryption algorithm [10] by reducing the exponent from $\Delta^3$ to $\Delta^2$, where $\Delta = n!$.

### A. Construction

Setup first distributively initializes a class group for CLE and samples a random base element $\mathfrak{h}$ for the class-group commitment. Each party then invokes KGen (with ZKP of knowledge[10]) to obtain the CLE key pairs for DKG-CL. KGen runs DKG-CL to generate a threshold key pair $(\mathsf{ek}_i, \mathsf{dk}_i)$ of t-CL, where $\mathsf{ek}_i = (\mathfrak{g}_q^{\Delta\mathsf{dk}_i})$ and $\mathsf{ek} = \mathfrak{g}_q^{\Delta^3\mathsf{dk}} = \prod_{i\in\mathcal{P}} \mathsf{ek}_i^{\Delta L_{i,\mathcal{P}}}$.

- $\mathsf{Setup}\langle 1^\lambda\rangle \to \langle \mathsf{pp}\rangle$:
  1) Distributive setup of the class group:
     a) $\mathsf{P}_i$ picks $q_i \leftarrow\$ \{0,1\}^\lambda$, broadcasts $h_{q_i} := \mathsf{H}(q_i)$.
     b) After all $h_{q_i}$ are received, $\mathsf{P}_i$ broadcasts $q_i$.
     c) For each $j \in \mathcal{P}$, if $h_{q_j} \neq \mathsf{H}(q_j)$, set $\mathcal{P} := \mathcal{P}\setminus\{j\}$. Set $q' \leftarrow \mathsf{next\text{-}prime}(\oplus_j q_j)$, where next-prime returns the prime number just larger than the input number. $\mathsf{P}_i$ runs $\mathsf{CLE.Setup}(q, q', 1^\lambda) \to \mathsf{pp}_{\mathsf{CL}}$.
  2) Distributive sampling of $\mathfrak{h}$:
     a) $\mathsf{P}_i$ picks $\mathfrak{h}_i \leftarrow\$ \mathfrak{G}^q$, broadcasts $h_{\mathfrak{h}_i} := \mathsf{H}(\mathfrak{h}_i)$.
     b) After all $h_{\mathfrak{h}_i}$ are received, $\mathsf{P}_i$ broadcasts $\mathfrak{h}_i$.
     c) For each $j \in \mathcal{P}$, if $h_{\mathfrak{h}_i} \neq \mathsf{H}(\mathfrak{h}_i)$, sets $\mathcal{P} := \mathcal{P} \setminus \{j\}$.
     d) Returns $\mathfrak{h} := \prod_{j\in\mathcal{P}} \mathfrak{h}_j$ if $|\mathcal{P}| \geq t$; abort otherwise.
  3) $\mathsf{P}_i$ runs $\mathsf{CLE.KGen}(\mathsf{pp}) \to (\mathsf{CLE.ek}_i, \mathsf{CLE.dk}_i)$, broadcasts $\mathsf{CLE.ek}_i$, invokes the multi-party $\Sigma$-protocol $\Sigma^*_{\mathsf{CL}}\langle\{\mathsf{CLE.ek}_j\}_{j\in\mathcal{P}}; \mathsf{CLE.dk}_i\rangle_{i\in\mathcal{P}}$, and sets $\mathcal{P} := \mathcal{P}\setminus\{j\}$ if $\mathsf{P}_j$ fails in $\Sigma^*_{\mathsf{CL}}$. Abort if $|\mathcal{P}| < t$.
  4) Everyone in $\mathcal{P}$ returns $\mathsf{pp} := (\mathsf{pp}_{\mathsf{CL}}, \mathfrak{h}, \{\mathsf{CLE.ek}_j\}_{j\in\mathcal{P}})$.
- $\mathsf{KGen}\langle \mathsf{pp}; \mathsf{CLE.dk}_i\rangle_{i\in\mathcal{P}} \to \langle \mathsf{ek}, \{\mathsf{ek}_j\}_{j\in\mathcal{P}}; \mathsf{dk}_i\rangle_{i\in\mathcal{P}}$:
  1) Invoke $\mathsf{DKG\text{-}CL}\langle \mathsf{pp}; \mathsf{CLE.dk}_i\rangle_{i\in\mathcal{P}} \to \langle \mathfrak{X}; x_i\rangle_{i\in\mathcal{P}}$.
  2) Return $\mathsf{ek} := \mathfrak{X}$, $\mathsf{ek}_j := \mathfrak{g}_q^{\Delta x_j}$, and $\mathsf{dk}_i := x_i$.

The encryption algorithm Enc slightly differs from that in CLE. It uses $\mathfrak{g}_q^{\Delta^2}$ as the base element to generate $c_0 = (\mathfrak{g}_q^{\Delta^2})^r$.

Partial decryption PartDec returns a share $\mathsf{cpd}_i = c_0^{\Delta\mathsf{dk}_i}$ and its proof $\pi_{\mathsf{cpd}_i}$. Final decryption FinDec discards $\mathsf{pd}_j$ if $\pi_{\mathsf{cpd}_j}$ is invalid. Interpolation of correct shares $\{\mathsf{cpd}_j\}_{j\in\mathcal{P}'\in\mathbb{A}}$ reconstructs $\prod_{j\in\mathcal{P}'} \mathsf{cpd}_j^{\Delta L_{j,\mathcal{P}'}} = c_0^{\Delta^3\mathsf{dk}} = (\mathsf{ek}^r)^{\Delta^2}$. With $c_1^{\Delta^2} = (\mathfrak{f}^m \mathsf{ek}^r)^{\Delta^2}$, $m$ is recovered as $\mathsf{Dlog}(\mathfrak{f}^{\Delta^2 m})/\Delta^2 \bmod q$.

- $\mathsf{Enc}(\mathsf{ek}, m; r) \to c_m$: Pick $r \leftarrow\$ \mathcal{D}_q$, output $((\mathfrak{g}_q^{\Delta^2})^r, \mathfrak{f}^m \mathsf{ek}^r)$.
- $\mathsf{PartDec}(\mathsf{ek}_i, c, \mathsf{dk}_i) \to \mathsf{pd}_i$:
  1) Parse $c$ as $(c_0, c_1)$, compute $\mathsf{cpd}_i := c_0^{\Delta\mathsf{dk}_i}$.
  2) Run $\mathsf{Z}_{\mathsf{PartDec}}.\mathsf{P}((\mathsf{ek}_i, c_0^\Delta, \mathsf{cpd}_i); \mathsf{dk}_i) \to \pi_{\mathsf{cpd}_i}$.

[10]The simulation argument for security necessitates extracting secrets from other participants, which can be done by share reconstruction with an honest majority. In a dishonest majority setting, our design relies on extracting the CLE decryption key, which is accomplished by a knowledge extractor. This choice justifies using a multi-party $\Sigma$-protocol instead of non-interactive ZKP.

3) Output $(\mathsf{cpd}_i, \pi_{\mathsf{cpd}_i})$.
- $\mathsf{FinDec}(\mathsf{ek}, \{\mathsf{ek}_j\}_{j\in\mathcal{P}'}, c, \{\mathsf{pd}_j\}_{j\in\mathcal{P}'}) \to m$:
  1) Parse $c$ as $(c_0, c_1)$, $\{\mathsf{pd}_j\}_{j\in\mathcal{P}'}$ as $\{(\mathsf{cpd}_j, \pi_{\mathsf{cpd}_j})\}_{j\in\mathcal{P}'}$.
  2) For $j \in \mathcal{P}'$, if $\mathsf{Z}_{\mathsf{PartDec}}.\mathsf{Vf}((\mathsf{ek}_j, c_0^{\Delta}, \mathsf{cpd}_j), \pi_{\mathsf{cpd}_j}) = 0$, set $\mathcal{P}' := \mathcal{P}' \setminus \{j\}$. Abort if $|\mathcal{P}'| < t$.
  3) Compute $M := c_1^{\Delta^2} / (\prod_{j\in\mathcal{P}'} \mathsf{cpd}_j^{\Delta L_{j,\mathcal{P}'}})$.
  4) Output $m := \mathsf{Dlog}(M)/\Delta^2 \bmod q$.

### B. Security Analysis

**Lemma 5.** *If both* CLE *and* DKG-CL *satisfy (evaluation) correctness,* t-CL *satisfies evaluation correctness, i.e., honest executions of algorithms/protocols in* t-CL *give* $m$ *correctly.*

t-CL satisfies t-ind-cpa-security, a variant of ciphertext indistinguishability under chosen plaintext attacks of LHE that allows an adversary to corrupt a set of parties $\mathcal{C} \notin \mathbb{A}$.

**Lemma 6.** *If* DKG-CL *has secrecy and* CLE *is indistinguishable under chosen plaintext attacks,* t-CL *is* t-ind-cpa*-secure.*

Threshold encryption also requires simulation security [6] to ensure the decryption key $\mathsf{dk}_i$ cannot be learnt from partial decryption $\mathsf{pd}_i$, *i.e.*, given ciphertext $c$ and message $m$, $\mathsf{pd}_i$ can be simulated indistinguishably from the real one.

**Lemma 7.** *If* DKG-CL *satisfies secrecy and* $\mathsf{Z}_{\mathsf{PartDec}}$ *is honest-verifier zero-knowledge,* t-CL *has simulation security.*

The proof strategy for CL encryption aligns with classic threshold ElGamal encryption due to structural similarity. Appendix E shows detailed proofs for the above lemmas.

The threshold decryption process $(\mathsf{PartDec}, \mathsf{FinDec})$ of t-CL is robust that the decryption succeeds whenever $\geq t$ honest parties are involved. Users can verify partial decryptions, allowing them to exclude problematic ones and proceed with decryption using only the correct partial decryptions.

### C. Property for Simple Security Proof

Given t-ind-cpa-security, the simulator for threshold decryption can simulate the partial decryptions with the knowledge of the final decryption result $m$. We outline the argument with a more accessible game-based definition, complementing its existing UC-security guarantees [10]. Given the challenge ciphertext $c_b$ from the t-ind-cpa game encrypting 0 or 1, the simulator first assumes $c_b$ encrypts $b' = 1$ and generates a ciphertext $c_m := m \boxdot c_b$. If the adversary (breaking the indistinguishability) aborts, *i.e.*, $b \neq b'$, then the simulator learns $b$ and can win the t-ind-cpa game in both cases. As a result, we conclude that the simulator can simulate the partial decryption for any desired decryption result, which is useful when using threshold encryption as a building block.

## VI. New Threshold ECDSA and BBS+ Signatures

Using threshold CL encryption in Section V, we propose efficient threshold signing protocols for ECDSA and BBS+ signatures, backgrounds of which will also be introduced. To defend against rushing adversaries, we use threshold lifted ElGamal encryption to commit to group elements. We instantiate it in the same elliptic-curve group as the signature.

### A. Definition

Let $\mathsf{DS} := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ be a digital signature scheme.

- $\mathsf{KGen}(\mathsf{pp}) \to (\mathsf{vk}, \mathsf{sk})$: On input the public parameter pp, KGen outputs a verification-signing key pair $(\mathsf{vk}, \mathsf{sk})$.
- $\mathsf{Sign}(\mathsf{sk}, \mathsf{msg}) \to \sigma$: On input sk and the message msg, Sign outputs a signature $\sigma$.
- $\mathsf{Vf}(\mathsf{vk}, \sigma, \mathsf{msg}) \to 0/1$: Vf outputs 1 if and only if $\sigma$ is a signature of msg under vk.

DS is existentially unforgeable if, given access to a signing oracle that on query msg outputs $\sigma$, the probability of any PPT adversary outputting a valid signature $\sigma^*$ on $\mathsf{msg}^* \notin \mathsf{Qry}$ is $\mathsf{negl}(\lambda)$, where Qry is the set of queried messages.

Threshold signature features a (possibly decentralized) setup, a threshold key generation protocol TKeygen, and a threshold signing protocol. $\mathsf{TKeygen}\langle\mathsf{pp}\rangle \to \langle\mathsf{vk}; \mathsf{sk}_i\rangle_{i\in\mathcal{P}}$ outputs to each $\mathsf{P}_i$ verification key vk and threshold signing key $\mathsf{sk}_i$. The threshold signing protocol outputs signature $\sigma$ if a subset $\mathcal{P}$ of parties agrees on the message msg and participates honestly with their threshold signing keys $\{\mathsf{sk}_i\}_{i\in\mathcal{P}}$.

Let $\mathsf{TS} := (\mathsf{TKeygen}, \mathsf{TSign}, \mathsf{Vf})$ be a $(t, n)$-threshold signature scheme. TS is unforgeable [33] if any PPT adversary $\mathcal{A}$, having corrupted at most $t-1$ parties and given the view of TKeygen and TSign on input messages of its adaptive choices, as well as signatures on those messages, $\mathcal{A}$ outputs a valid signature $\sigma^*$ on $\mathsf{msg}^* \notin \mathsf{Qry}$ is of negligible probability in $\lambda$.

TS is simulatable [33] if TKeygen and TSign are simulatable: TKeygen (resp. TSign) is simulatable if there exists a simulator such that, on input vk (resp. $(\mathsf{vk}, \mathsf{msg})$) and the public output of TKeygen (resp. TSign), generates an indistinguishable view of TKeygen (resp. TSign, which outputs $\sigma$).

If TKeygen is simulatable, the reduction can generate the view of TKeygen such that the threshold signature and regular signature use the same vk. Thus, a forged threshold signature is also a valid regular signature. If TSign is simulatable, the reduction queries the signing oracle of the regular signature to obtain a signature $\sigma$ of the queried message msg and uses it to simulate the view of TSign.

Protocol continuation depends on the majority setting. For dishonest majority without $t$ honest parties, the best is identifiable abort, ensuring that a corrupted party is identified. For honest majority with at least $t$ honest parties, robustness is considered, which ensures the protocol completes successfully.

### B. Threshold ECDSA

ECDSA is a variant of DSA instantiated over a group of points on an elliptic curve. Let the public parameter pp be the group description $(\mathbb{G}, q, g)$ and $m$ be the hash of the message msg. The ECDSA algorithms $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ are as follows:

- $\mathsf{KGen}(\mathsf{pp}) \to (\mathsf{vk}, \mathsf{sk})$: Randomly sample $x \leftarrow\!\$\, \mathbb{F}_q$, set $X := g^x$, and output $(\mathsf{vk}, \mathsf{sk}) := (X, x)$.
- $\mathsf{Sign}(\mathsf{sk}, \mathsf{msg}) \to \sigma$: Set $m := \mathsf{H}(\mathsf{msg})$, randomly sample $k \leftarrow\!\$\, \mathbb{F}_q$, set $R := g^{1/k}$, and output $\sigma := (r, s := k(m + rx))$, where $r$ is the x-projection of point $R$ on $\mathbb{F}_q$.
- $\mathsf{Vf}(\mathsf{vk}, \sigma, \mathsf{msg}) \to 0/1$: Parse $\sigma$ as $(r, s)$, accept if $r$ is the x-projection of $R' := (g^m X^r)^{1/s} \in \mathbb{G}$, where $m := \mathsf{H}(\mathsf{msg})$.

*a) Construction:* Our threshold ECDSA uses threshold CL encryption t-CL to encrypt plaintexts over $\mathbb{F}_q$ and threshold ElGamal encryption t-ElG over $\mathbb{G}$ to commit to elements of $\mathbb{G}$. We define $\overline{k}$, $\overline{g^k}$ to be the CL/ElGamal ciphertext of $k \in \mathbb{F}_q$, $g^k \in \mathbb{G}$ under encryption key clek, elek, respectively.

Our scheme sometimes requires homomorphic evaluation without randomization. For differentiation, randomizable evaluation is marked with $*$, *i.e.*, $\boxplus^*$ and $\boxdot^*$.

Our setup runs t-CL.Setup to set up a class group for t-CL and invokes two instances of the distributed key generation protocol DKG-DL to generate key pairs $(X, \{x_i\}_{i\in\mathcal{P}})$ for threshold ECDSA and $(\mathsf{elek}, \{\mathsf{eldk}_i\}_{i\in\mathcal{P}})$ for t-ElG. Concurrently, it invokes distributed key generation protocol DKG-CL to generate a key pair $(\mathsf{clek}, \{\mathsf{cldk}_i\}_{i\in\mathcal{P}})$ for threshold CL encryption t-CL. We assume that $\{g^{x_i}, g^{\mathsf{eldk}_i}, \mathfrak{g}_q^{\mathsf{cldk}_i}\}_{i\in\mathcal{P}}$ are broadcasted. Note that $X = \prod_{i\in\mathcal{P}} (g^{x_i})^{L_{i,\mathcal{P}}}$.

Our protocol for the offline phase of ECDSA signing before knowing the message has the following skeleton.

Step 1-2: distributively generate encrypted randomness $k$;
Step 3-4: distributively generate encrypted random factor $g^\gamma$, pseudo-key $xk$, and pseudo-nonce $\gamma k$;
Step 5-6: distributively extract (*i.e.*, threshold-decrypt) $g^\gamma$ and $\gamma k$ to compute nonce $g^{1/k} = (g^\gamma)^{\frac{1}{\gamma k}}$.

1) $\mathsf{P}_i$ encrypts $k_i$:
   a) $k_i \leftarrow\$ \mathbb{F}_q$, $\mathsf{t\text{-}CL.Enc}(\mathsf{clek}, k_i; r_{k_i}) \to \overline{k_i}$.
   b) $\mathsf{Z}_{\mathsf{Enc}}.\mathsf{P}((\mathsf{clek}, \overline{k_i}); (k_i, r_{k_i})) \to \pi_{k_i}$.
   c) Broadcast $(\overline{k_i}, \pi_{k_i})$.
2) $\mathsf{P}_i$ uses homomorphism to generate LHE of $k := \sum_{j\in\mathcal{P}} k_j$:
   a) $\forall j \in \mathcal{P}\backslash\{i\}$: if $\neg\mathsf{Z}_{\mathsf{Enc}}.\mathsf{Vf}(\pi_{k_j}, (\mathsf{clek}, \overline{k_i}))$, $\mathcal{P} := \mathcal{P}\backslash\{j\}$.
   b) $\overline{k} := \boxplus_{j\in\mathcal{P}} \overline{k_j}$.
3) $\mathsf{P}_i$ encrypts $g^{\gamma_i}$, and generates LHE of $x_i k$ and $\gamma_i k$:
   a) $\overline{x_i k} := x_i \boxdot^* \overline{k}$, $\mathsf{Z}_{\mathsf{DL\text{-}CL}}.\mathsf{P}((X_i, \overline{k}, \overline{x_i k}); x_i) \to \pi_{x_i k}$.
   b) $\gamma_i \leftarrow\$ \mathbb{F}_q$, $\mathsf{t\text{-}ElG.Enc}(\mathsf{elek}, g^{\gamma_i}; r_{\gamma_i}) \to \overline{g^{\gamma_i}}$.
   c) $\overline{\gamma_i k} := \gamma_i \boxdot^* \overline{k}$.
   d) $\mathsf{Z}_{\mathsf{El\text{-}CL}}.\mathsf{P}(g, \overline{g^{\gamma_i}}, \mathsf{elek}, \overline{k}, \overline{\gamma_i k}); (\gamma_i, r_{\gamma_i})) \to \pi_{\gamma_i k}$.
   e) Broadcast $(\overline{x_i k}, \pi_{x_i k}, \overline{g^{\gamma_i}}, \overline{\gamma_i k}, \pi_{\gamma_i k})$.
4) $\mathsf{P}_i$ computes homomorphically encryption of $xk := \sum_{j\in\mathcal{P}} L_{j,\mathcal{P}} x_j k$, $\gamma k := \sum_{j\in\mathcal{P}} \gamma_j k$, and $g^\gamma := \prod_{j\in\mathcal{P}} g^{\gamma_j}$:
   a) $\forall j \in \mathcal{P} \setminus \{i\}$, set $\mathcal{P} := \mathcal{P} \setminus \{j\}$ if
      - $\mathsf{Z}_{\mathsf{DL\text{-}CL}}.\mathsf{Vf}(\pi_{x_j k}, (X_j, \overline{k}, \overline{x_j k})) = 0$ or
      - $\mathsf{Z}_{\mathsf{El\text{-}CL}}.\mathsf{Vf}(\pi_{\gamma_j k}, (g, \overline{g^{\gamma_j}}, \mathsf{elek}, \overline{k}, \overline{\gamma_j k})) = 0$.
   b) Set $\overline{xk} := \boxplus_{j\in\mathcal{P}} (L_{j,\mathcal{P}} \boxdot \overline{x_j k})$, $\overline{\gamma k} := \boxplus_{j\in\mathcal{P}} \overline{\gamma_j k}$, and $\overline{g^\gamma} := \boxtimes_{j\in\mathcal{P}} \overline{g^{\gamma_j}}$ (ciphertext of $\prod_{j\in\mathcal{P}} g^{\gamma_j}$).
5) $\mathsf{P}_i$ partially decrypts $\overline{\gamma k}$ and $\overline{g^\gamma}$:
   a) $\mathsf{t\text{-}CL.PartDec}(\mathsf{clek}_i, \overline{\gamma k}, \mathsf{cldk}_i) \to \mathsf{pd}_{\gamma k, i}$.
   b) $\mathsf{t\text{-}ElG.PartDec}(\mathsf{elek}_i, \overline{g^\gamma}, \mathsf{eldk}_i) \to \mathsf{pd}_{g^\gamma, i}$.
   c) Broadcast $(\mathsf{pd}_{\gamma k, i}, \mathsf{pd}_{g^\gamma, i})$.
6) $\mathsf{P}_i$ fully decrypts $\overline{\gamma k}$ and $\overline{g^\gamma}$:
   a) $\mathsf{t\text{-}CL.FinDec}(\mathsf{clek}, \{\mathsf{clek}_j\}_{j\in\mathcal{P}}, \overline{\gamma k}, \{\mathsf{pd}_{\gamma k, j}\}_{j\in\mathcal{P}}) \to \gamma k$.
   b) $\mathsf{t\text{-}ElG.FinDec}(\mathsf{elek}, \{\mathsf{elek}_j\}_{j\in\mathcal{P}}, \overline{g^\gamma}, \{\mathsf{pd}_{g^\gamma, j}\}_{j\in\mathcal{P}}) \to g^\gamma$.
   c) $R := (g^\gamma)^{\frac{1}{k\gamma}}$, return $(R, \overline{k}, \overline{xk})$.

The offline phase is 3-round. Messages are broadcasted at the end of Steps 1, 3, and 5. Steps 2, 4, and 6 start when a sufficient number of messages are received.

The online signing phase is non-interactive, broadcasts happen only at the end of the second last step, and goes by:

Step 1: generate encrypted signature $km + rkx$;
Step 2-3: distributively extract (threshold-decrypt) $km + rkx$.

1) $\mathsf{P}_i$ homomorphically derives the encryption of $km + rkx$: Let the x-coordinate of $R$ be $r$ and $m$ be the hash of message msg, set $\overline{km + rkx} := (m \boxdot \overline{k}) \boxplus (r \boxdot \overline{xk})$.
2) $\mathsf{P}_i$ partially decrypts $\overline{km + rkx}$: $\mathsf{t\text{-}CL.PartDec}(\mathsf{clek}_i, \overline{km + rkx}, \mathsf{cldk}_i) \to \mathsf{pd}_{km+rkx,i}$, broadcast $\mathsf{pd}_{km+rkx,i}$.
3) $\mathsf{P}_i$ fully decrypts $\overline{km + rkx}$: $\mathsf{t\text{-}CL.FinDec}(\mathsf{clek}, \{\mathsf{clek}_j\}_{j\in\mathcal{P}}, \overline{km + rkx}, \mathsf{pd}_{km+rkx,j}) \to km + rkx$, return $(r, s := km + rkx)$.

We can further optimize the procedure for enhanced efficiency. Specifically, since users can verify the ECDSA signature using the signature verification algorithm, we only conduct partial decryption verification when ECDSA verification fails.

**Theorem 1.** *If* DKG-DL, DKG-CL, t-CL, *and* t-ElG *are correct, our threshold ECDSA is correct. If* $\mathsf{Z}_{\mathsf{Enc}}$, $\mathsf{Z}_{\mathsf{DL\text{-}CL}}$, *and* $\mathsf{Z}_{\mathsf{El\text{-}CL}}$ *are sound, our threshold ECDSA is robust (with* $<t$ *corrupted parties and an honest majority).*

Parties compute the signature over ciphertexts. By ensuring the ciphertext validity through ZKPs, the signature computation aligns with the correctness of TLHE. Robustness is guaranteed by the robust threshold decryption process. The proof details are available in Appendix F-A.

**Theorem 2.** *If* DKG-DL *and* DKG-CL *satisfy secrecy, and* t-CL *and* t-ElG *satisfy simulation security and* t-ind-cpa*, our threshold signature protocol for ECDSA is simulatable.*

At a high level, there exists a simulator on input $X$, msg, and the public output $(R, s)$ of ECDSA, generates an indistinguishable view of the protocol that outputs $(R, s)$. The transcript only contains ciphertexts (with ZKPs) and partial decryption. The ciphertexts leak nothing about the intermediate values, while the property described in Section V-C allows the simulator to decrypt the ciphertexts as any desired values safely, in this case $(R, s)$. Proof details are in Appendix F-B.

### C. Threshold BBS+ Signatures

BBS+ signatures sign on vectors of $\ell$ messages. The public parameter pp contains the bilinear group description $(\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, q, \mathsf{e})$ and $\ell + 1$ random group elements $(h_0, \dots h_\ell) \leftarrow\$ \mathbb{G}_1^{\ell+1}$. BBS+ algorithms $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ are:

- $\mathsf{KGen}(\mathsf{pp}) \to (\mathsf{vk}, \mathsf{sk})$: Randomly sample $x \leftarrow\$ \mathbb{F}_q$, output $\mathsf{vk} := (X := g_2^x)$ and $\mathsf{sk} := x$.
- $\mathsf{Sign}(\mathsf{sk} = x, (m_1, \dots, m_\ell)) \to \sigma$: Randomly sample $e, s \leftarrow\$ \mathbb{F}_q$, output $\sigma := (A := (g_1 h_0^s \prod_{k=1}^{\ell} h_k^{m_k})^{\frac{1}{x+e}}, e, s)$.
- $\mathsf{Vf}(\mathsf{vk}, \sigma, (m_1, \dots, m_\ell)) \to 0/1$: Parse vk as $X$ and $\sigma$ as $(A, e, s)$, accept if $\mathsf{e}(A, X g_2^e) = \mathsf{e}(g_1 h_0^s \prod_{k=1}^{\ell} h_k^{m_k}, g_2)$.

We present a threshold BBS+ construction that works similarly to our threshold ECDSA. We use threshold ElGamal encryption t-ElG to commit $\mathbb{G}_1$ elements to cater rushing adversaries. Our setup runs DKG-CL to generate clek for t-CL, but DKG-DL over $\mathbb{G}_2$ to generate $\mathsf{vk} \in \mathbb{G}_2$ for threshold BBS+, and DKG-DL over $\mathbb{G}_1$ to generate $\mathsf{elek} \in \mathbb{G}_1$ for t-ElG. $(h_0, \dots, h_\ell)$ are distributively generated (like the setup of Pedersen commitment) to hide their discrete logarithms.

We can prepare encryption of $x$ during the revelation phase of DKG, verifying against $X_i := g_2^{x_i}$.

1) $\mathsf{P}_i$ runs $\mathsf{t\text{-}CL.Enc}(\mathsf{clek}, x_i; r_{x_i}) \to \overline{x_i}$ and then $\mathsf{Z}_{\mathsf{Enc\text{-}DL}}.\mathsf{P}((\overline{x_i}, X_i); (x_i, r_{x_i})) \to \pi_{x_i}$.
2) $\forall j \in \mathcal{P} \setminus \{i\}$, if $\mathsf{Z}_{\mathsf{Enc\text{-}DL}}.\mathsf{Vf}(\pi_{x_j}, (\overline{x_j}, X_i)) = 0$, set $\mathcal{P} := \mathcal{P} \setminus \{j\}$. After that, set $\overline{x} := \boxplus_{j\in\mathcal{P}}(L_{j,\mathcal{P}} \boxdot \overline{x_j})$.

Our threshold BBS+ signing follows the skeleton below.

Step 1-2 distributively generate encrypted $e$, $s$, and $x+e$;
Step 3-4 distributively extract (threshold-decrypt) $e$ and $s$;
Step 5-6 distributively generate encrypted $D^\gamma$ where $D := g_1 h_0^s \prod_{k=1}^{\ell} h_k^{m_k}$ and $\gamma(x+e)$ from encrypted $x+e$;
Step 7-8 distributively extract (threshold-decrypt) $D^\gamma$ and $\gamma(x+e)$ to obtain $A = (D^\gamma)^{\frac{1}{\gamma(x+e)}}$.

The protocol is 4-round, and messages are broadcasted at the end of Steps 1, 3, 5, and 7. Steps 2, 4, 6, and 8 start when a sufficient number of messages are received.

1) $\mathsf{P}_i$ encrypts $e_i$ and $s_i$:
   a) $e_i \leftarrow\$ \mathbb{F}_q$, $\mathsf{t\text{-}CL.Enc}(\mathsf{clek}, e_i; r_{e_i}) \to \overline{e_i}$, $s_i \leftarrow\$ \mathbb{F}_q$, $\mathsf{t\text{-}CL.Enc}(\mathsf{clek}, s_i; r_{s_i}) \to \overline{s_i}$.
   b) $\mathsf{Z}_{\mathsf{Enc}}.\mathsf{P}(\overline{e_i}; (e_i, r_{e_i})) \to \pi_{e_i}$, $\mathsf{Z}_{\mathsf{Enc}}.\mathsf{P}(\overline{s_i}; (s_i, r_{s_i})) \to \pi_{s_i}$.
   c) Broadcast $(\overline{e_i}, \overline{s_i}, \pi_{e_i}, \pi_{s_i})$.
2) $\mathsf{P}_i$ generates LHE of $e := \sum_{j\in\mathcal{P}} e_j$, $s := \sum_{j\in\mathcal{P}} s_j$, $x+e$.
   a) $\forall j \in \mathcal{P} \setminus \{i\}$, set $\mathcal{P} := \mathcal{P} \setminus \{j\}$ if $\mathsf{Z}_{\mathsf{Enc}}.\mathsf{Vf}(\pi_{e_j}, \overline{e_j}) = 0$ or $\mathsf{Z}_{\mathsf{Enc}}.\mathsf{Vf}(\pi_{s_j}, \overline{s_j}) = 0$.
   b) Set $\overline{e} := \boxplus_{j\in\mathcal{P}}\overline{e_j}$, $\overline{s} := \boxplus_{j\in\mathcal{P}}\overline{s_j}$, $\overline{x+e} := \overline{x} \boxplus \overline{e}$.
3) $\mathsf{P}_i$ partially decrypts $e$ and $s$:
   a) $\mathsf{t\text{-}CL.PartDec}(\mathsf{clek}_i, \overline{e}, \mathsf{cldk}_i) \to \mathsf{pd}_{e,i}$. $\mathsf{t\text{-}CL.PartDec}(\mathsf{clek}_i, \overline{s}, \mathsf{cldk}_i) \to \mathsf{pd}_{s,i}$.
   b) Broadcast $(\mathsf{pd}_{e,i}, \mathsf{pd}_{s,i})$.
4) $\mathsf{P}_i$ fully decrypts $e$ and $s$:
   a) $\mathsf{t\text{-}CL.FinDec}(\mathsf{clek}, \{\mathsf{clek}_j\}_{j\in\mathcal{P}}, \overline{e}, \{\mathsf{pd}_{e,j}\}_{j\in\mathcal{P}}) \to e$.
   b) $\mathsf{t\text{-}CL.FinDec}(\mathsf{clek}, \{\mathsf{clek}_j\}_{j\in\mathcal{P}}, \overline{s}, \{\mathsf{pd}_{s,j}\}_{j\in\mathcal{P}}) \to s$.
5) $\mathsf{P}_i$ generates LHE of $(g_1 h_0^s \prod_{k=1}^{\ell} h_k^{m_k})^{\gamma_i}$ and $\gamma_i(x+e)$.
   a) $\gamma_i \leftarrow\$ \mathbb{F}_q$, $\mathsf{t\text{-}ElG.Enc}(\mathsf{elek}, D^{\gamma_i}; r_{\gamma_i}) \to \overline{D^{\gamma_i}}$.
   b) $\overline{\gamma_i(x+e)} := \gamma_i \boxdot^* \overline{x+e}$.
   c) $\mathsf{Z}_{\mathsf{El\text{-}CL}}.\mathsf{P}(D, \overline{D^{\gamma_i}}, \mathsf{elek}, \overline{x+e}, \overline{\gamma_i(x+e)}); (\gamma_i, r_{\gamma_i})) \to \pi_{\gamma_i}$.
   d) Broadcast $(\overline{D^{\gamma_i}}, \overline{\gamma_i(x+e)}, \pi_{\gamma_i})$.
6) $\mathsf{P}_i$ uses homomorphic operation to generate LHE of $D^\gamma := \prod_{j\in\mathcal{P}} D^{\gamma_j}$ and $\gamma(x+e) := \sum_{j\in\mathcal{P}} \gamma_j(x+e)$.
   a) $\forall j \in \mathcal{P} \setminus \{i\}$: set $\mathcal{P} := \mathcal{P} \setminus \{j\}$ if $\neg\mathsf{Z}_{\mathsf{El\text{-}CL}}.\mathsf{Vf}(\pi_{\gamma_i}, (D, \overline{D^{\gamma_i}}, \mathsf{elek}, \overline{x+e}, \overline{\gamma_i(x+e)}))$.
   b) $\overline{\gamma(x+e)} := \boxplus_{j\in\mathcal{P}}\overline{\gamma_j(x+e)}$, $D^\gamma := \boxtimes_{j\in\mathcal{P}}\overline{D^{\gamma_j}}$.
7) $\mathsf{P}_i$ partially decrypts $D^\gamma$ and $\gamma(x+e)$:
   a) $\mathsf{t\text{-}ElG.PartDec}(\mathsf{elek}_i, \overline{D^\gamma}, \mathsf{eldk}_i) \to \mathsf{pd}_{D^\gamma,i}$. $\mathsf{t\text{-}CL.PartDec}(\mathsf{clek}_i, \overline{\gamma(x+e)}, \mathsf{cldk}_i) \to \mathsf{pd}_{\gamma(x+e),i}$.
   b) Broadcast $(\mathsf{pd}_{D^\gamma,i}, \mathsf{pd}_{\gamma(x+e),i})$.
8) $\mathsf{P}_i$ fully decrypts $D^\gamma$ and $\gamma(x+e)$:
   a) $\mathsf{t\text{-}ElG.FinDec}(\mathsf{elek}, \{\mathsf{elek}_j\}, \overline{D^\gamma}, \{\mathsf{pd}_{D^\gamma,j}\}_{j\in\mathcal{P}}) \to D^\gamma$.
   b) $\mathsf{t\text{-}CL.FinDec}(\mathsf{clek}, \{\mathsf{clek}_j\}_{j\in\mathcal{P}}, \overline{\gamma(x+e)}, \{\mathsf{pd}_{\gamma(x+e),j}\}_{j\in\mathcal{P}}) \to \gamma(x+e)$.
   c) Return $(A, e, s) := ((D^\gamma)^{\frac{1}{\gamma(x+e)}}, e, s)$.

**Theorem 3.** *If* DKG-DL, DKG-CL, t-CL, *and* t-ElG *are correct, so does our threshold BBS+. If* $\mathsf{Z}_{\mathsf{Enc}}$ *and* $\mathsf{Z}_{\mathsf{El\text{-}CL}}$ *are sound, our threshold BBS+ signing protocol is robust (with* $<t$ *corrupted parties and an honest majority).*

**Theorem 4.** *If* DKG-DL *and* DKG-CL *satisfy secrecy, and* t-CL *and* t-ElG *satisfy simulation security and* t-ind-cpa*, our threshold signature protocol for BBS+ is simulatable.*

Both proofs employ strategy as the ones used for threshold ECDSA. Details can be found in Appendices F-C and F-D.

## VII. Experiments

We implement our constructions using `BICYCL` [9], an open-source C++ library for class-group arithmetic. We use the class group $\mathsf{CL\text{-}HSM_q}$ with a 256-bit plaintext space and 1827-bit $\Delta_K$ to achieve a 128-bit security level. SHA-3 is used to instantiate $\mathsf{H}(\cdot)$ for the Fiat–Shamir transform. Experiments were run 100 times for reporting averages on a desktop computer[11] with AMD Ryzen 5 2600 CPU and 64GB RAM.

In line with the state of the art of Wong *et al.* [48], we opt for $n = t-1$ to compare computational costs with another CL-encryption-based work [16]. This choice represents our worst-case scenario for utilizing threshold decryption (unnecessary in existing approaches) with $O(t)$-exponentiation.

With no multi-thread optimization, Figure 3 illustrates the runtime and (average) incoming communication costs of the DKG protocols over different numbers of parties $n \in \{5, 10, 15, 20\}$. The revelation phase of DKG-CL (DKG-DL) has a constant-size communication cost of 1.31 (0.77) KBytes per party. The key generation phase of DKG-CL (Figure 2) is the most time-consuming part and its runtime in Gen and GenVf scales with $n$len and $n^2$len, respectively (*cf.*, Figure 3a).

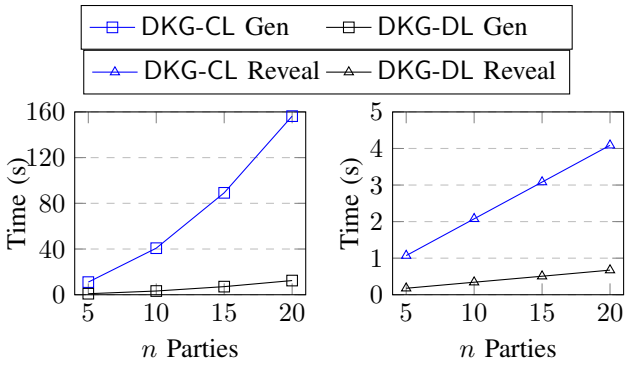
For threshold CL encryption t-CL, PartDec takes ∼93 ms, and each party broadcasts 0.8 KBytes. FinDec scales with $n$ with runtime 443 ms for $n = 5$ and 1704 ms for $n = 20$. Homomorphic evaluation of t-CL follows the basic CLE scheme, costing 24ms (or <1 ms if ciphertext re-randomization is not needed). The t-CL.KGen protocol is essentially the DKG-CL protocol that only runs once. This is the tradeoff for efficiency gain in online phases.

We implement the t-CL-based threshold ECDSA protocol on the secp256k1 elliptic curve. For homomorphic evaluations of t-CL ciphertext, we employ ciphertext re-randomization for publicly-computable ciphertexts involved in Z (otherwise, there will be inconsistencies). Figures 4a and 4b compare schemes using CLE with $t = n$. Compared to Castagnos *et al.* [16] with identifiable abort, ours saves ∼20% in total communication cost in the offline stage. Compared to Wong *et al.* [48] (with weaker "self-healing"), we save over half of the runtime and total communication costs. The offline communication cost of Paillier-based threshold ECDSA [11] is 153 KBytes and 613 KBytes for $n = 5$ and 20, which is at least twice compared to the CLE-based schemes. Encryption/decryption of CLE is also faster than the Paillier cryptosystem for 128-bit security level or above [9]. With the proposed t-CL, our robust threshold ECDSA protocols can be more efficient.

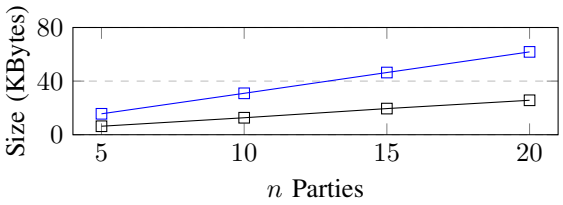
We remark that the OT-based instantiation[12] of Gągol *et al.* [35] only requires half our running time but incurs

[11] We save/load the protocol outputs/inputs on a solid state drive and sequentially ran as different parties. The saving and loading times are neglected.

[12] This is adapted from the Golang library available at https://gitlab.com/alephledger/threshold-ecdsa.

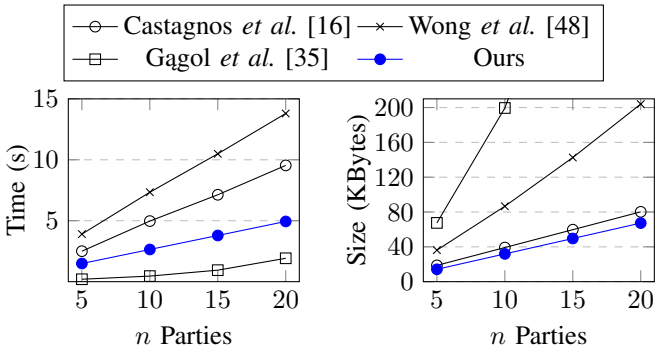



(a) Runtime per party (b) Runtime per party




(c) Communication cost per party

Fig. 3: Total runtime and (incoming) communication for the DKG protocols, including the verification




(a) Runtime per party (b) Communication per party

Fig. 4: Total runtime and (incoming) communication of the offline stage (excluding setup) of state-of-the-art threshold ECDSA protocols given $n$ parties participate

higher communication costs, *e.g.*, ∼400 KBytes for 15 parties and ∼700 KBytes for 20 parties.

## VIII. Conclusion and Future Works

We put forth 2-round robust DKG protocols for discrete-logarithm-based and class-group-based cryptosystems. We showcase efficient threshold signing protocols for ECDSA and BBS+ signatures using the threshold CL encryption with our DKG. The resulting protocol provides identifiable abort and robustness for a sufficient number of honest parties. Our empirical results show that it saves ∼50% of computational and communication costs compared to the threshold ECDSA by Wong *et al.* [48] with the same level of security guarantee. In summary, we close 2-out-of-3 open problems they left [48].

One of the future directions includes upgrading our scheme with UC security. In our scheme, all ZKPs do not need rewinding, except the one for discrete logarithm over class groups during the setup phase. This minimizes the number of expensive straight-line extractors, making UC security attainable. Another future direction is to simplify DKG for class groups, the most costly part of our scheme. If a biased key can be used in threshold CL encryption, we wonder if a one-round DKG for threshold CL encryption (with UC security) is achievable. This way, the computational cost can be minimized by enabling generation in an asynchronous sense.

Our schemes allow the signer to assume an "unstable" state and exit the protocol anytime. Nevertheless, the scheme operates on stable broadcast channels, achievable through a public bulletin board, to reach consensus on a shared ciphertext. We posit that this form of consensus is more readily attainable than existing proposals relying on a conceptual agreement for a shared secret. Exploring weaker communication assumptions is a potential avenue to enhance practicality.

## References


[1] D. Abram, I. Damgård, C. Orlandi, and P. Scholl, "An algebraic framework for silent preprocessing with trustless setup and active security," in *CRYPTO Part IV*, 2022, pp. 421–452.

[2] D. Abram, A. Nof, C. Orlandi, P. Scholl, and O. Shlomovits, "Low-bandwidth threshold ECDSA via pseudorandom correlation generators," in *S&P*, 2022, pp. 2554–2572.

[3] S. Atapoor, K. Baghery, D. Cozzo, and R. Pedersen, "VSS from distributed ZK proofs and applications," in *ASIACRYPT Part I*, 2023, pp. 405–440.

[4] M. H. Au, W. Susilo, Y. Mu, and S. S. M. Chow, "Constant-size dynamic $k$-times anonymous authentication," *IEEE Syst. J.*, vol. 7, no. 2, pp. 249–261, 2013.

[5] E. Bangerter, J. Camenisch, and S. Krenn, "Efficiency limitations for $\Sigma$-protocols for group homomorphisms," in *TCC*, 2010, pp. 553–571.

[6] D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. R. Rasmussen, and A. Sahai, "Threshold cryptosystems from threshold fully homomorphic encryption," in *CRYPTO Part I*, 2018, pp. 565–596.

[7] W. Bosma and P. Stevenhagen, "On the computation of quadratic 2-class groups," *Journal de théorie des nombres de Bordeaux*, vol. 8, no. 2, pp. 283–313, 1996.

[8] A. Bouez and K. Singh, "One round threshold ECDSA without roll call," in *Cryptographers' Track at the RSA Conf.*, 2023, pp. 389–414.

[9] C. Bouvier, G. Castagnos, L. Imbert, and F. Laguillaumie, "I want to ride my BICYCL: BICYCL Implements CryptographY in CLass groups," *J. Cryptol.*, vol. 36, no. 3, p. 17, 2023.

[10] L. Braun, I. Damgård, and C. Orlandi, "Secure multiparty computation from threshold encryption based on class groups," in *CRYPTO Part I*, 2023, pp. 613–645.

[11] R. Canetti, R. Gennaro, S. Goldfeder, N. Makriyannis, and U. Peled, "UC non-interactive, proactive, threshold ECDSA with identifiable aborts," in *CCS*, 2020, pp. 1769–1787.

[12] I. Cascudo and B. David, "SCRAPE: scalable randomness attested by public entities," in *App. Crypto. & Net. Sec. (ACNS)*, 2017, pp. 537–556.

[13] ——, "ALBATROSS: Publicly attestable batched randomness based on secret sharing," in *ASIACRYPT Part III*, 2020, pp. 311–341.

[14] ——, "Publicly verifiable secret sharing over class groups and applications to DKG and YOSO," Cryptol. ePrint Arch. 2023/1651, 2023.

[15] G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker, "Bandwidth-efficient threshold EC-DSA," in *PKC*, 2020, pp. 266–296.

[16] ——, "Bandwidth-efficient threshold EC-DSA revisited: Online/offline extensions, identifiable aborts proactive and adaptive security," *Theor. Comput. Sci.*, vol. 939, pp. 78–104, 2023.

[17] G. Castagnos and F. Laguillaumie, "Linearly homomorphic encryption from DDH," in *CT-RSA*, 2015, pp. 487–505.

[18] G. Castagnos, F. Laguillaumie, and I. Tucker, "Practical fully secure unrestricted inner product functional encryption modulo $p$," in *ASIACRYPT Part II*, 2018, pp. 733–764.

[19] ——, "Threshold linearly homomorphic encryption on $\mathbf{Z}/2^k\mathbf{Z}$," in *ASIACRYPT Part II*, 2022, pp. 99–129.

[20] S. S. M. Chow, J. P. K. Ma, and T. H. Yuen, "Scored anonymous credentials," in *ACNS Part II*. Springer, 2023, pp. 484–515.

[21] G. Couteau, M. Klooß, H. Lin, and M. Reichle, "Efficient range proofs with transparent setup from bounded integer commitments," in *EUROCRYPT Part III*, 2021, pp. 247–277.

[22] R. Cramer, I. Damgård, and J. B. Nielsen, "Multiparty computation from threshold homomorphic encryption," in *EUROCRYPT*, 2001, pp. 280–299.

[23] R. Cramer and S. Fehr, "Optimal black-box secret sharing over arbitrary abelian groups," in *CRYPTO*, 2002, pp. 272–287.

[24] A. P. K. Dalskov, C. Orlandi, M. Keller, K. Shrishak, and H. Schulmann, "Securing DNSSEC keys via threshold ECDSA from generic MPC," in *ESORICS Part II*, 2020, pp. 654–673.

[25] I. Damgård and M. Jurik, "A generalisation, a simplification and some applications of Paillier's probabilistic public-key system," in *Public Key Cryptography (PKC)*, 2001, pp. 119–136.

[26] J. Doerner, Y. Kondi, E. Lee, A. shelat, and L. Tyner, "Threshold BBS+ signatures for distributed anonymous credential issuance," in *S&P*, 2023, pp. 773–789.

[27] J. Doerner, Y. Kondi, E. Lee, and A. shelat, "Secure two-party threshold ECDSA from ECDSA assumptions," in *S&P*, 2018, pp. 980–997.

[28] ——, "Threshold ECDSA from ECDSA assumptions: The multiparty case," in *S&P*, 2019, pp. 1051–1066.

[29] P. Fouque, G. Poupard, and J. Stern, "Sharing decryption in the context of voting or lotteries," in *Financial Cryptography*, 2000, pp. 90–104.

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

[31] R. Gennaro and S. Goldfeder, "Fast multiparty threshold ECDSA with fast trustless setup," in *CCS*, 2018, pp. 1179–1194.

[32] ——, "One round threshold ECDSA with identifiable abort," IACR Cryptol. ePrint Arch. 2020/540, 2020.

[33] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Robust threshold DSS signatures," in *EUROCRYPT*, 1996, pp. 354–371.

[34] ——, "Secure distributed key generation for discrete-log based cryptosystems," in *EUROCRYPT*, 1999, pp. 295–310.

[35] A. Gągol, J. Kula, D. Straszak, and M. Świętek, "Threshold ECDSA for decentralized asset custody," Cryptol. ePrint Arch. 2020/498, 2020.

[36] J. Groth, "Non-interactive distributed key generation and key resharing," IACR Cryptol. ePrint Arch. 2021/339, 2021.

[37] J. Groth and V. Shoup, "Fast batched asynchronous distributed key generation," Cryptol. ePrint Arch. 2023/1175, 2023.

[38] B. Kachouh, L. Sliman, A. E. Samhat, and K. Barkaoui, "Demystifying threshold elliptic curve digital signature algorithm for multiparty applications," in *ACSW*, 2023, pp. 112–121.

[39] A. Kate, E. V. Mangipudi, P. Mukherjee, H. Saleem, and S. A. K. Thyagarajan, "Non-interactive VSS using class groups and application to DKG," Cryptol. ePrint Arch. 2023/451, 2023.

[40] J. Katz, "Round optimal robust distributed key generation," Cryptol. ePrint 2023/1094, 2023.

[41] Y. Lindell and A. Nof, "Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody," in *CCS*, 2018, pp. 1837–1854.

[42] J. P. K. Ma and S. S. M. Chow, "SMART credentials in the multi-queue of slackness (or Secure management of anonymous reputation traits without global halting)," in *EuroS&P*, 2023, pp. 896–912.

[43] L. K. L. Ng, S. S. M. Chow, D. P. H. Wong, and A. P. Y. Woo, "LDSP: shopping with cryptocurrency privately and quickly under leadership," in *ICDCS*, 2021, pp. 261–271.

[44] T. Rabin, "A simplified approach to threshold and proactive RSA," in *CRYPTO*, 1998, pp. 89–104.

[45] V. Shoup, "Practical threshold signatures," in *EUROCRYPT*, 2000, pp. 207–220.

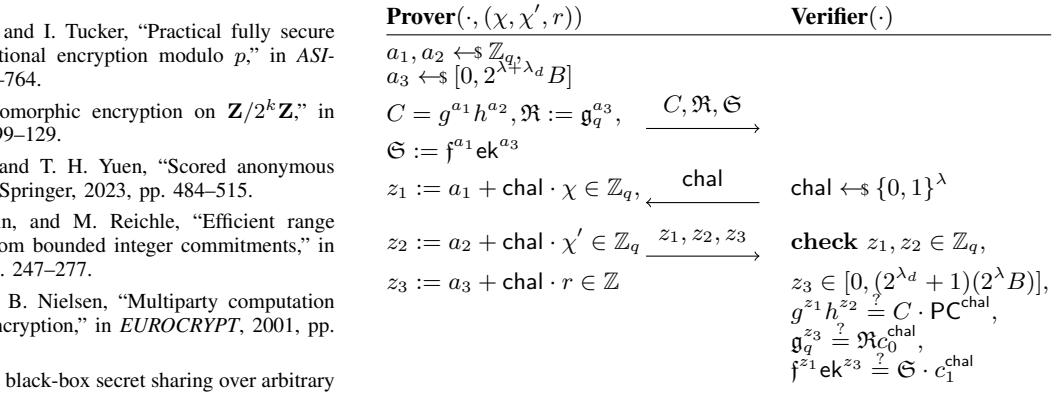




Fig. 5: $\Sigma$-Protocol for $\mathcal{R}_{\mathsf{Enc\text{-}PC}}$ over $(\mathsf{PC}, (c_0, c_1), \mathsf{ek})$


[46] B. Terelius and D. Wikström, "Efficiency limitations of $\Sigma$-protocols for group homomorphisms revisited," in *SCN*, 2012, pp. 461–476.

[47] H. W. H. Wong, J. P. K. Ma, H. H. F. Yin, and S. S. M. Chow, "How (not) to build threshold EdDSA," in *RAID*, 2023, pp. 123–134.

[48] H. W. H. Wong, J. P. K. Ma, H. Y. F. Yin, and S. S. M. Chow, "Real threshold ECDSA," in *NDSS*, 2023.

[49] T. H. Yuen, H. Cui, and X. Xie, "Compact zero-knowledge proofs for threshold ECDSA with trustless setup," in *PKC I*, 2021, pp. 481–511.

[50] H. Zhang, S. Duan, C. Liu, B. Zhao, X. Meng, S. Liu, Y. Yu, F. Zhang, and L. Zhu, "Practical asynchronous distributed key generation: Improved efficiency, weaker assumption, and standard model," in *DSN*, 2023, pp. 568–581.


## APPENDIX A
## DETAIL IN ZKP

### A. Non-interactive ZKP

The zero-knowledge proof requires standard soundness: If $x \notin \mathcal{L}$, no cheating prover can convince an honest verifier that the statement $x$ is true with non-negligible probability.

Zero-knowledge proof of knowledge guarantees that the prover knows a witness $w$ such that $(x, w) \in \mathcal{R}$. More formally, the standard soundness is replaced by special soundness, *i.e.*, there exists a PPT extractor Ext that can extract a witness $w$ with $(x, w) \in \mathcal{R}$ from multiples accepting transcripts.

### B. ZKP Relations

We provide the $\Sigma$-protocol for the following NP-relations. All only require standard soundness except $\mathcal{R}_{\mathsf{CL}}$. The relation in class groups follows the argument in known order groups. The key point for standard soundness is to show the existence of a witness but without explicitly computing it. For special soundness, taking discrete logarithm over class groups as an example, the extractor obtains $(\alpha, \beta)$ such that $\mathfrak{X}^\beta = \mathfrak{g}_q^\alpha$. However, computing $x = \alpha/\beta$ becomes challenging when $\alpha/\beta \notin \mathbb{Z}$ due to the unknown order. Given that the exponent space of class groups allows for the division of a power of 2 without knowing the order [7], the 2-fractional-root assumption [21] assumes that the denominator $(\frac{\beta}{\gcd(\alpha,\beta)})$ of the simplified fraction of $\alpha/\beta$ is not a power of 2 with negligible probability. As a result, this assumption guarantees the successful extraction of the witness by ensuring the validity of division.

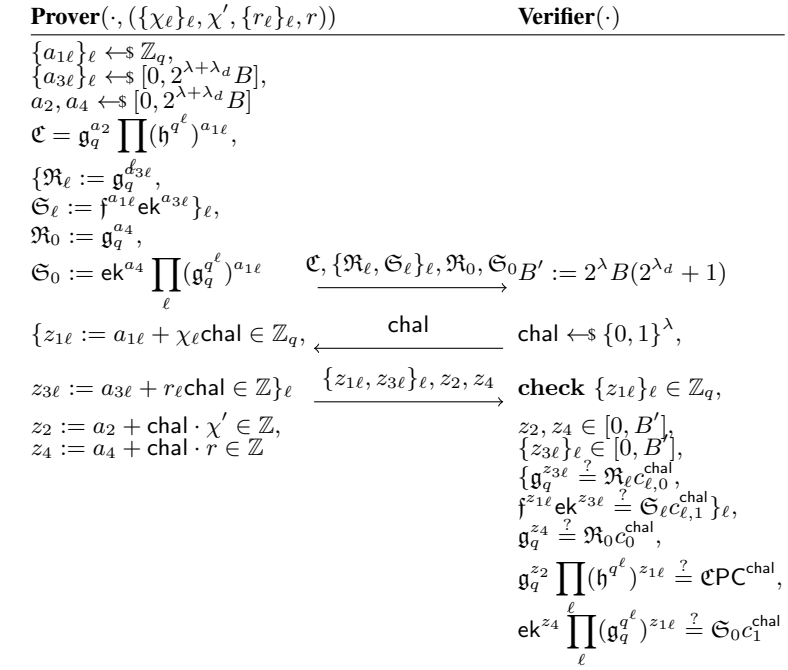



Fig. 6: $\Sigma$-Protocol for $\mathcal{R}_{\mathsf{BInt}}$ over $(\mathsf{PC}, \{c_{\ell,0}, c_{\ell,1}\}_\ell, (c_0, c_1), \mathsf{ek})$

*a) Discrete Logarithm over Class Groups:* For $\mathfrak{X} \in \mathfrak{G}^q$, $\mathcal{R}_{\mathsf{CL}} = \{(\mathfrak{X}, x) : \mathfrak{X} = \mathfrak{g}_q^x\}$ verifies that $\mathfrak{X} = \mathfrak{g}_q^x$.

*b) CL Ciphertext and Commitment over $\mathbb{G}$:* The following relation verifies that $(\chi, \chi')$ is the opening of commitment $\mathsf{PC}$ and $c = (c_0, c_1)$ is generated by $\mathsf{CLE.Enc}(\mathsf{ek}, \chi; r)$.

$$\mathcal{R}_{\mathsf{Enc\text{-}PC}} = \{((\mathsf{PC}, (c_0, c_1), \mathsf{ek}), (\chi, \chi', r)) : \mathsf{PC} = g^\chi h^{\chi'} \wedge c_0 = \mathfrak{g}_q^r \wedge c_1 = \mathfrak{f}^\chi \mathsf{ek}^r\}$$

*c) Decryption of CL Ciphertext and Discrete Logarithm over $\mathbb{G}$:* The following relation verifies that decrypting ciphertext $(c_0, c_1)$ with $\mathsf{dk}$ results in $x$ where $X = g^x$.

$$\mathcal{R}_{\mathsf{Dec\text{-}DL}} = \{((X, (c_0, c_1), \mathsf{ek}), (x, \mathsf{dk})) : c_1 = \mathfrak{f}^x c_0^{\mathsf{dk}} \wedge \mathsf{ek} = \mathfrak{g}_q^{\mathsf{dk}} \wedge X = g^x\}$$

One can prove Dec-DL via ZKP for Enc-PC by setting $\chi' = 0$ and rearranging: $\mathsf{Z}_{\mathsf{Dec\text{-}DL}}(X, (c_0, c_1), \mathsf{ek}; (x, \mathsf{dk})) := \mathsf{Z}_{\mathsf{Enc\text{-}PC}}((X, (\mathsf{ek}, c_1), c_0); (x, 0, \mathsf{dk}))$.

*d) Encryption and Commitment of Big Integers:* The relation below verifies that a big integer $\chi$, which is (1) committed in $\mathsf{PC}$, (2) encrypted via $\mathfrak{g}_q^\chi$ in $(c_0, c_1)$, and (3) decomposed as $\sum_\ell q^\ell \chi_\ell$, with each $\chi_\ell$ encrypted in $(c_{\ell,0}, c_{\ell,1})$.

$$\begin{aligned}\mathcal{R}_{\mathsf{BInt}} = \{((\mathsf{PC}, \{c_{\ell,0}, c_{\ell,1}\}_\ell, (c_0, c_1), \mathsf{ek}), (\{\chi_\ell\}_\ell, \chi', \{r_\ell\}_\ell, r)) : \\ \mathsf{PC} = \mathfrak{g}_q^{\chi'} \prod_\ell (\mathfrak{h}^{q^\ell})^{\chi_\ell} \\ \wedge \{c_{\ell,0} = \mathfrak{g}_q^{r_\ell} \wedge c_{\ell,1} = \mathfrak{f}^{\chi_\ell} \mathsf{ek}^{r_\ell}\}_\ell \\ \wedge c_0 = \mathfrak{g}_q^r \wedge c_1 = \mathsf{ek}^r \prod_\ell (\mathfrak{g}_q^{q^\ell})^{\chi_\ell}\}\end{aligned}$$

*e) Decryption of CL Encryption for Group Elements and Discrete Logarithm over Class Groups:* The following relation verifies that $x$ such that $\mathfrak{X} = (\mathfrak{g}_q^\Delta)^x$ and $(\mathfrak{g}_q)^x$ is the result of decrypting ciphertext $(c_0, c_1)$ using $\mathsf{dk}$.

$$\mathcal{R}_{\mathsf{GDec\text{-}CL}} = \{((\mathfrak{X}, (c_0, c_1), \mathsf{ek}, \mathfrak{g}_q^\Delta), (\mathsf{dk}, x)) : \mathfrak{X} = (\mathfrak{g}_q^\Delta)^x \wedge c_1 = \mathfrak{g}_q^x c_0^{\mathsf{dk}} \wedge \mathsf{ek} = \mathfrak{g}_q^{\mathsf{dk}}\}$$

*f) Correct Partial Decryption:* The relation below verifies that $\mathsf{cpd}_i$ is a partial decryption of $c_0$, *i.e.*, $\mathsf{cpd}_i = (c_0^\Delta)^{\mathsf{dk}_i}$.

$$\mathcal{R}_{\mathsf{PartDec}} = \{((\mathsf{ek}_i, c_0^\Delta, \mathsf{cpd}_i), \mathsf{dk}_i) : \mathsf{ek}_i = (\mathfrak{g}_q^\Delta)^{\mathsf{dk}_i} \wedge \mathsf{cpd}_i = (c_0^\Delta)^{\mathsf{dk}_i}\}$$

*g) Encryption:* The following relation verifies that $c = (c_0, c_1)$ is generated by $\mathsf{CLE.Enc}(\mathsf{ek}, \chi; r)$.

$$\mathcal{R}_{\mathsf{Enc}} = \{((c_0, c_1), (k, r)) : c_0 = (\mathfrak{g}_q^{\Delta^2})^r \wedge c_1 = \mathfrak{f}^k \mathsf{ek}^r\}$$

One can prove Enc via ZKP for Enc-PC by setting $\mathsf{PC} = 0$ and $\chi' = 0$: $\mathsf{Z}_{\mathsf{Enc}}((c_0, c_1); (x, r)) := \mathsf{Z}_{\mathsf{Enc\text{-}PC}}((0, (c_0, c_1), \mathsf{ek}); (x, 0, r))$.

*h) Multiplying CL Ciphertext with Discrete Logarithm over $\mathbb{G}$:* The relation below verifies that CL ciphertext $(c_{xk,0}, c_{xk,1})$ is a multiplication of $(c_{k,0}, c_{k,1})$ by $x$ in $X = g^x$.

$$\mathcal{R}_{\mathsf{DL\text{-}CL}} = \{((X, (c_{k,0}, c_{k,1}), (c_{xk,0}, c_{xk,1})), x) : X = g^x \wedge c_{xk,0} = c_{k,0}^x \wedge c_{xk,1} = c_{k,1}^x\}$$

*i) Multiplying CL Ciphertext with Plaintext of ElGamal Encryption:* The following relation verifies that the plaintext $\gamma$ of ElGamal ciphertext $(g^r, D^\gamma \mathsf{elek}^r)$ is used in homomorphic multiplication of CL ciphertext $(c_{k,0}, c_{k,1})$ to $(c_{\gamma k,0}, c_{\gamma k,1})$.

$$\mathcal{R}_{\mathsf{El\text{-}CL}} = \{((D, (c_0, c_1), \mathsf{elek}, (c_{k,0}, c_{k,1}), (c_{\gamma k,0}, c_{\gamma k,1})), (\gamma, r)) : c_0 = g^r \wedge c_1 = D^\gamma \mathsf{elek}^r \wedge c_{\gamma k,0} = c_{k,0}^\gamma \wedge c_{\gamma k,1} = c_{k,1}^\gamma\}$$

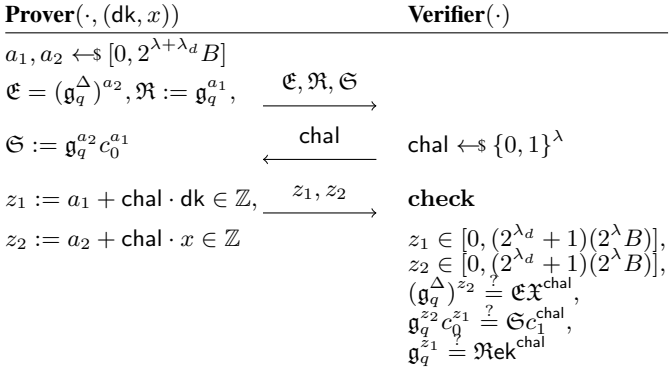



Fig. 7: $\Sigma$-Protocol for $\mathcal{R}_{\mathsf{GDec\text{-}CL}}$ over $(\mathfrak{X},(c_0,c_1),\mathsf{ek},\mathfrak{g}_q^{\Delta})$

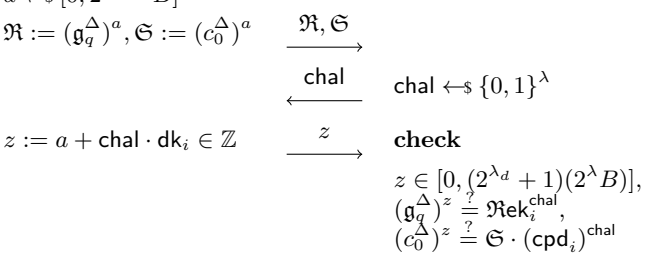



Fig. 8: $\Sigma$-Protocol for $\mathcal{R}_{\mathsf{PartDec}}$

One can prove DL-CL via ZKP for El-CL by setting $D=g$, $(c_0,c_1)=(0,X)$, $\mathsf{elek}=1$, $\gamma=x$, and $r=0$: $\mathsf{Z}_{\mathsf{DL\text{-}CL}}((X,(c_{k,0},c_{k,1}),(c_{xk,0},c_{xk,1})),x):=\mathsf{Z}_{\mathsf{El\text{-}CL}}((g,(0,X),1,(c_{k,0},c_{k,1}),(c_{xk,0},c_{xk,1})),(x,0))$.

*j) Correct Decryption of two CL Ciphertexts:* The following relation verifies that two ciphertexts $c$ and $c'$ can be correctly decrypted using $dk$.

$$\mathcal{R}_{\mathsf{2Dec}}=\{(((c_0,c_1),(c_0',c_1'),\mathsf{ek}),\mathsf{dk}):\\ c_1=c_0^{\mathsf{dk}}\wedge c_1'=c_0^{'\mathsf{dk}}\wedge\mathsf{ek}=\mathfrak{g}_q^{\mathsf{dk}}\}$$

One can prove $\mathcal{R}_{\mathsf{2Dec}}$ by running ZKP for $\mathcal{R}_{\mathsf{PartDec}}$ twice: $\mathcal{R}_{\mathsf{2Dec}}(((c_0,c_1),(c_0',c_1'),\mathsf{ek}),\mathsf{dk}):=\mathcal{R}_{\mathsf{PartDec}}((\mathsf{ek}^{\Delta},c_0,c_1^{\Delta}),\mathsf{dk})\circ\mathcal{R}_{\mathsf{PartDec}}((\mathsf{ek}^{\Delta},c_0',c_1'^{\Delta}),\mathsf{dk})$

*k) Discrete Logarithm and Commitment over Class Groups:* The following relation verifies that $(x,x')$ is the opening of commitment and $\mathfrak{X}=(\mathfrak{g}_q^{\Delta})$.

$$\mathcal{R}_{\mathsf{PC\text{-}DL}}=\{((\mathsf{PC},\mathfrak{X}),(x,x')):\mathsf{PC}=\mathfrak{g}_q^{x'}\mathfrak{h}^{x}\wedge\mathfrak{X}=(\mathfrak{g}_q^{\Delta})^{x}\}$$

Figures 5, 6, 7, 8, and 9 depict the $\Sigma$-protocols.

## Appendix B
## Security Analysis of DKG Protocols

We provide Lemmas 8 and 10 for correctness and Lemmas 9 and 11 for security. In our DKG-DL, we make the following changes: (1) Replacing the shares verification method, which involves polynomial evaluation in the DRG construction [48], with dual-code-based verification. (2) Substituting

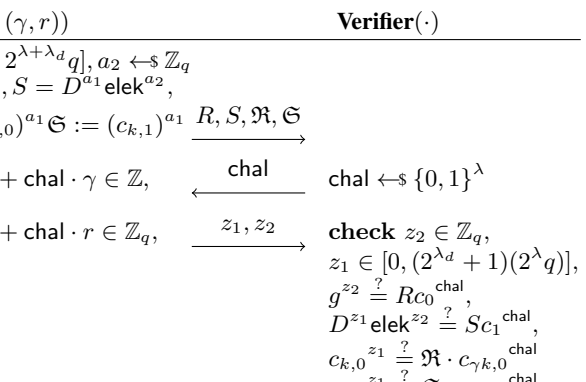



Fig. 9: $\Sigma$-Protocol for $\mathcal{R}_{\mathsf{El\text{-}CL}}$ over $(D,(c_0,c_1),\mathsf{elek},(c_{k,0},c_{k,1}),(c_{\gamma k,0},c_{\gamma k,1}))$

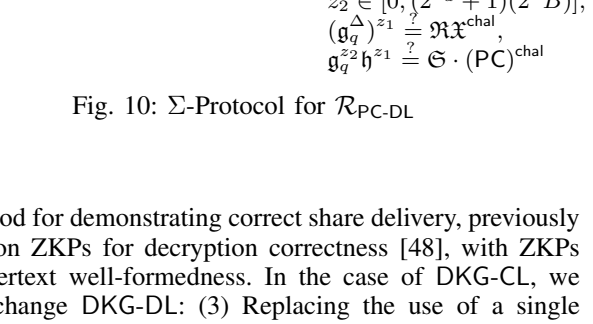



Fig. 10: $\Sigma$-Protocol for $\mathcal{R}_{\mathsf{PC\text{-}DL}}$

the method for demonstrating correct share delivery, previously relying on ZKPs for decryption correctness [48], with ZKPs for ciphertext well-formedness. In the case of DKG-CL, we further change DKG-DL: (3) Replacing the use of a single ciphertext for distributing a share in DKG-DL with the utilization of multiple ciphertexts. Despite these modifications, the verification of inconsistent shares passing with negligible probability after change (1) and the preservation of ciphertexts' resistance to information leakage after changes (2) and (3) ensure that the alterations do not disrupt the workflow of the existing correctness and security arguments of Wong *et al.* [48, Theorems 1 and 2]. Therefore, we only state the lemmas here and refer to the existing arguments [48].

**Lemma 8.** *If the Pedersen commitment is computationally binding and statistically hiding,* Z *is (standard)-sound and honest-verifier zero-knowledge,* SS *satisfies correctness, robustness, and unconditional secrecy, and* CLE *is correct and indistinguishable against adaptive chosen-plaintext attacks (*IND-CPA*-secure), then our* DKG-DL *is correct and robust (with $<t$ corrupted parties and an honest majority).*

**Lemma 9.** *If* Z *is honest-verifier zero-knowledge,* SS *has unconditional secrecy, Pedersen commitment is perfect hiding, and* CLE *is* IND-CPA*-secure, then our* DKG-DL *has secrecy.*

For DKG in class groups, we need a slight change the requirement on secrecy and correctness to match the reconstruction result from $\mathbb{Z}$-SS, *i.e.*, $\Delta^2 x=\sum_{i\in\mathcal{P}'}(\Delta L_{i,\mathcal{P}'})x_i$

DKG-CL-Gen

$\mathsf{Gen}(\mathsf{pp}) \to \{\mathsf{PC}_{\chi_{ij}}, (c_{\chi_{ij}}, C_{\chi_{ij}}), (c_{\chi'_{ij}}, C_{\chi'_{ij}})\}_{j\in\mathcal{P}}$

1 : $\chi_i \leftarrow\$ [0, B], \{\chi_{ij}\}_{j\in\mathcal{P}} \leftarrow \mathbb{Z}\text{-SS.Share}(\chi_i, \mathcal{P})$
2 : $\chi'_i \leftarrow\$ [0, B], \{\chi'_{ij}\}_{j\in\mathcal{P}} \leftarrow \mathbb{Z}\text{-SS.Share}(\chi'_i, \mathcal{P})$
3 : **foreach** $j \in \mathcal{P}$ **do**
4 : $\quad \mathsf{PC}_{\chi_{ij}} := \mathfrak{h}^{\chi_{ij}} \mathfrak{g}_q^{\chi'_{ij}}$
5 : $\quad K_{ij} \leftarrow\$ \mathbb{F}_q, c_{\chi_{ij}} \leftarrow \mathsf{Enc}(\mathsf{ek}_j, K_{ij})$
6 : $\quad K'_{ij} \leftarrow\$ \mathbb{F}_q, c_{\chi'_{ij}} \leftarrow \mathsf{Enc}(\mathsf{ek}_j, K'_{ij})$
7 : $\quad C_{\chi_{ij}} \leftarrow \mathsf{SEnc}(K_{ij}, \chi_{ij}), C_{\chi'_{ij}} \leftarrow \mathsf{SEnc}(K'_{ij}, \chi'_{ij})$
8 : **return** $\{\mathsf{PC}_{\chi_{ij}}, (c_{\chi_{ij}}, C_{\chi_{ij}}), (c_{\chi'_{ij}}, C_{\chi'_{ij}})\}_{j\in\mathcal{P}}$

$\mathsf{GenVf}(\{\{\mathsf{PC}_{\chi_{i\ell}}\}_{\ell\in\mathcal{P}}, (c_{\chi_{ij}}, C_{\chi_{ij}}), (c_{\chi'_{ij}}, C_{\chi'_{ij}})\}_{i\in\mathcal{P}}, \mathsf{dk}_j)$

1 : $\{x_\ell^\perp\}_{\ell\in\mathcal{P}} \leftarrow \mathbb{Z}\text{-SS.Dual}(\mathcal{P})$
2 : **foreach** $i \in \mathcal{P}$ **do**
3 : $\quad$ **if** $\prod_{\ell\in\mathcal{P}} \mathsf{PC}_{\chi_{i\ell}}^{x_\ell^\perp} \neq 1$ **then** $\mathcal{P} := \mathcal{P} \setminus \{i\}$
4 : $\quad$ **else**
5 : $\quad\quad K_{ij} \leftarrow \mathsf{Dec}(\mathsf{dk}_j, c_{\chi_{ij}}), \chi_{ij} \leftarrow \mathsf{SDec}(K_{ij}, C_{\chi_{ij}})$
6 : $\quad\quad K'_{ij} \leftarrow \mathsf{Dec}(\mathsf{dk}_j, c_{\chi'_{ij}}), \chi'_{ij} \leftarrow \mathsf{SDec}(K'_{ij}, C_{\chi'_{ij}})$
7 : $\quad\quad$ **if** $\mathsf{PC}_{\chi_{ij}} \neq \mathfrak{h}^{\chi_{ij}} \mathfrak{g}_q^{\chi'_{ij}}$
8 : $\quad\quad\quad \mathsf{Blame}(K_{ij}, c_{\chi_{ij}}, K'_{ij}, c_{\chi'_{ij}}, \mathsf{dk}_j)$
9 : $\quad\quad\quad \mathcal{P} := \mathcal{P} \setminus \{i\}$

Fig. 11: Generation Phase of Our 3-round Variant

for $\mathcal{P} \in \mathbb{A}$. For secrecy, the simulator on input $\mathfrak{g}_q^x$ outputs $\mathfrak{g}_q^{\Delta^3 x}$. The secret key is defined as $\Delta^2 x$ to base $\mathfrak{g}_q^\Delta$ for public key $\mathfrak{g}_q^{\Delta^3 x}$. For correctness, as $x$ is an integer computed as sum of $\chi_i$ from a uniform distribution, we can only guarantee that $\mathfrak{g}_q^{\Delta^3 x}$ is uniformly distributed over the group generated by $\mathfrak{g}_q^{\Delta^3}$.

**Lemma 10.** *If the Pedersen Commitment over class groups is computationally binding and statistically hiding,* $\mathsf{Z}$ *is (standard)-sound and honest-verifier zero-knowledge,* $\mathbb{Z}$-SS *satisfies correctness, robustness, and secrecy, and* CLE *is correct and* IND-CPA*-secure, then our* DKG-CL *is correct and robust (with* $<t$ *corrupted parties and an honest majority).*

**Lemma 11.** *If* $\mathsf{Z}$ *is honest-verifier zero-knowledge,* $\mathbb{Z}$-SS *has secrecy, Pedersen commitment over* $\mathfrak{G}$ *is perfect hiding, and* CLE *is* IND-CPA*-secure, our* DKG-CL *has secrecy.*

## APPENDIX C
## 3-ROUND VARIANT OF DKG PROTOCOLS

### A. Generation Phase (Gen ↔ GenVf)

Figure 11 shows the 2-round generation phase that Gen and GenVf now involve $O(n)$ and $O(n^2)$ operations, respectively. Let $(\mathsf{SEnc}, \mathsf{SDec})$ be a symmetric key encryption scheme with key $K \in \mathbb{F}_q$. On input message $m$ and symmetric key $K$, algorithm SEnc returns $C_m$. On input ciphertext $C_m$ and symmetric key $K$, the deterministic algorithm SDec returns $m$.

Similar to DKG-DL and DKG-CL, dealer $\mathsf{P}_i$ distributes $(t, n)$-threshold shares of an initial secret $\chi_i$ to all others.

DKG-CL-Blame

$\mathsf{Blame}(K_{ij}, c_{\chi_{ij}}, K'_{ij}, c_{\chi'_{ij}}, \mathsf{dk}_j) \to (\pi_{\chi_{ij}})$

1 : **parse** $c_{\chi_{ij}} = (c_{\chi_{ij},0}, c_{\chi_{ij},1}),\ c_{\chi'_{ij}} = (c_{\chi'_{ij},0}, c_{\chi'_{ij},1})$
2 : $c^*_{\chi_{ij}} = (c_{\chi_{ij},0}, c_{\chi_{ij},1}/\mathfrak{f}^{K_{ij}}), c^*_{\chi'_{ij}} = (c_{\chi'_{ij},0}, c_{\chi'_{ij},1}/\mathfrak{f}^{K'_{ij}})$
3 : $\pi_{\chi_{ij}} \leftarrow \mathsf{Z}_{\mathsf{2Dec}}.\mathsf{P}((c^*_{\chi_{ij}}, c^*_{\chi'_{ij}}, \mathsf{ek}_j); \mathsf{dk}_j)$
4 : **return** $(K_{ij}, K'_{ij}, \pi_{\chi_{ij}})$

$\mathsf{BlameVf}(\mathsf{PC}_{\chi_{ij}}, c_{\chi_{ij}}, C_{\chi_{ij}}, c_{\chi'_{ij}}, C_{\chi'_{ij}}, (K_{ij}, K'_{ij}, \pi_{\chi_{ij}}))$

1 : $\chi_{ij} = \mathsf{SDec}(K_{ij}, C_{\chi_{ij}}), \chi'_{ij} = \mathsf{SDec}(K'_{ij}, C_{\chi'_{ij}})$
2 : **if** $\mathsf{PC}_{\chi_{ij}} \neq \mathfrak{h}^{\chi_{ij}} \mathfrak{g}_q^{\chi'_{ij}}$
3 : $\quad$ **parse** $(c_{\chi_{ij}}, c_{\chi'_{ij}}) = ((c_{\chi_{ij},0}, c_{\chi_{ij},1}), (c_{\chi'_{ij},0}, c_{\chi'_{ij},1}))$
4 : $\quad$ **parse** $c^*_{\chi_{ij}} = (c_{\chi_{ij},0}, c_{\chi_{ij},1}/\mathfrak{f}^{K_{ij}})$
5 : $\quad$ **parse** $c^*_{\chi'_{ij}} = (c_{\chi'_{ij},0}, c_{\chi'_{ij},1}/\mathfrak{f}^{K'_{ij}})$
6 : $\quad$ **if** $\mathsf{Z}_{\mathsf{2Dec}}.\mathsf{Vf}(\pi_{\chi_{ij}}, (c^*_{\chi_{ij}}, c^*_{\chi'_{ij}}, \mathsf{ek}_j)) = 1$
7 : $\quad\quad \mathcal{P} := \mathcal{P} \setminus \{i\}$
8 : $\mathcal{Q} := \mathcal{P}$

Fig. 12: Complain Part of Our 3-round Variant

Instead of CL encryption, this variant uses hybrid encryption to encrypt shares $\{\chi_{ij}\}_{j\in\mathcal{P}}$ (Line 5-7 of Gen) for efficiency.

The receiver $\mathsf{P}_j$ verifies the consistency of the committed shares $\{\mathsf{PC}_{i\ell}\}_{\ell\in\mathcal{P}}$ (Line 3 of GenVf) from each dealer $\mathsf{P}_i$. $\mathsf{P}_j$ verifies the commitment $\mathsf{PC}_{\chi_{ij}}$ (Line 7 of GenVf) via the decryption result $\chi_{ij}, \chi'_{ij}$ (Line 5-6 of GenVf). $\mathsf{P}_j$ broadcasts evidence from Blame if verification fails.

When the verification of the committed share fails, receiver $\mathsf{P}_j$ invokes Blame to generate evidence, to be verified by BlameVf, both presented in Figure 12. The evidence contains two symmetric keys $K_{ij}, K'_{ij}$, and the proof from $\mathsf{Z}_{\mathsf{Dec}}$ to show the correctness of two keys $K_{ij}, K'_{ij}$ (correct decryption of the ciphertexts). Parties verify each complaint (evidence) from $\mathsf{P}_j$ against $\mathsf{P}_i$ by cross-checking with the commitment $\mathsf{PC}_{\chi_{ij}}$ from Gen (Line 2 of BlameVf), and the proof (Line 6 of BlameVf).

### B. Revelation Phase (Reveal ↔ RevealVf)

Figure 12 shows the reveal phase of our variant. Similar to DKG-DL/DKG-CL, we generate $(t, n)$-threshold shares $\{x_i\}_{i\in\mathcal{P}}$ of the final secret $x = \sum_{j\in\mathcal{Q}} \chi_j$ via linear homomorphism of the shares. This phase concludes with $\mathsf{P}_i$ holding secret share $x_i$ of secret key $x$ and public key $\mathfrak{X} = (\mathfrak{g}_q^{\Delta^3})^x$.

Each party $\mathsf{P}_i \in \mathcal{Q}$ invokes Reveal, which inputs the decrypted shares $\chi_{ij}, \chi'_{ij}$ and returns the public key's shares $\mathfrak{X}_i = (\mathfrak{g}_q^\Delta)^{x_i}$. The correctness of the public key's share $\mathfrak{X}_i$ is verified against the (homomorphically-added) committed share via the proof $\pi_{\mathfrak{X}_i}$. At the end of this phase, the public key $\mathfrak{X}$ is computed via interpolating in exponent $\prod_{i\in\mathcal{P}} \mathfrak{X}_i^{\Delta L_{i,\mathcal{P}}} = \prod_{i\in\mathcal{P}} \mathfrak{g}_q^{\Delta^2 L_{i,\mathcal{P}} x_i} = \mathfrak{g}_q^{\sum_{i\in\mathcal{P}} \Delta^2 L_{i,\mathcal{P}} x_i} = \mathfrak{g}_q^{\Delta^3 x} = \mathfrak{X}$.

DKG-CL-Reveal

$\mathsf{Reveal}(\{\mathsf{PC}_{\chi_{ji}}, \chi_{ji}, \chi'_{ji}\}_{j\in\mathcal{Q}}) \to (x_i, \mathfrak{X}_i, \pi_{X_i})$

$x_i = \sum_{j\in\mathcal{Q}} \chi_{ji},\ x'_i = \sum_{j\in\mathcal{Q}} \chi'_{ji}$
$\mathsf{PC}_{x_i} := \prod_{j\in\mathcal{Q}} \mathsf{PC}_{\chi_{ji}},\ \mathfrak{X}_i = (\mathfrak{g}_q^{\Delta})^{x_i}$
$\pi_{\mathfrak{X}_i} \leftarrow \mathsf{Z}_{\mathsf{PC\text{-}DL}}.\mathsf{P}((\mathsf{PC}_{x_i}, \mathfrak{X}_i); (x_i, x'_i))$
**return** $(x_i, \mathfrak{X}_i, \pi_{\mathfrak{X}_i})$

$\mathsf{RevealVf}(\{\{\mathsf{PC}_{\chi_{ji}}\}_{j\in\mathcal{Q}}, \mathfrak{X}_i, \pi_{\mathfrak{X}_i}\}_{i\in\mathcal{P}}) \to \mathfrak{X}$

**foreach** $i \in \mathcal{P}$ **do**
  $\mathsf{PC}_{x_i} := \prod_{j\in\mathcal{Q}} \mathsf{PC}_{\chi_{ji}}$
  **if** $\mathsf{Z}_{\mathsf{PC\text{-}DL}}.\mathsf{Vf}(\pi_{X_i}, (X_i, \mathsf{PC}_{x_i}, \mathfrak{g}_q^{\Delta})) = 0$
    $\mathcal{P} := \mathcal{P} \setminus \{i\}$
**return** $\mathfrak{X} := \prod_{i\in\mathcal{P}} \mathfrak{X}_i^{\Delta L_{i,\mathcal{P}}}$

Fig. 13: Revelation Phase of Our 3-round Variant

## APPENDIX D
## INTEGER SECRET SHARING RELATED PROOFS

### A. Proof of Lemma 3

*Proof:* Let $\prod_{i\in\mathcal{P}} \mathsf{PC}_{F(i)}^{x_i^{\perp}} = 1$, if $\deg F > t-1$, we can break the binding property of Pedersen Commitment. Given shares $\{F(i), F'(i)\}_{i\in\mathcal{P}}$, we can always interpolate $F(z) = \sum_{d=0}^{n-1} a_d z^d$ and $F'(z) = \sum_{d=0}^{n-1} a'_d z^d$. Let $h = g^e$, the commitment $\mathsf{PC}_{F(i)}$ can be seen as the lifted share from the polynomial $F^*(z) := F(z) + eF'(z)$, *i.e.*, $\mathsf{PC}_{F(i)} = g^{F^*(i)}$. Since $\prod_{i\in\mathcal{P}} \mathsf{PC}_{F(i)}^{x_i^{\perp}} = 1$ implies that $\deg F^* \le t-1$ with overwhelming probability, we have $g^{a_d} h^{a'_d} = 1$ for $d \in [t, n-1]$.

By $\deg F > t-1$, we have $a_d \neq 0$ for some $d \in [t, n-1]$. By $g^{a_d} h^{a'_d} = 1$, we have $g^{a_d} = h^{-a'_d}$, implies $a'_d \neq 0$ for some $d \in [t, n-1]$. Using the tuple $(a_d, a'_d)$, we can break the binding property by outputting $(m_0, r_0)$ and $(m_1, r_1) := (m_0 + a_d, r_0 - a'_d)$ for arbitrary $(m_0, r_0) \in \mathbb{F}_q^2$. Say, the commitment $g^{m_0} h^{r_0}$ can be opened to $(m_0, r_0)$ and $(m_1, r_1)$ at the same time. ∎

### B. Proof of Lemma 4

*Proof:* The high-level idea follows Lemma 3. We prove by contradiction. Let $\prod_{i\in\mathcal{P}} (\mathfrak{h}^{F'(i)} \mathfrak{g}_q^{F(i)})^{x_i^{\perp}} = 1$, if $\deg F > t-1$, we can break the binding property of commitments.

Given $\{F(i), F'(i)\}_{i\in\mathcal{P}}$, we interpolate the polynomials $\Delta F(z) = \sum_{d=0}^{n-1} (\Delta b_d) z^d$ and $\Delta F'(z) = \sum_{d=0}^{n-1} (\Delta b'_d) z^d$ in $\mathbb{Z}[z]$. Let $\mathfrak{h} = \mathfrak{g}_q^e$ and $\Delta F^*(z) := e(\Delta F(z)) + (\Delta F'(z))$, we consider the commitment $(\mathfrak{h}^{F'(i)} \mathfrak{g}_q^{F(i)})^{\Delta} = \mathfrak{g}_q^{\Delta F^*(i)}$.

For $\{x_i^{\perp}\}_{i\in\mathcal{P}} \leftarrow \mathbb{Z}\text{-}\mathsf{SS.Dual}(\mathcal{P})$, $\prod_{i\in\mathcal{P}} (\mathfrak{g}_q^{\Delta F^*(i)})^{x_i^{\perp}} = 1$ when $\prod_{i\in\mathcal{P}} (\mathfrak{h}^{F'(i)} \mathfrak{g}_q^{F(i)})^{x_i^{\perp}} = 1$. If $\deg \Delta F^* > t-1$, the probability that $\prod_{i\in\mathcal{P}} (\mathfrak{g}_q^{\Delta F^*(i)})^{x_i^{\perp}} = 1$ is $1/\mathsf{ord}(\mathfrak{g}_q^{\Delta})$, which is negligible. So, $\mathfrak{h}^{\Delta b_d} \mathfrak{g}_q^{\Delta b'_d} = 1$ for $d \in [t, n-1]$.

Using the tuple $(\Delta b_d, \Delta b'_d)$, we can break the binding property by outputting $(m_0, r_0)$ and $(m_1, r_1) := (m_0 + \Delta b_d, r_0 - \Delta b'_d)$ for arbitrary $(m_0, r_0) \in \mathbb{Z}^2$, *i.e.*, commitment $g^{m_0} h^{r_0}$ can be opened to $(m_0, r_0)$ and $(m_1, r_1)$ simultaneously. ∎

### C. Argument for $\Delta L_{i,\mathcal{P}}(z) \in \mathbb{Z}$ for $z \in \mathbb{Z}$

We show that $\Delta L_{i,\mathcal{P}} = \prod_{j\in\mathcal{P}\setminus\{i\}} \frac{\Delta}{j-i}$ equals $K \cdot \frac{n!}{r!(n-r)!} \in \mathbb{Z}$ for some integer $K$ and the binomial coefficient $\frac{n!}{r!(n-r)!}$, which implies that $\Delta L_{i,\mathcal{P}}(z) \in \mathbb{Z}$ for variable $z \in \mathbb{Z}$. Let $\mathcal{P} \subseteq [1, n]$. We write $\prod_{j\in\mathcal{P}\setminus\{i\}} (j-i)$ as $\left(\prod_{j\in\mathcal{P}, j>i} j-i\right) \cdot \left(\prod_{j\in\mathcal{P}, j<i} j-i\right)$, splitting positive and negative parts. For positive part, let $r$ be the maximum value of $j-i$ for $j \in \mathcal{P}$ and $j > i$, since $r!$ is divisible by $\prod_{j\in\mathcal{P}, j>i} j-i$, we can express $r! = K^+ \cdot \prod_{j\in\mathcal{P}, j>i} j-i$ for some $K^+ \in \mathbb{Z}$. Similarly, for negative part, let $-(n-r)$ be the minimum value of $j-i$ for $j \in \mathcal{P}$ and $j < i$, since $(n-r)!$ is divisible by $\prod_{j\in\mathcal{P}, j<i} j-i$, we can express $(n-r)! = K^- \cdot \prod_{j\in\mathcal{P}, j<i} j-i$ for some $K^- \in \mathbb{Z}$ . Combining these results, we have $r!(n-r)! = (K^+K^-) \cdot \prod_{j\in\mathcal{P}\setminus\{i\}} j-i$. Thus, for $\frac{n!}{r!(n-r!)} = \frac{\Delta}{(K^+K^-)\cdot(\prod_{j\in\mathcal{P}\setminus\{i\}} j-i)} \in \mathbb{Z}$ and $K^+K^- \in \mathbb{Z}$, we have $\prod_{j\in\mathcal{P}\setminus\{i\}} \frac{\Delta}{j-i} = (K^+K^-) \cdot \frac{n!}{r!(n-r)!} \in \mathbb{Z}$, which concludes that the denominators can be canceled by $\Delta$ alone.

## APPENDIX E
## t-CL RELATED PROOFS

### A. Proof of Lemma 5

*Proof:* Let $\mathcal{P}$ be the set of honest parties and $\mathcal{P}' \in \mathbb{A}$ be a subset of $\mathcal{P}$. Let $\mathsf{ek}_j = \mathfrak{g}_q^{\Delta \mathsf{dk}_j}$. For honestly-executed Setup and KGen, by the correctness of DKG-CL run within KGen, we have $\sum_{j\in\mathcal{P}'} \Delta L_{j,\mathcal{P}'} \mathsf{dk}_j = \Delta^2 \mathsf{dk}$ and $\prod_{j\in\mathcal{P}'} \mathsf{ek}_j^{\Delta L_{j,\mathcal{P}'}} = \mathfrak{g}_q^{\Delta^3 \mathsf{dk}}$.

Given a honestly-generated and honestly-evaluated ciphertext[13] $c = (c_0, c_1) = ((\mathfrak{g}_q^{\Delta^2})^r, \mathfrak{f}^m \mathsf{ek}^r)$ and correct partial decryption $\mathsf{cpd}_j = (c_0^{\Delta})^{\mathsf{dk}_j}$ for all $j \in \mathcal{P}'$ from $\mathsf{PartDec}(\mathsf{ek}_j, c, \mathsf{dk}_j)$, we have $\prod_{j\in\mathcal{P}} \mathsf{cpd}_j^{\Delta L_{j,\mathcal{P}}}$ equals to:

$$c_0^{\sum_{j\in\mathcal{P}} (\Delta L_{j,\mathcal{P}})(\Delta \mathsf{dk}_j)} = c_0^{\Delta^3 \mathsf{dk}} = \mathfrak{g}_q^{r\Delta^5 \mathsf{dk}} = (\mathsf{ek}^r)^{\Delta^2}.$$

The reconstruction result can be used to cancel the mask of $c_1^{\Delta^2} = \mathfrak{f}^{\Delta^2 m} \mathsf{ek}^{\Delta^2 r}$ to obtain $M := \mathfrak{f}^{\Delta^2 m}$. Thus, we can obtain $m$ by computing $\mathsf{Dlog}(M)/\Delta^2 \bmod q$. ∎

### B. Proof of Lemma 6

The experiment $\mathsf{Expt}_{\mathsf{t\text{-}CL}}^{\mathsf{t\text{-}ind\text{-}cpa}}$ of t-ind-cpa-security is defined with the following interaction between a challenger and an adversary $\mathcal{A}$: Assumed that $\mathcal{A}$ takes control of a set of parties $\mathcal{C}$. $\mathcal{A}$ and the challenger collaboratively run the protocols Setup and KGen (so $\mathcal{A}$ knows $\{\mathsf{dk}_j\}_{j\in\mathcal{C}}$). After that, $\mathcal{A}$ sends two challenge plaintexts $m_0, m_1$, the challenger picks a random bit $b$ and replies with $c_b \leftarrow \mathsf{Enc}(\mathsf{ek}, m_b)$. $\mathcal{A}$ wins $\mathsf{Expt}_{\mathsf{t\text{-}CL}}^{\mathsf{t\text{-}ind\text{-}cpa}}$ if $\mathcal{A}$ successfully guesses $b$ with probability $> 1/2 + \mathsf{negl}(\lambda)$.

*Proof:* We show how to use any adversary $\mathcal{A}$ who wins $\mathsf{Expt}_{\mathsf{t\text{-}CL}}^{\mathsf{t\text{-}ind\text{-}cpa}}$ to win the experiment $\mathsf{Expt}_{\mathsf{CLE}}^{\text{IND-CPA}}$ of IND-CPA-security of CLE. The reduction takes the role of challenger of $\mathsf{Expt}_{\mathsf{t\text{-}CL}}^{\mathsf{t\text{-}ind\text{-}cpa}}$ and adversary of $\mathsf{Expt}_{\mathsf{CLE}}^{\text{IND-CPA}}$. Upon receiving

[13] The homomorphic evaluation part is the same as CL encryption, so we directly consider an evaluated ciphertext here.

the encryption key $\mathsf{ek}$ from the challenger of $\mathsf{Expt}^{\text{IND-CPA}}_{\mathsf{CLE}}$, we invoke the simulator of DKG-CL to generate an indistinguishable view of KGen protocol such that it outputs encryption key $\mathsf{ek}^*$ of t-CL as $\mathsf{ek}^* = \mathsf{ek}^{\Delta^3}$ and $\mathsf{ek}_i = (\mathfrak{g}_q^{\Delta})^{\mathsf{dk}}$ (which can be simulated by the simulator of DKG-CL).

Upon receiving $(m_0, m_1)$ from $\mathcal{A}$, we submit to the $\mathsf{Expt}^{\text{IND-CPA}}_{\mathsf{CLE}}$ challenger $(m_0/\Delta^3 \bmod q, m_1/\Delta^3 \bmod q)$ and obtain $(c_{b,0}, c_{b,1}) = (\mathfrak{g}_q^r, \mathfrak{f}^{m_b/\Delta^3}\mathsf{ek}^r)$. We then compute $((\mathfrak{g}_q^r)^{\Delta^2}, (\mathfrak{f}^{m_b/\Delta^3}\mathsf{ek}^r)^{\Delta^3}) = ((\mathfrak{g}_q^{\Delta^2})^r, \mathfrak{f}^{m_b}(\mathsf{ek}^{\Delta^3})^r)$, which match the ciphertext form of our scheme since $\mathsf{ek}^* = \mathsf{ek}^{\Delta^3}$.

This reduction algorithm differs from the real experiment $\mathsf{Expt}^{\mathsf{t\text{-}ind\text{-}cpa}}_{\mathsf{t\text{-}CL}}$ in the key generation and the challenge ciphertext. By Lemma 11, the view generated by the simulator of DKG-CL is indistinguishable from the real one. Meanwhile, the challenge ciphertext $c_b^*$ has the exact form as the real one. If $\mathcal{A}$ guesses $b$ successfully with a non-negligible advantage, relaying $b$ wins $\mathsf{Expt}^{\text{IND-CPA}}_{\mathsf{CLE}}$ with the same advantage. ■

### C. Proof of Lemma 7

Given a ciphertext $c$ and correct message $m$, there exists Sim for PartDec that simulates a partial decryption $\mathsf{pd}_i$ (which is indistinguishable from the real one) without the decryption key $\mathsf{dk}$ and threshold decryption key $\{\mathsf{dk}_j\}_{j\in\mathcal{P}\setminus\mathcal{C}}$. Hence, no information about the (threshold) decryption key is leaked.

*Proof:* Sim invokes the simulator of DKG-CL to generate an indistinguishable view of KGen protocol, obtain $\{\mathsf{dk}_j\}_{j\in\mathcal{C}}$, and output encryption key $\mathsf{ek}$ of t-CL. By Lemma 11, this view of DKG-CL is indistinguishable from the real one.

To simulate partial decryption of $(c_0, c_1) = (\mathfrak{g}_q^{\Delta^2 r}, \mathfrak{f}^m\mathsf{ek}^r)$ encrypting plaintext $m$, Sim computes $c_1/\mathfrak{f}^m = \mathfrak{g}_q^{\Delta^3\mathsf{dk}r} = c_0^{\Delta\mathsf{dk}} = c_0^{F(0)}$, where $F(z) = \Delta\mathsf{dk} + \sum_{d=0}^{t-1} a_d z^d$ is the polynomial used in secret sharing. Sim computes from shares $\{\mathsf{dk}_j\}_{j\in\mathcal{C}}$ and $c_0^{F(0)}$ the lifted polynomial $c_0^{\Delta F(z)} = \prod_{i\in\mathcal{C}\cup\{0\}}(c_0^{F(i)})^{\Delta L_{i,\mathcal{P}}(z)}$ by Lemma 1. For $i \in \mathcal{P}\setminus\mathcal{C}$, Sim computes partial decryption $\mathsf{cpd}_i = c_0^{\Delta F(i)}$, runs the simulator of $\mathsf{Z}_{\mathsf{PartDec}}$ to get $\pi_{\mathsf{cpd}_i}$, and broadcasts $\{\mathsf{cpd}_i, \pi_{\mathsf{cpd}_i}\}_{i\in\mathcal{P}\setminus\mathcal{C}}$. As $\mathsf{cpd}_i$ is perfectly simulated and $\mathsf{Z}_{\mathsf{PartDec}}$ is zero-knowledge, the view is indistinguishable from the real one. ■

## APPENDIX F
## THRESHOLD SIGNATURES RELATED PROOFS

### A. Proof of Theorem 1

*Proof:* By the correctness of DKG-DL and DKG-CL, our protocol generates correct key pairs $(\mathsf{vk}, \{\mathsf{sk}_i\}_{i\in\mathcal{P}})$ for threshold ECDSA, $(\mathsf{elek}, \{\mathsf{eldk}_i\}_{i\in\mathcal{P}})$ for t-ElG, and $(\mathsf{clek}, \{\mathsf{cldk}_i\}_{i\in\mathcal{P}})$ for t-CL. Our protocol computes $s = km + rkx$, $k\gamma$, and $g^\gamma$ in plaintext space homomorphically, which is guaranteed to be correct by the evaluation correctness of LHE.

For robustness, assuming there are $<t$ corrupted parties and an honest majority, we show that removing any cheater does not make the protocol abort. First, $k$ is not fixed in Step 1, so we can safely remove the cheaters (by verifying the proofs). $k = \sum_i k_i$ is fixed after Step 2, but its ingredient $k_i$ has been correctly encrypted, so $k$ will not be lost. Due to the similar reason above, the value $k\gamma = \sum_i \gamma_i k$ and $g^\gamma = \prod_i g^{\gamma_i}$ can be correctly generated even if some parties cheat or become absent after Step 4. Notice that $x_i$ is a share of $x$, we can compute encrypted $xk$ from encrypted $x_i k$, as long as a subset $>t$ of honest parties exists. By the threshold decryption feature of t-CL and t-ElG, $\overline{km + rkx}$, $\overline{\gamma_j k}$, and $\overline{g^{\gamma_j}}$ can always be obtained from $\overline{km + rkx}$, $\overline{\gamma_j k}$, and $\overline{g^{\gamma_j}}$ as long as $t$ parties are involved. ■

### B. Proof of Theorem 2

*Proof:* An adversary corrupts a set $\mathcal{C} \notin \mathbb{A}$ of parties. Notice that TKeygen includes two instances of DKG-DL and an instance of DKG-CL. By the secrecy of DKG-DL, DKG-CL, there exists a simulator that can simulate the view of any real execution of DKG-DL, DKG-CL given only the protocol output. Hence, given (public) key of threshold ECDSA, t-ElG, and t-CL, we can invoke the corresponding simulator to generate view indistinguishable from the real execution of all three public keys. Hence, TKeygen is simulatable.

By the property in Section V-C, for $(R, s)$ received from ECDSA signing oracle, the simulator of partial decryption can simulate $\{\mathsf{pd}_{\overline{km+rkx},i}, \mathsf{pd}_{\overline{\gamma k},i}, \mathsf{pd}_{\overline{g^\gamma},i}\}_{i\in\mathcal{P}\setminus\mathcal{C}}$, s.t. $s, \omega, \Gamma$ are the final decryption of $\overline{km + rkx}$, $\overline{\gamma k}$, and $\overline{g^\gamma}$ with $R = \Gamma^{1/\omega}$. ■

### C. Proof of Theorem 3

*Proof:* Correct DKG-DL and DKG-CL generate correct key pairs $(\mathsf{vk}, \{\mathsf{sk}_i\}_{i\in\mathcal{P}})$ for threshold ECDSA, $(\mathsf{elek}, \{\mathsf{eldk}_i\}_{i\in\mathcal{P}})$ for t-ElG, and $(\mathsf{clek}, \{\mathsf{cldk}_i\}_{i\in\mathcal{P}})$ for t-CL. Homomorphical computations of $s$, $e$, $D^\gamma$, and $\gamma(x + e)$ are guaranteed to be correct by the LHE evaluation correctness.

For robustness, with $<t$ corrupted parties and an honest majority, $s$ ($e$) is not fixed in Step 1, so we can safely remove the cheater. Step 2 fixed $s = \sum_i s_i$ ($e = \sum_i e_i$), but its ingredient $s_i$ ($e_i$) has been correctly encrypted. As long as $t$ parties are involved, t-CL or t-ElG decryption of $\overline{s}$ or $\overline{e}$ can recover $s$ and $e$, respectively. Similarly, the rest of the steps make $D^\gamma$ and $\gamma(x + e)$ available to threshold decryption for outputting the BBS+ signature $((D^\gamma)^{\frac{1}{\gamma(x+e)}}, s, e)$. ■

### D. Proof of Theorem 4

*Proof:* Let an adversary corrupt a set $\mathcal{C} \notin \mathbb{A}$ of parties. By the secrecy of DKG-DL, DKG-CL, we have a simulator that generates an indistinguishable view of TKeygen that outputs the required key pair (the same public key of the underlying scheme) of threshold ECDSA, t-ElG, and t-CL. By the property in Section V-B, for $(A, s, e)$ received from signing oracle of BBS+ signature, we can simply use the simulator of partial decryption to simulate $\{\mathsf{pd}_{s,i}, \mathsf{pd}_{e,i}, \mathsf{pd}_{D^\gamma,i}\mathsf{pd}_{\gamma(x+e),i}\}_{i\in\mathcal{P}\setminus\mathcal{C}}$, such that $s, e, D^\gamma, \gamma(x + e)$ are the final decryption result of $\overline{s}$, $\overline{e}$, $\overline{D^\gamma}$ and $\overline{\gamma(x + e)}$, where $A = (D^\gamma)^{\frac{1}{\gamma(x+e)}}$. ■