

Adaptively Secure (Aggregatable) PVSS from Standard Assumptions

Renas Bacho^{1*}

Yanbo Chen^{2†}

Julian Loss^{1‡}

¹Ruhr University Bochum, Germany

²University of Ottawa, Canada

May 29, 2026

Abstract

Publicly verifiable secret sharing (PVSS) is a fundamental primitive in threshold cryptography that allows a dealer to share a secret S among a set of n parties via a publicly verifiable transcript. Any subset of $t+1$ parties can then use their individual shares to reconstruct the full secret S , whereas t or fewer shares give no information about S . As such, the secret S remains hidden from an adversary that corrupts up to t parties. Recently, Bacho and Loss (CCS 2023) gave the first proof of any PVSS scheme under an adaptive adversary. However, their security proof relies on strong and non-standard assumptions such as the algebraic group model (AGM) and the hardness of the one-more discrete logarithm (OMDL) problem. In particular, any protocol (e.g., distributed randomness beacon or distributed key generation) that makes use of a PVSS scheme either inherits these limitations or is not provably adaptively secure.

In this work, we present for the first time an adaptively secure PVSS scheme from well-established assumptions. In more detail, we provide two PVSS schemes with different properties. Our first scheme works over any pairing-free cyclic group and its security relies on the decisional Diffie-Hellman (DDH) assumption. Our second scheme works over an asymmetric pairing group, its security relies on the DDH and the co-computational Diffie-Hellman (co-CDH) assumption, and has the particularly valuable feature of *aggregatability*, which allows the aggregation of multiple PVSS transcripts into a single transcript while preserving verifiability. Notably, both our schemes are highly efficient, non-interactive, and work in the established plain public key model. These properties along with their provable adaptive security make them suitable candidates as building block in higher-level distributed protocols that aim to minimize communication.

Keywords. Threshold Cryptography, PVSS, Aggregatability, Adaptive Security

1 Introduction

Publicly verifiable secret sharing (PVSS) [Sta96] is a fundamental tool in threshold cryptography. It allows a dealer to share a secret S among a group of n parties by broadcasting a transcript tr_x such that any third party can verify that the secret was shared correctly and can be recovered by any subset of $t+1$ parties. Conversely, any (potentially colluding) t or fewer parties should gain no information about S . As such, the secret S remains completely hidden from an adversary

*This work was done while this author was at CISP. Email: renas.bacho@rub.de

†This work was partially done while this author was visiting CISP. Email: yichen918@uottawa.ca

‡This work was partially done while this author was at CISP. Email: julian.loss@rub.de

that corrupts up to t parties. In a *non-interactive* PVSS, the dealer can generate the transcript trx without any prior communication with the parties.¹ This makes non-interactive PVSS particularly attractive for large-scale distributed protocols where minimizing the communication overhead is essential. A prominent class of such protocols includes distributed randomness beacons (DRB) [CMB23, KWJ24] and distributed key generation (DKG) [BLL⁺24, BK25], which serve as fundamental building blocks in many distributed cryptographic applications. Other key applications of PVSS include electronic voting systems, robust public key infrastructures, threshold key escrow, secure multiparty computation, and more [Sch99, HMW18, CDGK22].

Adversary in Distributed Protocols. The security model of such protocols generally considers an adversary who may corrupt up to t (out of n) parties and cause them to deviate arbitrarily from the prescribed protocol description, also known as *Byzantine* adversary. Here, one further distinguishes two types of Byzantine adversaries: a *static* adversary and an *adaptive* adversary. A static adversary has to declare which signers to corrupt at the beginning of the protocol, and even before the public keys are set up. In contrast, an adaptive adversary can dynamically decide which signers to corrupt over the course of the protocol execution. Clearly, an adaptive adversary is a safer and more realistic model for the decentralized setting, and there is also a large body of works that have focused on the design of adaptively secure distributed protocols [CFGN96, CGJ⁺99, DGKR18, GHK⁺21].

Critical Gap in Adaptive PVSS. Despite its importance and popularity, achieving adaptive security in PVSS has remained an open problem until very recently. At ACM CCS 2023, Bacho and Loss [BL23] gave the first proof of any PVSS scheme under an adaptive adversary. Unfortunately, their security proof relies on strong and non-standard assumptions such as the algebraic group model (AGM) [FKL18] and the hardness of the one-more discrete logarithm (OMDL) problem [BNPS03]. In particular, any protocol (e.g., distributed randomness beacon or distributed key generation) that employs a PVSS scheme either inherits these limitations or is not provably adaptively secure.

1.1 Our Contributions

Motivated by this unsatisfactory state of affairs, we present **Illusi**, a novel non-interactive PVSS scheme. Our scheme works over any pairing-free cyclic group and is secure for up to $t < n$ adaptive corruptions under a well-studied and non-interactive assumption, namely, the DDH assumption. As a second contribution, we present a variant of **Illusi** that supports *aggregatability*, namely the aggregation of multiple transcripts into a single transcript while preserving verifiability.² This feature has proven to be of particular value, as demonstrated by recent works on communication-efficient distributed key and randomness generation [BLL⁺24, ABL25].³ Our second scheme **Illusia** works over any asymmetric pairing group and its security relies on the DDH and the co-computational Diffie-Hellman (co-CDH) assumption. Notably, both our schemes are highly efficient, non-interactive, and work in the established plain public key model. These properties along with their provable adaptive security make them suitable candidates as building block in higher-level distributed protocols that aim to minimize communication.

2 Technical Overview

We next provide an overview of our techniques. We begin by recalling some background on PVSS. For a discussion on additional related work, we refer to Appendix A.

¹However, we assume that the dealer knows the public keys of the parties (which is a standard requirement), which can be established via a bulletin-board public key infrastructure (PKI).

²In particular, the size of the aggregated transcript is of the same order as that of a single transcript.

³We note that these and other works [GJM⁺21, AJM⁺21] leverage aggregatability to obtain major efficiency improvements over prior approaches—for example, reducing communication complexity from cubic to quadratic.

2.1 Background: Non-Interactive PVSS

We begin by recalling the common design pattern that underlies most efficient non-interactive PVSS constructions in the literature. For this purpose, let $(\mathbb{G}, p, g, h)$ define a cyclic group $\mathbb{G}$ of prime order p with two independent generators g and h . Assume a dealer wants to share a secret S among a set of n parties $\{1, \dots, n\}$ by generating a transcript non-interactively, from which each party can extract its share of S . To this end, the dealer first samples a uniformly random value $s \leftarrow \mathbb{Z}_p$ and a degree- t polynomial $a(X) \leftarrow \mathbb{Z}_p[X]$ with $a(0) = s$. Then, it computes a commitment to the evaluation points⁴ of $a(\cdot)$ as $\text{com} := (g^s, g^{a(1)}, \dots, g^{a(n)})$, and it also forms a (t, n) -sharing of the secret $S := h^s \in \mathbb{G}$: For $i \in [n]$, the i -th share is defined as $S_i := h^{a(i)} \in \mathbb{G}$. Further, for each $i \in [n]$, it computes an “encryption” $c_i := \text{pk}_i^{a(i)}$ of the secret share S_i under party i ’s public key $\text{pk}_i = h^{\text{sk}_i}$, along with a (non-interactive zero-knowledge) proof⁵ π_i that attests correct computation of c_i (the proof uses the commitment $g^{a(i)}$ and the public key pk_i as statement, and the evaluation point $a(i)$ as witness). In summary, the transcript trx has the form

$$\text{trx} := (\text{com}, \{c_i, \pi_i\}_{i \in [n]}), \quad \text{com} := (g^s, g^{a(1)}, \dots, g^{a(n)}), \quad c := (\text{pk}_1^{a(1)}, \dots, \text{pk}_n^{a(n)}),$$

and using the commitments and proofs, anyone (knowing the list of public keys $\mathcal{L} := \{\text{pk}_1, \dots, \text{pk}_n\}$) can check that trx indeed forms a correct threshold secret sharing. In order to reconstruct the secret S , each party i extracts its share from the transcript trx as

$$S_i := c_i^{1/\text{sk}_i} = \left(\text{pk}_i^{a(i)}\right)^{1/\text{sk}_i} = \left(h^{\text{sk}_i \cdot a(i)}\right)^{1/\text{sk}_i} = h^{a(i)}.$$

Then, any collection of $t + 1$ valid shares⁶ suffices to recover the full secret S via Lagrange interpolation in the exponent. The literature considers two security notions for non-interactive PVSS: *indistinguishability* [RV05, CD17] and *unpredictability* [Sch99, BL23]. Indistinguishability requires that an adversary corrupting at most t of the n parties gains no information (from the transcript trx and the secret keys $\text{sk}_{i_1}, \dots, \text{sk}_{i_t}$ of the corrupted parties) about the shared secret S prior to reconstruction. Unpredictability, on the other hand, allows the adversary to obtain partial information about the secret, with the only condition being that it cannot fully recover it. Importantly, prior works [BL23, GJM⁺21, BLL⁺24] have shown that unpredictability suffices for most applications, including distributed randomness beacons and distributed key generation.

2.2 The Problem with Adaptive Security

We next discuss the limitations of this approach under adaptive corruptions, and explain why the (unpredictability) security proofs by Bacho and Loss [BL23] have to resort to the AGM and the OMDL assumption.

Assume we have an efficient reduction R that uses an adversary A breaking the security of the PVSS scheme to solve an underlying number-theoretic hardness problem (e.g., the decisional Diffie-Hellman problem). To this end, the reduction R simulates the respective security game for A . First, observe that R must simulate the game without knowing the secret keys sk_i and secret shares S_i of the honest parties in advance. (And note that it should also not know the discrete logarithm of h .) Otherwise, if the reduction knew those values, it could compute the secret S already by itself, making the adversary’s eventual output completely useless⁷. Moreover, guessing the set of adaptive corruptions results in an undesirable security loss of $\binom{n}{t}$, which grows exponentially in n .

⁴Early constructions commit directly to the coefficients of the polynomial. But the two approaches are equivalent and can be transformed into one another via Lagrange interpolation in the exponent.

⁵The proof can be realized using Fiat-Shamir-type proofs of discrete logarithm equality in the random oracle model, or using pairings.

⁶In some schemes, party i must also provide a proof of correct decryption, using its secret key sk_i as witness.

⁷This statement is true unless one assumes non-black-box access to the adversary, such as in the AGM.

While this might be tolerable for small n , many real-world applications—particularly in blockchain systems—have n in the range of several thousands. At the same time, when the adversary adaptively corrupts a party, the reduction must be able to provide that party’s secret key sk_i and share S_i . Importantly, observe that the public key pk_i is a *perfectly binding* commitment to the secret key sk_i via the relation $\text{pk}_i = h^{\text{sk}_i}$. As such, upon an adaptive corruption, the reduction somehow has to be able to (efficiently) return the discrete logarithm value of a group element pk_i whose discrete logarithm value it should not know.

To address this issue, Bacho and Loss [BL23] leverage the k -OMDL assumption with $k = n$. Roughly, the k -OMDL assumption (where k is called the *degree*) for $k \in \mathbb{N}$ asserts that it is hard to compute the discrete logarithm values $z_1, \dots, z_k \in \mathbb{Z}_p$ of k challenge group elements $g^{z_1}, \dots, g^{z_k} \in \mathbb{G}$ when given $(k - 1)$ -time access to a discrete logarithm oracle DL_g (which on input $\zeta \in \mathbb{G}$ returns its discrete logarithm $z \in \mathbb{Z}$ to base g). In more detail, their security reduction strategy is to embed the n -OMDL challenge $\zeta_1, \dots, \zeta_n \in \mathbb{G}$ in either the public keys $\text{pk}_1, \dots, \text{pk}_n$ of parties, the secret polynomial $a(\cdot) \in \mathbb{Z}_p[X]$ chosen by the dealer, or the second generator $h \in \mathbb{G}$, a choice that remains hidden from the adversary. During simulation, the discrete logarithm oracle DL_g is then used in the first case to handle adaptive corruptions: Upon corruption query $i \in \mathcal{H}$, the reduction calls DL_g on input pk_i to obtain the secret key sk_i (it knows the discrete logarithm value of h in that case). We emphasize that their reduction can compute the secret S on its own *in all cases*, since it always knows the secret keys of honest parties, the secret polynomial $a(\cdot)$, or the discrete logarithm of h . Notably, this observation explains why their proof crucially relies on the algebraic group model. Using the algebraic equation that comes from the adversary’s output (with its algebraic representation), the reduction can solve the n -OMDL challenge. Recall that the reduction can compute the secret S by itself, but this would not allow it to solve the n -OMDL challenge, as the secret is an element in the group $\mathbb{G}$, while the n -OMDL problem asks for values in the field $\mathbb{Z}_p$. Finally, we note that one could try to avoid the AGM in their proof by replacing OMDL with an even more non-standard assumption—akin to a “one-more CDH” variant—that would allow simulating $\text{com}_i = g^{a(i)}$, $\text{pk}_i = h^{\text{sk}_i}$, and $c_i = h^{\text{sk}_i a(i)}$ without knowing $a(i)$, sk_i , $\text{dlog}_g(h)$ (otherwise, the reduction could already compute the secret S on its own).

After revisiting the details of their proof and its limitations, it becomes clear that the PVSS construction itself must be reconsidered. In particular, to achieve adaptive security without relying on the AGM and OMDL assumptions, we need to depart from the common design template used in existing PVSS schemes.

2.3 Overview of Our Solution

One source of the need for the OMDL assumption under adaptive corruptions stems from the perfectly binding commitments to the secret key via the public key. A natural way to address this is to switch to perfectly hiding commitments instead, for example Pedersen commitments. In our PVSS construction, each user i holds a secret key $(x_i, y_i) \leftarrow \mathbb{Z}_p^2$ and a public key $\text{pk}_i := g^{x_i} u^{y_i} \in \mathbb{G}$, where u is an additional uniformly random group element. To share a secret $S := h^s \in \mathbb{G}$, the dealer follows the standard procedure: it samples a uniformly random polynomial $a(X) \in \mathbb{Z}_p[X]$ of degree t with $a(0) = s$, and outputs the transcript trx as

$$\text{trx} := (\text{com}, \{c_i, \pi_i\}_{i \in [n]}),$$

where $\text{com} := (g^s, g^{a(1)}, \dots, g^{a(n)})$ commits to $a(\cdot)$. For $i \in [n]$, c_i is an encryption of the secret share $S_i := h^{a(i)}$, and π_i is a non-interactive zero-knowledge proof of the correctness of encryption. Since we modified the structure of the public keys, it is no longer immediate how to compute c_i . To address this, we employ a variant of ElGamal encryption [Dam92], which supports our public key

structure.⁸ The encryption of S_i takes the form

$$c_i := (c_{i,1}, c_{i,2}, c_{i,3}) := (g^{r_i}, u^{r_i}, \text{pk}_i^{r_i} S_i)$$

where $r_i \leftarrow \mathbb{Z}_p$ is sampled uniformly at random. The proof π_i proves the existence of $a(i)$ and r_i that are consistent with the commitment $g^{a(i)}$ and the encryption c_i . From the transcript, each party i can then extract its share as $S_i := c_{i,3}/(c_{i,1}^{x_i} c_{i,2}^{y_i})$.

Adding Aggregatability. We next consider the goal of extending our above construction to an aggregatable PVSS (APVSS) scheme [BL23]. Such a scheme supports combining multiple individual transcripts $\{\text{tr}_l\}_{l \in [k]}$ into a single compact aggregated transcript tr . The relevant security notion is that of unpredictability [BL23], which requires that the adversary cannot fully recover the secret of an aggregated transcript that contains at least one honest party’s contribution, even if the adversary is allowed to contribute to the aggregate itself.

The homomorphic properties of the commitment com and of the encryptions $\{c_i\}_{i \in [n]}$ in our above construction naturally enable aggregation. Our non-interactive Fiat–Shamir-type proofs $\{\pi_i\}_{i \in [n]}$, however, are not homomorphic. By contrast, Groth–Sahai proofs [GS08] admit homomorphism for linear relations [DHLW10], which is precisely the setting of our construction. We can therefore directly swap the Fiat–Shamir proofs for homomorphic Groth–Sahai proofs.

Nonetheless, this introduces a new challenge: we must prevent an adversarial strategy that exploits the homomorphic property. Suppose the adversary observes a transcript tr_{hon} generated by an honest party. It can then construct a transcript tr_{bad} for which it does not know the underlying secret, but which aggregates with tr_{hon} to yield a transcript tr whose secret it can fully recover. Concretely, it can compute $\text{tr}_{\text{bad}} := \text{tr} \cdot \text{tr}_{\text{hon}}^{-1}$, where tr is any transcript of its choice whose secret it knows (e.g., the trivial transcript that shares $1 \in \mathbb{G}$). This type of malleability attack is a serious concern, as it would allow an adversary to learn the secrets of aggregated transcripts and thereby compromise the security of applications such as distributed randomness beacons.

The common approach [BSKN23, GJM⁺21] to prevent such attacks is to attach a signed proof of knowledge of the value $s \in \mathbb{Z}_p$ to each individual transcript. Concretely, a proof of knowledge π for s is generated using the commitment g^s as statement (and then signing this pair). However, this approach introduces a critical challenge under adaptive corruptions. Extracting the witness from the proof of knowledge requires rewinding the adversary.⁹ Since the adversary is adaptive, it may corrupt up to $2t$ parties across the two runs. This could force the reduction in the proof to recover the secret (since it would need to output $2t \geq t + 1$ secret shares), which brings us back to the challenges outlined in Section 2.2. We therefore aim for a *rewinding-free* solution. Besides avoiding this issue, such a solution also yields a tighter security bound, which is particularly important for real-world applications [BR96, KW03].

Our solution is both simple and efficient. The key observation is that requiring online extraction of $s \in \mathbb{Z}_p$ is unnecessarily strong. Ultimately, what matters is not s itself, but the shared secret $S = h^s \in \mathbb{G}$, which lies in the group. Online extraction over $\mathbb{Z}_p$ is rather inefficient and requires on the order of λ many field elements [Fis05]. In contrast, online extraction over $\mathbb{G}$ is very efficient and can be realized using plain ElGamal encryption under a public parameter $v \in \mathbb{G}$.

Our final solution is a signed ElGamal encryption of the secret $S \in \mathbb{G}$ under the public parameter $v \in \mathbb{G}$ along with a (simulation-sound) NIZK proof π_0 of correct encryption. Specifically, the dealer samples $r \leftarrow \mathbb{Z}_p$ uniformly at random and encrypts the secret S towards v as $c_0 := (g^r, v^r h^S)$. Further, an efficient (Σ -protocol-based) proof π_0 of correct encryption relative to g^S , included in the commitment com , is computed and appended to the encryption c_0 in the form of a tuple (c_0, π_0) . Knowledge of the discrete logarithm of v (to base g) then enables online extraction of the committed value $S = h^s$, while simulation-soundness of π_0 guarantees its correctness.

⁸The resulting encryption scheme can also be derived from the Cramer-Shoup hash proof system [CS02].

⁹Recall that all prior works, except [BL23], show only static security. And [BL23] shows adaptive security in the AGM, which allows for online (i.e., rewinding-free) extraction of proofs of knowledge.

2.4 Proving Adaptive Security

We prove the indistinguishability-based security of our plain PVSS scheme under the DDH assumption. To aid the reader’s understanding, we describe the reduction idea to the notion of unpredictability from the CDH assumption. The difference compared to the indistinguishability-based proof is only minor and appears essentially only in the final hybrid game. Recall that indistinguishability [RV05, CD17] requires that the adversary cannot distinguish the secret S in a transcript trx from a uniformly random element S' . In contrast, unpredictability [Sch99, BL23] only requires that the adversary cannot output (i.e., predict) the secret S in trx (after observing the public information, the transcript, and corrupting up to t parties). We note that the notion of unpredictability traces back to the work of Schoenmakers [Sch99]. In the following, we provide a high-level *intuition* for our security proof. We emphasize, however, that the formal proof is highly technical and requires considerable care. This is further underscored by the fact that prior work [BL23] has had to rely on strong assumptions, such as OMDL and the AGM, in order to obtain adaptive security proofs.

The CDH instance is embedded into h and $g^{a(0)}$. If the adversary succeeds in recovering the secret $S = h^{a(0)}$, then we can extract a solution to the CDH problem. The central challenge is that, since S is unknown, we cannot directly generate the secret shares $S_1, \dots, S_n$. To overcome this, we switch to a hybrid game in which the shares $S_1, \dots, S_n$ are not sampled during transcript generation. Instead, each S_i is sampled only when user i becomes corrupted. As the adversary can corrupt at most t users, we only need to generate t shares. Furthermore, because $S, S_1, \dots, S_n$ are $(t+1)$ -wise independent, these shares can be sampled uniformly at random without reference to S .

To reach this hybrid game, we must eliminate the dependence of the transcript trx on the shares $S_1, \dots, S_n$. The commitment $\text{com} := (g^s, g^{a(1)}, \dots, g^{a(n)})$ to the polynomial $a(\cdot)$ can, by the DDH assumption, be replaced with a commitment to another independent polynomial $b(\cdot)$. The non-interactive proof π_i can be simulated in zero-knowledge. The most delicate part is the encryption. Using another application of DDH, the encryption c_i can be generated in an “equivocable mode” as follows:

$$c_i := (c_{i,1}, c_{i,2}, c_{i,3}) := (g^{r_i}, u^{r'_i}, c_{i,1}^{x_i} c_{i,2}^{y_i} S_i),$$

where $r_i \neq r'_i$. The randomness of (x_i, y_i) , conditioned on pk_i , guarantees that $c_{i,3}$ can be generated independently of S_i . If user i is corrupted later, we can then compute the corresponding (x_i, y_i) .

Aggreateable PVSS Scheme. For our APVSS scheme, we prove the security notion of unpredictability [BL23] under the CDH assumption.¹⁰ This notion requires that the secret shared by an aggregated transcript $\text{trx} = \text{Agg}(\text{trx}_1, \dots, \text{trx}_k)$ remains unpredictable to the adversary, provided that at least one transcript in $\{\text{trx}_l\}_{l \in [k]}$ is generated by an uncorrupted party, even if the adversary generates all of the remaining transcripts.¹¹

The security proof of our APVSS scheme is not difficult, given we have already established the security of the plain PVSS scheme. Namely, we embed a CDH instance into an honest transcript trx_{hon} (i.e., the transcript of a party that remains honest throughout the experiment). The additional signed proof of knowledge can be simulated by switching to a fake encryption (indistinguishable under DDH) and simulating the corresponding proof of correctness in zero knowledge. The adversary then outputs an aggregated transcript

$$\text{trx} = \text{Agg}(\text{trx}_{\text{bad},1}, \dots, \text{trx}_{\text{bad},k}, \text{trx}_{\text{hon}}),$$

which contains the honest transcript trx_{hon} , along with the secret S^* of trx . To construct a successful reduction to CDH, we must cancel out the adversarial contributions. This is where the online-extraction property of the ElGamal encryption becomes crucial: it enables us to extract the secrets $S_{\text{bad},i}$ of each $\text{trx}_{\text{bad},i}$ for all $i \in [k]$, while simulation-soundness of the proofs π_0 guarantees their

¹⁰Strictly speaking, our reduction is to co-CDH in asymmetric pairing groups. For the purpose of this technical overview, we omit this subtle distinction.

¹¹This also covers the case $k = 1$, which corresponds to the notion of unpredictability for plain PVSS schemes.

well-formedness. Having extracted these, we can finally compute the secret of the honest transcript as $S_{hon} := S^* / \prod_{i \in [k]} S_{bad,i}$, thereby solving the given CDH instance.

3 Preliminaries

General Notation. We denote the security parameter by λ . For a finite set S , we use $s \leftarrow S$ to denote that s is sampled uniformly at random from S , and $|S|$ denotes the cardinality of S . Further, “ $:=$ ” denotes deterministic assignment. For a natural number $a \in \mathbb{N}$, we define $[a] := \{1, \dots, a\}$ and $\llbracket a \rrbracket := \{0, \dots, a\}$. For any $i \in [n]$ and any $\mathcal{S} \subseteq [n]$, we use $\ell_{i,\mathcal{S}} := \prod_{k \in \mathcal{S} \setminus \{i\}} k/(k-i)$ to denote the i -th Lagrange coefficient. We use standard cryptographic notions such as negligible and probabilistic polynomial-time (PPT). Unless stated otherwise, we assume all algorithms to be probabilistic and written in uppercase serif font. We use game-based security definitions, and write $\mathbf{G}^A \Rightarrow 1$ to denote the event that a security game $\mathbf{G}$ involving adversary A outputs 1.

Shamir Secret Sharing. The Shamir secret sharing [Sha79] embeds a secret $s \in \mathbb{Z}_p$ in the constant term of a polynomial $a(T) := s + a_1T + \dots + a_tT^t \in \mathbb{Z}_p[T]$, where the coefficients $a_1, \dots, a_t$ are sampled uniformly at random from $\mathbb{Z}_p$. The i -th share of the secret is $s_i := a(i)$. Any $t+1$ distinct shares suffice to reconstruct the polynomial (and hence the secret s) using Lagrange interpolation. For our constructions, we denote by $\text{Shamir}(n, t, s)$ an algorithm that outputs the n Shamir shares $s_1, \dots, s_n$ of s using a degree- t polynomial.

Linear Relations. For a binary NP relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$, we let $\mathcal{L} := \{x \mid \exists w : (x, w) \in \mathcal{R}\}$ be its associated language. We call x the statement and w the witness. A *linear relation* $\mathcal{R}_{\text{lin}}$ is a relation where the statement is a system of linear equations over some group $\mathbb{G}$ of prime order p , and the witness is a solution to that system. For this purpose, let $\mathbf{B} = \{(B_{i,1}, \dots, B_{i,N})\}_{i \in [M]} \in \mathbb{G}^{M \times N}$ and let $\mathbf{T} = (T_1, \dots, T_M) \in \mathbb{G}^M$ be a system of linear equations over $\mathbb{G}$ consisting of M equations and N variables. Further, let $\mathbf{x} = (x_1, \dots, x_N) \in \mathbb{Z}_p^N$ be an element in the N -dimensional vector space $\mathbb{Z}_p^N$ (over $\mathbb{Z}_p$). Then, we define

$$((\mathbf{B}, \mathbf{T}), \mathbf{x}) \in \mathcal{R}_{\text{lin}} \iff \forall i \in [M] : B_{i,1}^{x_1} \cdots B_{i,N}^{x_N} = T_i \quad (\text{i.e., } \mathbf{B}\mathbf{x} = \mathbf{T}).$$

Then, we have $(\mathbf{B}, \mathbf{T}) \in \mathcal{L}_{\text{lin}}$ if there exists an element $\mathbf{x} \in \mathbb{Z}_p^N$ such that $((\mathbf{B}, \mathbf{T}), \mathbf{x}) \in \mathcal{R}_{\text{lin}}$ (i.e., the linear system is satisfiable). Further, we let $\mathcal{R}_{\text{lin}}^{\mathbf{B}}$ be the relation $\mathcal{R}_{\text{lin}}$ for a fixed $\mathbf{B}$. Concretely, $(\mathbf{T}, \mathbf{x}) \in \mathcal{R}_{\text{lin}}^{\mathbf{B}}$ if $((\mathbf{B}, \mathbf{T}), \mathbf{x}) \in \mathcal{R}_{\text{lin}}$. Similarly, $\mathbf{T} \in \mathcal{L}_{\text{lin}}^{\mathbf{B}}$ if there exists an $\mathbf{x}$ such that $(\mathbf{T}, \mathbf{x}) \in \mathcal{R}_{\text{lin}}^{\mathbf{B}}$. Looking ahead, we will make use of linear relations for our non-interactive zero-knowledge proofs.

Non-Interactive Zero-Knowledge (NIZK) Proofs. A non-interactive proof system for an NP relation $\mathcal{R}$ with respect to a random oracle $\mathbf{H}$ is a pair of PPT algorithms $\text{NIZK} = (\text{NIZK.P}, \text{NIZK.V})$ such that (i) $\text{NIZK.P}^{\mathbf{H}}$ takes as input a statement x and a witness w with $(x, w) \in \mathcal{R}$ and outputs a proof π , and (ii) $\text{NIZK.V}^{\mathbf{H}}$ takes as input x and π and outputs 1 (accept) or 0 (reject). Completeness requires that honestly generated proofs for $(x, w) \in \mathcal{R}$ are always accepted. Soundness requires that no malicious prover can produce an accepting proof for a false statement $x \notin \mathcal{L}$, where $\mathcal{L}$ is the associated language. Zero-knowledge requires that there exists a simulator that can simulate proofs without knowing w by programming the random oracle $\mathbf{H}$. Most of our NIZK proofs are derived from efficient Σ -protocols [Dam10] via the Fiat–Shamir transform in the random oracle model [FS87].

Our Model. We consider a set of n parties denoted by indices $\{1, \dots, n\}$. We consider a PPT adversary A who can corrupt up to $t < n$ parties *adaptively*. Corrupted parties can deviate from the protocol specification arbitrarily, also known as *Byzantine* adversary. We assume a *plain public key infrastructure* (PKI), where each party $i \in [n]$ has a locally generated public-secret key pair $(\text{pk}_i, \text{sk}_i)$ and pk_i is known to all parties. In particular, we do not require any proofs of possession of secret keys and corrupted parties may choose their keys dependent on other keys. We do not

make any further restrictions on the network model (i.e., the network can be either synchronous or asynchronous),¹² and we do not assume secure erasures.

4 PVSS in Pairing-Free Groups

In this section, we present our non-interactive PVSS scheme over pairing-free groups **Illusi** and then give an adaptive security proof for it under the DDH assumption. We begin by giving the syntax and security definition of non-interactive PVSS schemes.

4.1 Formal Definitions

Computational Assumptions. Let GrGen be a group generation algorithm that on the input 1^λ outputs group parameters $(\mathbb{G}, p, g)$, where $\mathbb{G}$ is a cyclic group of prime order p and g is a generator.

Definition 1 (DDH Assumption). The DDH problem is (τ, ε) -hard for group generation algorithm GrGen if for any $\lambda \in \mathbb{N}$ and any algorithm A that runs in time at most τ ,

$$\begin{aligned} & |\Pr[A(\mathcal{G}, h, g^a, h^a) = 1 \mid \mathcal{G} := (\mathbb{G}, p, g) \leftarrow \text{GrGen}(1^\lambda), h \leftarrow \mathbb{G}, a \leftarrow \mathbb{Z}_p] \\ & - \Pr[A(\mathcal{G}, h, \hat{g}, \hat{h}) = 1 \mid \mathcal{G} := (\mathbb{G}, p, g) \leftarrow \text{GrGen}(1^\lambda), h, \hat{g}, \hat{h} \leftarrow \mathbb{G}]| \leq \varepsilon. \end{aligned}$$

Definition 2 (M-DDH Assumption). For fixed $k \in \mathbb{N}$, the M-DDH _{k} problem is (τ, ε) -hard for group generation algorithm GrGen if for any $\lambda \in \mathbb{N}$ and any algorithm A that runs in time at most τ ,

$$\begin{aligned} & |\Pr[A(\mathcal{G}, h, g^{a_1}, h^{a_1}, \dots, g^{a_k}, h^{a_k}) = 1 \mid \mathcal{G} := (\mathbb{G}, p, g) \leftarrow \text{GrGen}(1^\lambda), h \leftarrow \mathbb{G}, a_1, \dots, a_k \leftarrow \mathbb{Z}_p] \\ & - \Pr[A(\mathcal{G}, h, \hat{g}_1, \hat{h}_1, \dots, \hat{g}_k, \hat{h}_k) = 1 \mid \mathcal{G} := (\mathbb{G}, p, g) \leftarrow \text{GrGen}(1^\lambda), h, \hat{g}_1, \hat{h}_1, \dots, \hat{g}_k, \hat{h}_k \leftarrow \mathbb{G}]| \leq \varepsilon. \end{aligned}$$

Remark 1. The multi decisional Diffie-Hellman (M-DDH) assumption is implied by DDH [BCP02] via the relation $\varepsilon_{\text{m-ddh}_k} \leq k \cdot \varepsilon_{\text{ddh}}$, $\tau_{\text{m-ddh}_k} = \tau_{\text{ddh}}$. The proof follows from a standard hybrid argument. For improved clarity in our security proof, we will directly use the M-DDH assumption.

Syntax. A non-interactive (t, n) -threshold PVSS scheme consists of the following algorithms. Each party independently generates a public-secret key pair using the key-generation algorithm KGen . On input the public keys of the n receiving parties, a dealer runs the non-interactive sharing algorithm Share to generate a sharing transcript trx encoding a sharing of a secret S . The validity of a transcript is verified using the algorithm Ver . Upon receiving a valid transcript trx , each party i locally derives its secret share S_i by executing the decryption algorithm DecShare on input trx and its secret key sk_i . The correctness of a decrypted share S_i is verified using the algorithm DecShareVer . Finally, given any set of at least $t + 1$ valid secret shares, the deterministic reconstruction algorithm Rec reconstructs the dealer's original secret S .

Definition 3 (Non-Interactive PVSS Scheme). Let $t < n$ be natural numbers. A non-interactive (t, n) -threshold PVSS scheme consists of algorithms with the following syntax:

- $\text{Setup}(1^\lambda) \rightarrow \text{par}$: The *setup algorithm* takes the security parameter 1^λ as input, and outputs the public parameters par . We assume par implicitly defines a set of *input secrets* $\mathcal{M}_{\text{in}}$ and a set of *output secrets* $\mathcal{M}_{\text{out}}$. Further, all other algorithms implicitly take par as input.
- $\text{KGen}() \rightarrow (\text{pk}, \text{sk})$: The *key generation algorithm* outputs a secret key sk and a public key pk .
- $\text{Share}(\mathcal{L}, s) \rightarrow (\text{trx}, S)$: The *secret sharing algorithm* takes a set of n public keys $\mathcal{L} = \{\text{pk}_1, \dots, \text{pk}_n\}$ and a secret $s \in \mathcal{M}_{\text{in}}$ as input, and outputs a transcript trx and a secret $S \in \mathcal{M}_{\text{out}}$.

¹²Our PVSS schemes are non-interactive and thus we do not need to make any timing assumptions on the delivery of messages between parties. We also note that this is typically specified by the higher-level distributed protocol that makes use of the PVSS scheme as a building block.

- $\text{Ver}(\mathcal{L}, \text{trx}) \rightarrow 0/1$: The deterministic *verification algorithm* takes a set of n public keys $\mathcal{L} = \{\text{pk}_1, \dots, \text{pk}_n\}$ and a transcript trx as input, and outputs 0 (invalid) or 1 (valid).
- $\text{DecShare}(\text{trx}, i, \text{sk}_i) \rightarrow (S_i, \mu_i)$: The *share decryption algorithm* takes a transcript trx , an identity index $i \in [n]$, and a secret key sk_i as input, and outputs a secret share $S_i \in \mathcal{M}_{out}$ and a proof μ_i .
- $\text{DecShareVer}(\text{trx}, i, \text{pk}_i, S_i, \mu_i) \rightarrow 0/1$: The deterministic *share decryption verification algorithm* takes a transcript trx , an identity index $i \in [n]$, a public key pk_i , a secret share $S_i \in \mathcal{M}_{out}$, and a proof μ_i as input, and outputs 0 (invalid) or 1 (valid).
- $\text{Rec}(\mathcal{S}, \{S_i\}_{i \in \mathcal{S}}) \rightarrow S$: The deterministic *reconstruction algorithm* takes a set of $t + 1$ indices $\mathcal{S} \subseteq [n]$ and corresponding secret shares $\{S_i\}_{i \in \mathcal{S}}$ as input, and outputs a secret $S \in \mathcal{M}_{out}$.

For correctness, we require that for any $\lambda \in \mathbb{N}$, any $t, n \in \text{poly}(\lambda)$ with $t < n$, any $\text{par} \in \text{Setup}(1^\lambda)$, any $(\text{pk}_1, \text{sk}_1), \dots, (\text{pk}_n, \text{sk}_n) \in \text{KGen}()$, any $s \in \mathcal{M}_{in}$, any $(\text{trx}, S) \in \text{Share}(\mathcal{L}, s)$, any $\mathcal{S} \subseteq [n]$ with $|\mathcal{S}| = t + 1$, and any $i \in \mathcal{S}$, we have

$$\text{Ver}(\mathcal{L}, \text{trx}) = 1, \quad \text{DecShareVer}(\text{trx}, i, \text{pk}_i, S_i, \mu_i) = 1, \quad \text{Rec}(\mathcal{S}, \{S_i\}_{i \in \mathcal{S}}) = S,$$

where $\mathcal{L} := \{\text{pk}_1, \dots, \text{pk}_n\}$ and $(S_i, \mu_i) := \text{DecShare}(\text{trx}, i, \text{sk}_i)$.

Public Verifiability. We define public verifiability of non-interactive PVSS schemes via the security game PV described in Fig. 1. Intuitively, public verifiability requires that any verifying transcript trx —even one generated maliciously—uniquely determines a secret $S \in \mathcal{M}_{out}$ that is reconstructed by any set $\mathcal{S} \subseteq [n]$ of $t + 1$ parties, even in the presence of malicious parties that may provide arbitrary but verifying decryption shares (S_i, μ_i) during the reconstruction phase.¹³ That is, except with negligible probability, any deviation by the dealer from the prescribed transcript generation procedure, as well as any attempt by malicious parties to provide invalid or inconsistent shares, is detected. Note that our notion of public verifiability is only computational, in contrast to correctness, which holds perfectly.

In the corresponding security game, the adversary is first given honestly generated public keys, after which it may specify a set of initially corrupted parties together with updated public keys for those parties. Subsequently, the adversary outputs a transcript trx that verifies with respect to the resulting public-key list and is then provided with honestly generated decryption shares of all honest parties. The adversary next specifies two sets $\mathcal{S}_1, \mathcal{S}_2 \subseteq [n]$ of size $t + 1$, together with decryption shares (S_j, μ_j) for all corrupted parties $j \in \mathcal{C} \cap (\mathcal{S}_1 \cup \mathcal{S}_2)$. Finally, the secrets S and S' are reconstructed from the shares corresponding to $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively, and the adversary wins the game if and only if $S \neq S'$.

Definition 4 (Public Verifiability of Non-interactive PVSS Scheme). A non-interactive (t, n) -threshold PVSS scheme PVSS is (τ, ε) -publicly verifiable if for every $\lambda \in \mathbb{N}$ and every adversary A that runs in time at most $\tau = \tau(\lambda)$, we have $\Pr[\text{PV}_{\text{PVSS}}^A(1^\lambda) \Rightarrow 1] \leq \varepsilon = \varepsilon(\lambda)$, where the security game PV is defined in Fig. 1.

Security. We define the security of non-interactive PVSS schemes via the established notion of indistinguishability following Cascudo-David [CD17] with slight adaptations to the adaptive security model and our syntax. The respective security game Ind is described in Fig. 2. At an intuitive level, the notion of indistinguishability (of secrets) states that the sharing phase (generating a transcript trx) together with the secret keys of any set of t parties gives no information about the shared secret (i.e., encoded in trx).

In more detail, let A be the adversary in this game. At the beginning, individual public-secret key pairs are generated honestly for all parties, and A is given the respective public keys $\{\text{pk}_i\}_{i \in [n]}$ as

¹³Naturally, the adversarially provided shares must pass verification; otherwise, the adversary could trivially submit invalid shares.

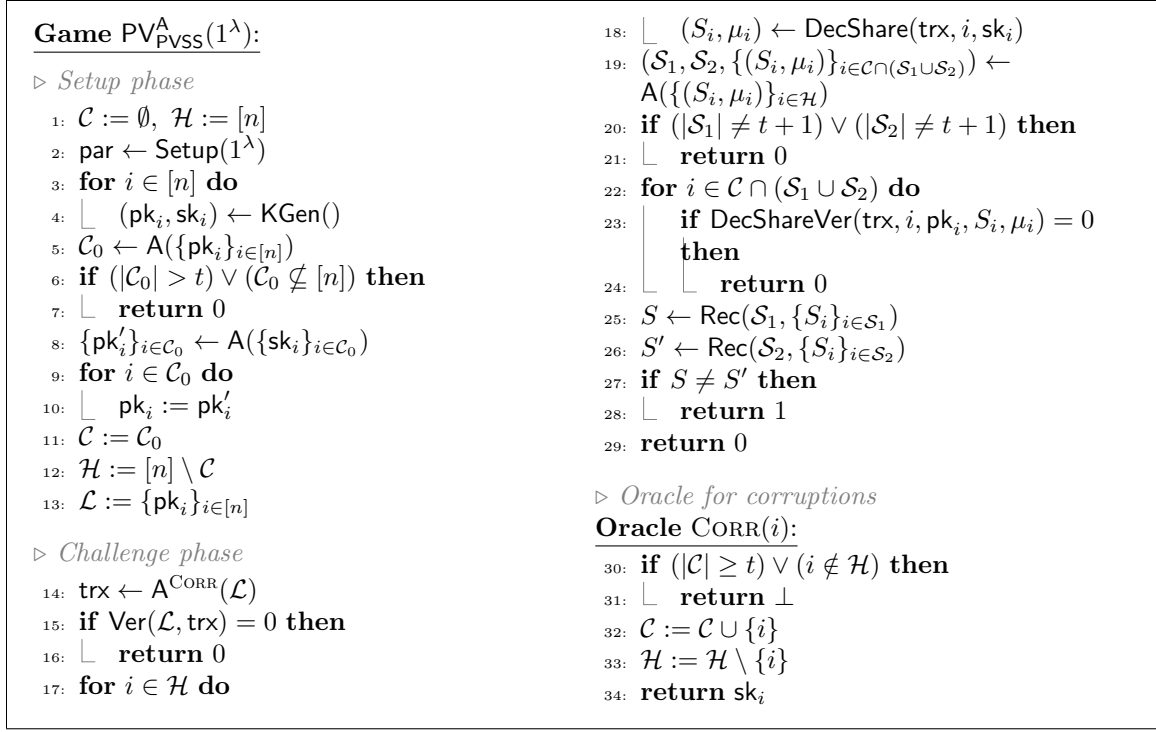


Figure 1: Public verifiability game PV for non-interactive PVSS scheme PVSS against adversary A.

input. Then, A returns a (possibly empty) set of initially corrupt parties $\mathcal{C}_0 \subseteq [n]$ of size at most t . A receives their respective secret keys $\{\text{sk}_i\}_{i \in \mathcal{C}_0}$, and then returns updated public keys $\text{pk}_i := \text{pk}'_i$ for $i \in \mathcal{C}_0$. Notably, these keys can be generated in any arbitrary and even malicious way. Henceforth, A can corrupt further parties throughout the protocol execution using the oracle CORR, but in total at most t (including the parties in $\mathcal{C}_0$). Upon corrupting an honest party $i \in \mathcal{H}$, A learns its secret key sk_i . In the challenge phase, two uniformly random secrets $s_0 \in \mathcal{M}_{\text{in}}$, $S_1 \in \mathcal{M}_{\text{out}}$ are sampled.¹⁴ Further, a uniformly random bit is sampled $b \leftarrow \{0, 1\}$, and then the secret sharing algorithm is run on input $(\mathcal{L}, s_0)$ to obtain a transcript trx (with encoded secret S_0). Here, $\mathcal{L}$ is the list of all public keys $\{\text{pk}_i\}_{i \in [n]}$. Finally, A is given the transcript trx along with the secret $S_b \in \mathcal{M}_{\text{out}}$, and wins the game if it returns the bit b .

Definition 5 (Security of Non-interactive PVSS Scheme). A non-interactive (t, n) -threshold PVSS scheme PVSS is (τ, ε) -secure if for every $\lambda \in \mathbb{N}$ and every adversary A that runs in time at most $\tau = \tau(\lambda)$, we have $\Pr[\text{Ind}_{\text{PVSS}}^{\text{A}}(1^\lambda) \Rightarrow 1] \leq \varepsilon = \varepsilon(\lambda)$, where the security game Ind is defined in Fig. 2.

Remark 2. We note that this security notion corresponds to Ind1 in the literature [RV05]. The stronger notion of Ind2-security allows the adversary to choose the two secrets s_0, S_1 itself. However, a well-known black-box transformation lifts any Ind1-secure scheme to Ind2-security [RV05, HV09]. The idea is to use a randomly shared secret as a one-time pad to encrypt the target secret. Following prior works [CD17, BSKN23], we thus restrict our attention to Ind1-security in this work.

¹⁴Note that one secret is sampled from the set of input secrets, while the other is sampled directly from the set of output secrets. This is necessary in order to guarantee efficient computation of the output transcript trx .

Game $\text{Ind}_{\text{PVSS}}^{\text{A}}(1^\lambda)$: ▷ <i>Setup phase</i> <pre> 1: $\mathcal{C} := \emptyset, \mathcal{H} := [n]$ 2: $\text{par} \leftarrow \text{Setup}(1^\lambda)$ 3: for $i \in [n]$ do 4: $(\text{pk}_i, \text{sk}_i) \leftarrow \text{KGen}()$ 5: $\mathcal{C}_0 \leftarrow \text{A}(\{\text{pk}_i\}_{i \in [n]})$ 6: if $(\mathcal{C}_0 > t) \vee (\mathcal{C}_0 \not\subseteq [n])$ then 7: return 0 8: $\{\text{pk}'_i\}_{i \in \mathcal{C}_0} \leftarrow \text{A}(\{\text{sk}_i\}_{i \in \mathcal{C}_0})$ 9: for $i \in \mathcal{C}_0$ do 10: $\text{pk}_i := \text{pk}'_i$ 11: $\mathcal{C} := \mathcal{C}_0$ 12: $\mathcal{H} := [n] \setminus \mathcal{C}$ 13: $\mathcal{L} := \{\text{pk}_i\}_{i \in [n]}$ </pre>	▷ <i>Challenge phase</i> <pre> 14: $s_0 \leftarrow \mathcal{M}_{in}, S_1 \leftarrow \mathcal{M}_{out}$ 15: $b \leftarrow \{0, 1\}$ 16: $(\text{trx}, S_0) \leftarrow \text{Share}(\mathcal{L}, s_0)$ 17: $b^* \leftarrow \text{A}^{\text{CORR}}(\mathcal{L}, \text{trx}, S_b)$ 18: if $b^* = b$ then 19: return 1 20: return 0 </pre> ▷ <i>Oracle for corruptions</i> Oracle $\text{CORR}(i)$: <pre> 21: if $(\mathcal{C} \geq t) \vee (i \notin \mathcal{H})$ then 22: return $\perp$ 23: $\mathcal{C} := \mathcal{C} \cup \{i\}$ 24: $\mathcal{H} := \mathcal{H} \setminus \{i\}$ 25: return sk_i </pre>
---	--

Figure 2: The indistinguishability security game Ind for non-interactive PVSS scheme PVSS against adversary A.

4.2 Our Scheme

We give a verbal description of our non-interactive PVSS scheme **Illusi** next and present it formally as pseudocode in Fig. 3.

Setup and Key Generation. The public parameters **par** of the scheme consist of group parameters $(\mathbb{G}, p, g)$ generated by **GrGen** and two uniformly random group elements $h, u \in \mathbb{G}$.¹⁵ It also implicitly defines the set of input secrets as $\mathbb{Z}_p$ and the set of output secrets as $\mathbb{G}$. The key generation algorithm **KGen** samples $x, y \leftarrow \mathbb{Z}_p$ uniformly at random. The public key is computed as $\text{pk} := g^x u^y \in \mathbb{G}$, and the secret key is $\text{sk} := (x, y) \in \mathbb{Z}_p^2$.

Secret Sharing. The dealer wants to (t, n) -secret-share the output secret $S := h^s \in \mathbb{G}$ (corresponding to the input secret $s \in \mathbb{Z}_p$). To this end, it first samples a uniformly random degree- t polynomial¹⁶ $a(T) \in \mathbb{Z}_p[T]$ such that $a(0) = s$. This defines the n values $s_i := a(i) \in \mathbb{Z}_p, i \in [n]$. The transcript $\text{trx} := (\text{com}, \{c_i, \pi_i\}_{i \in [n]})$ now consists of the following components:

- *Commitments:* This is a commitment to the polynomial shares, $\text{com} := (g^s, g^{s_1}, \dots, g^{s_n})$.
- *Encryptions:* These are the encrypted secret shares of parties. For each party $i \in [n]$, the dealer samples some uniform randomness $r_i \leftarrow \mathbb{Z}_p$ and encrypt party i 's secret share h^{s_i} towards its public key pk_i as $c_i := (g^{r_i}, u^{r_i}, \text{pk}_i^{r_i} h^{s_i})$.
- *NIZK Proofs:* Further, for each $i \in [n]$, a non-interactive zero-knowledge proof π_i of correct encryption is computed (derived from an efficient Σ -protocol). Concretely, it proves a statement in the relation $\mathcal{R}_{\text{lin}}$, i.e., the satisfiability of the following system of linear equations $(\mathbf{B}_i, \mathbf{T}_i)$, using (s_i, r_i) as the witness:

$$\mathbf{B}_i := \begin{pmatrix} g & \mathbb{1} \\ \mathbb{1} & g \\ \mathbb{1} & u \\ h & \text{pk}_i \end{pmatrix}, \quad \mathbf{T}_i := \begin{pmatrix} \text{com}[i] \\ c_{i,1} \\ c_{i,2} \\ c_{i,3} \end{pmatrix}. \quad (1)$$

¹⁵We also assume that it implicitly includes the random oracles from the NIZK proof systems.

¹⁶We use T as indeterminate for the one-dimensional polynomial ring over $\mathbb{Z}_p$.

Setup(1^λ): <pre> 1: $(\mathbb{G}, p, g) \leftarrow \text{GrGen}(1^\lambda)$ 2: $h, u \leftarrow \mathbb{G}$ 3: return $\text{par} := (\mathbb{G}, p, g, h, u)$ $\triangleright$ Secrets sets $\mathcal{M}_{in} := \mathbb{Z}_p$, $\mathcal{M}_{out} := \mathbb{G}$ $\triangleright$ All algorithms take par as input KGen($\cdot$): <pre> 4: $x, y \leftarrow \mathbb{Z}_p$ 5: $\text{sk} := (x, y) \in \mathbb{Z}_p^2$ 6: $\text{pk} := g^x u^y \in \mathbb{G}$ 7: return (pk, sk) Share($\mathcal{L}, s$): <pre> 8: $\{\text{pk}_1, \dots, \text{pk}_n\} := \mathcal{L}$ 9: $\{s_1, \dots, s_n\} \leftarrow \text{Shamir}(n, t, s)$ 10: $\text{com} := (g^s, g^{s_1}, \dots, g^{s_n})$ 11: for $i \in [n]$ do 12: $r_i \leftarrow \mathbb{Z}_p$ 13: $c_i := (g^{r_i}, u^{r_i}, \text{pk}_i^{r_i} h^{s_i})$ 14: $\text{stm} := (\mathbf{B}_i, \mathbf{T}_i) \triangleright \text{Eq. (1)}$ 15: $\pi_i := \text{NIZK.P}_1^H(\text{stm}, (s_i, r_i)) \triangleright \text{Fig. 8}$ 16: $\text{trx} := (\text{com}, \{c_i, \pi_i\}_{i \in [n]})$ 17: return trx Ver($\mathcal{L}, \text{trx}$): <pre> 18: $\{\text{pk}_1, \dots, \text{pk}_n\} := \mathcal{L}$ 19: $(\text{com}, \{c_i, \pi_i\}_{i \in [n]}) := \text{trx}$ </pre> </pre></pre></pre>	<pre> 20: for $i \in [n]$ do 21: $\text{stm} := (\mathbf{B}_i, \mathbf{T}_i) \triangleright \text{Eq. (1)}$ 22: if $\text{NIZK.V}_1^H(\text{stm}, \pi_i) = 0$ then $\triangleright \text{Fig. 8}$ 23: $\text{return } 0$ 24: return $\text{degTest}(\text{com}, n, t) \triangleright \text{Fig. 7}$ DecShare($\text{trx}, i, \text{sk}_i$): <pre> 25: $(\text{com}, \{c_i, \pi_i\}_{i \in [n]}) := \text{trx}$ 26: $(c_1, c_2, c_3) := c_i$ 27: $(x_i, y_i) := \text{sk}_i$ 28: $S_i := c_3 / (c_1^{x_i} c_2^{y_i})$ 29: $\text{stm} := (\mathbf{B}'_i, \mathbf{T}'_i) \triangleright \text{Eq. (2)}$ 30: $\mu_i := \text{NIZK.P}_2^H(\text{stm}, \text{sk}_i) \triangleright \text{Fig. 9}$ 31: return (S_i, μ_i) DecShareVer($\text{trx}, i, \text{pk}_i, S_i, \mu_i$): <pre> 32: $(\text{com}, \{c_i, \pi_i\}_{i \in [n]}) := \text{trx}$ 33: $(c_1, c_2, c_3) := c_i$ 34: $\text{stm} := (\mathbf{B}'_i, \mathbf{T}'_i) \triangleright \text{Eq. (2)}$ 35: if $\text{NIZK.V}_2^H(\text{stm}, \mu_i) = 0$ then $\triangleright \text{Fig. 9}$ 36: $\text{return } 0$ 37: return 1 Rec($S, \{S_i\}_{i \in \mathcal{S}}$): <pre> 38: for $i \in \mathcal{S}$ do 39: $\ell_{i, \mathcal{S}} := \prod_{k \in \mathcal{S} \setminus \{i\}} \frac{k}{k-i}$ 40: $S := \prod_{i \in \mathcal{S}} S_i^{\ell_{i, \mathcal{S}}}$ 41: return S </pre> </pre></pre></pre>
--	--

Figure 3: Our non-interactive PVSS scheme Illusi.

We use the notation $\pi_i = \text{NIZK.P}_1^H(\text{stm}, (s_i, r_i))$ to denote this, where $\text{stm} := (\mathbf{B}_i, \mathbf{T}_i)$ and H is the underlying random oracle. We refer to Fig. 8 for a detailed description of the NIZK proof.

Verification. A given transcript $\text{trx} := (\text{com}, \{c_i, \pi_i\}_{i \in [n]})$ is verified with respect to the list of n public keys $\mathcal{L} := \{\text{pk}_1, \dots, \text{pk}_n\}$ as follows. First, the verifier checks that the commitment com indeed commits to the evaluations of a polynomial of degree (at most) t via the standard low-degree test from Cascudo-David [CD17] (see Fig. 7). If the check passes, the verifier next checks that the encryptions c_i for all $i \in [n]$ are valid. For this, it verifies the NIZK proof π_i against the statement $\text{stm} := (\mathbf{B}_i, \mathbf{T}_i)$ (see Eq. (1) above) by invoking $\text{NIZK.V}_1^H(\text{stm}, \pi_i)$. The transcript trx verifies if and only if all these checks are successful.

Decryption. Given a transcript $\text{trx} := (\text{com}, \{c_i, \pi_i\}_{i \in [n]})$, a receiving party $i \in [n]$ with secret key $\text{sk}_i := (x_i, y_i) \in \mathbb{Z}_p^2$ extracts its share $S_i \in \mathbb{G}$ of the secret $S \in \mathbb{G}$ as follows. It first parses the encrypted share c_i into its components as $(c_1, c_2, c_3) := c_i$ and then computes S_i using its secret key as $S_i := c_3 / (c_1^{x_i} c_2^{y_i})$ (which is supposed to be h^{s_i}). Further, it computes a NIZK proof μ_i of correct decryption using its secret key sk_i as witness. Concretely, it proves a statement in the relation $\mathcal{R}_{\text{lin}}$, i.e., the satisfiability of the following system of linear equations $(\mathbf{B}'_i, \mathbf{T}'_i)$, using $\text{sk}_i = (x_i, y_i)$ as the witness:

$$\mathbf{B}'_i := \begin{pmatrix} g & u \\ c_1 & c_2 \end{pmatrix}, \quad \mathbf{T}'_i := \begin{pmatrix} \text{pk}_i \\ c_3 / S_i \end{pmatrix}. \quad (2)$$

That is, the proof is computed as $\mu_i := \text{NIZK.P}_2^H(\text{stm}, \text{sk}_i)$, where $\text{stm} := (\mathbf{B}'_i, \mathbf{T}'_i)$. We refer to Fig. 9 for a detailed description of the NIZK proof.

Decryption Share Verification. Given a transcript $\text{trx} := (\text{com}, \{c_i, \pi_i\}_{i \in [n]})$, an index $i \in [n]$, and a public key pk_i , a share S_i is verified using the accompanied proof μ_i of correct decryption. Specifically, one verifies the NIZK proof μ_i against the statement $\text{stm} := (\mathbf{B}'_i, \mathbf{T}'_i)$ (see Eq. (2) above) by invoking $\text{NIZK.V}_2^H(\text{stm}, \mu_i)$. The share S_i is correct if and only if this check is successful.

Reconstruction. On input a set of $t + 1$ secret shares $\{S_i\}_{i \in \mathcal{S}}$, the full secret S is reconstructed via standard Lagrange interpolation in the exponent. Specifically, the combiner computes Lagrange coefficients $\ell_{i,\mathcal{S}} := \prod_{k \in \mathcal{S} \setminus \{i\}} \frac{k}{k-i} \in \mathbb{Z}_p$ for $i \in \mathcal{S}$, and outputs the secret as $S := \prod_{i \in \mathcal{S}} S_i^{\ell_{i,\mathcal{S}}}$.

4.3 Security Analysis

We give an adaptive security proof for *Illusi* under the DDH assumption. Further, we note that correctness and public verifiability of our scheme follow from standard arguments, and we therefore defer their formal proofs to Appendix D. For completeness, we provide a high-level discussion and intuition for these properties at the end of this section.

Theorem 1. *If the DDH problem is $(\tau_{\text{ddh}}, \varepsilon_{\text{ddh}})$ -hard for group generation algorithm GrGen, then Illusi is a $(\tau_{\text{ind}}, \varepsilon_{\text{ind}})$ -secure non-interactive (t, n) -threshold PVSS scheme in the random oracle model against any adversary that makes at most q_h hash queries, where $\tau_{\text{ddh}} \approx \tau_{\text{ind}}$ and*

$$\varepsilon_{\text{ind}} \leq (n + t + 1) \cdot \varepsilon_{\text{ddh}} + \frac{n \cdot (q_h + 1) + 1}{p} + \frac{1}{2}.$$

Proof. We structure the proof as a sequence of hybrid games, where any two subsequent games are indistinguishable. In the final game, we will argue that winning the game is an unconditionally hard problem, which concludes our proof. For this purpose, we assume an adversary $\mathbf{A}$ that breaks the $(\tau_{\text{ind}}, \varepsilon_{\text{ind}})$ -security of *Illusi*.

GAME $\mathbf{G}_0$: This is the real security game $\text{Ind}_{\text{PVSS}}^A$ for non-interactive PVSS schemes (cf. Fig. 2). We briefly recall the game to fix some notation. At the beginning, the game samples public parameters $\text{par} := (\mathbb{G}, p, g, h, u) \leftarrow \text{Setup}(1^\lambda)$, where $h, u \leftarrow \mathbb{G}$ are sampled uniformly at random. This also defines the set of input secrets $\mathcal{M}_{\text{in}} := \mathbb{Z}_p$ and the set of output secrets $\mathcal{M}_{\text{out}} := \mathbb{G}$. Further, for all $i \in [n]$, the game samples $x_i, y_i \leftarrow \mathbb{Z}_p$ for a secret key $\text{sk}_i := (x_i, y_i) \in \mathbb{Z}_p^2$ and public key $\text{pk}_i := g^{x_i} u^{y_i} \in \mathbb{G}$. Then, the game runs the adversary $\mathbf{A}$ on input $(\text{par}, \mathcal{L})$ where $\mathcal{L} := \{\text{pk}_1, \dots, \text{pk}_n\}$, upon which $\mathbf{A}$ outputs a set of initially corrupted parties $\mathcal{C}_0 \subseteq [n]$ of size at most t along with new public keys $\text{pk}_i := \text{pk}'_i$ for $i \in \mathcal{C}_0$. The game returns the secret keys sk_i for all $i \in \mathcal{C}_0$ and initializes the set of corrupted parties $\mathcal{C} := \mathcal{C}_0$ and honest parties $\mathcal{H} := [n] \setminus \mathcal{C}$, respectively. Henceforth, the adversary gets access to a corruption oracle and a random oracle (underlying the NIZK proof systems). Importantly, the corruption oracle CORR returns the secret key sk_i of the queried (honest) party i and can be called at most $t - |\mathcal{C}_0|$ times.¹⁷ In the challenge phase, the game samples two uniformly random secrets $s_0 \leftarrow \mathbb{Z}_p$, $S_1 \leftarrow \mathbb{G}$ along with a uniformly random bit $b \leftarrow \{0, 1\}$. It then runs the secret sharing algorithm on the secret s_0 as $(\text{trx}, S_0) \leftarrow \text{Share}(\mathcal{L}, s_0)$. The adversary is then given the transcript trx (corresponding to output secret S_0) along with the secret S_b . Finally, $\mathbf{A}$ outputs a guess b^* and wins the game if and only if $b^* = b$. Clearly, we have

$$\Pr[\mathbf{G}_0 \Rightarrow 1] = \varepsilon_{\text{ind}}.$$

Henceforth, we will focus on how to generate the PVSS transcript for the secret $s := s_0 \in \mathbb{Z}_p$. Our plan is to reach a game where $S := h^s$ is replaced by $S' := h^{s'}$ for an independent and uniformly

¹⁷This is to ensure that the adversary corrupts at most t parties throughout the game.

random $s' \in \mathbb{Z}_p$. The transcript then shares S' which is independent of $b \in \{0, 1\}$, and it is clear that the distinguishing game becomes unconditionally hard.

GAME $\mathbf{G}_1$: This game is identical to the previous game, except we now use simulated NIZK proofs (of correct encryption) $\bar{\pi}_i$ for all $i \in [n]$ to compute the transcript $\text{trx} := (\text{com}, \{c_i, \bar{\pi}_i\}_{i \in [n]})$. Concretely, for each $i \in [n]$, we use the special honest-verifier zero-knowledge (SHVZK) simulation to compute $\bar{\pi}_i$. Clearly, the games $\mathbf{G}_1$ and $\mathbf{G}_0$ are indistinguishable until we generate a commitment $\text{cmt} \in \mathbb{G}^3$ for the simulated proof $\bar{\pi}_i$ that has already been queried to the random oracle $\mathbf{H}$ (which underlies the non-interactive proof systems). However, since the commitment is distributed uniformly over a space of size at least p and hidden from $\mathcal{A}$ (until we output the NIZK proof), the probability that the SHVZK simulation fails is at most q_h/p . By a union bound over all indices $i \in [n]$, we get

$$|\Pr[\mathbf{G}_1 \Rightarrow 1] - \Pr[\mathbf{G}_0 \Rightarrow 1]| \leq \frac{n \cdot q_h}{p}.$$

GAME $\mathbf{G}_2$: In this game, we change the secret shares of parties. Namely, for each $i \in [n]$, instead of encrypting the real secret share $h^{a(i)}$, we encrypt the share $h^{b(i)}$, where $b(T) \in \mathbb{Z}_p[T]$ is another degree- t polynomial sampled uniformly at random alongside $a(T)$, with the constraint that $b(0) = s$ (recall that also $a(0) = s$). We bound the distinguishing advantage between games $\mathbf{G}_2$ and $\mathbf{G}_1$ via a reduction $\mathbf{R}$ to the M-DDH $_t$ assumption. To this end, $\mathbf{R}$ gets a tuple $\mathcal{D}_t := (g, h, \hat{g}_1, \hat{h}_1, \dots, \hat{g}_t, \hat{h}_t)$ as input and then simulates game $\mathbf{G}_1$ for $\mathbf{A}$ with the following changes. Recall that in game $\mathbf{G}_1$, we first sample a uniformly random degree- t polynomial $a(T) \leftarrow \mathbb{Z}_p[T]$ with $a(0) = s$, and then compute the polynomial commitment and encrypted secret shares as $\text{com} := (g^s, g^{a(1)}, \dots, g^{a(n)})$ and

$$\forall i \in [n] : S_i := h^{a(i)}, \quad r_i \leftarrow \mathbb{Z}_p, \quad c_i := (g^{r_i}, u^{r_i}, \text{pk}_i^{r_i} S_i),$$

along with simulated NIZK proofs of correct encryption $\bar{\pi}_i$ for $i \in [n]$. Note that com and S_i are computed using knowledge of the polynomial shares $a(i)$, while the simulated proof $\bar{\pi}_i$ does not use the witness $(a(i), r_i)$ anymore. The reduction $\mathbf{R}$ changes this as follows. First, it samples $s \leftarrow \mathbb{Z}_p$ as before and then computes

$$\forall i \in [n] : \quad \text{com}[i] := g^s \cdot \prod_{j \in [t]} \hat{g}_j^{i^j}, \quad S_i := h^s \cdot \prod_{j \in [t]} \hat{h}_j^{i^j}.$$

Essentially, it defines a degree- t polynomial in the exponent using the base $(g, \hat{g}_1, \dots, \hat{g}_t)$, and then computes its evaluations in the exponent on the points $\{1, \dots, n\}$. This allows to generate the polynomial commitment com . For the secret shares S_i , it does the same but using the base $(h, \hat{h}_1, \dots, \hat{h}_t)$. The rest remains unchanged: NIZK proofs of correct encryption are simulated (without knowledge of the witness), and encryptions c_i for $i \in [n]$ are generated by sampling uniform randomness $r_i \leftarrow \mathbb{Z}_p$ and computing $c_i := (g^{r_i}, u^{r_i}, \text{pk}_i^{r_i} S_i)$. In the end, $\mathbf{R}$ outputs whatever the game outputs. Clearly, $\mathbf{R}$ perfectly simulates game $\mathbf{G}_1$ for the adversary if $\mathcal{D}_t$ is a multi-Diffie-Hellman (M-DH) tuple. On the other hand, if $\mathcal{D}_t$ is *not* an M-DH tuple, then $\mathbf{R}$ perfectly simulates game $\mathbf{G}_2$. Using Remark 1, we therefore get

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq \varepsilon_{\text{m-ddh}_t} \leq t \cdot \varepsilon_{\text{ddh}}.$$

GAME $\mathbf{G}_3$: In this game, we change how we compute the encryptions for “initially honest” parties. Namely, for each $i \in \mathcal{H}_0$,¹⁸ we compute the encryption c_i in the alternative way using the secret key $\text{sk}_i = (x_i, y_i)$ as follows:

$$S_i := h^{b(i)}, \quad r_i \leftarrow \mathbb{Z}_p, \quad c_i := (c_1, c_2, c_3) := (g^{r_i}, u^{r_i}, c_1^{x_i} c_2^{y_i} S_i).$$

¹⁸Recall that $\mathcal{H}_0 := [n] \setminus \mathcal{C}_0$, where $\mathcal{C}_0$ defines the set of initially corrupted parties. We therefore call $\mathcal{H}_0$ the set of *initially honest* parties.

It is clear that c_3 is identically distributed as in the previous game, and therefore we get

$$\Pr[\mathbf{G}_3 \Rightarrow 1] = \Pr[\mathbf{G}_2 \Rightarrow 1].$$

GAME $\mathbf{G}_4$: In this game, we change the encryptions of initially honest parties. Namely, for each $i \in \mathcal{H}_0$, we generate the encryption c_i differently as:

$$S_i := h^{b(i)}, \quad r_i, r'_i \leftarrow \mathbb{Z}_p, \quad c_i := (c_1, c_2, c_3) := (g^{r_i}, u^{r'_i}, c_1^{x_i} c_2^{y_i} S_i).$$

We bound the distinguishing advantage between games $\mathbf{G}_4$ and $\mathbf{G}_3$ via a reduction R to M-DDH_n . To this end, R gets a tuple $\mathcal{D}_n := (g, u, \hat{g}_1, \hat{u}_1, \dots, \hat{g}_n, \hat{u}_n)$ as input and then simulates game $\mathbf{G}_3$ for A with the following change.¹⁹ For each $i \in \mathcal{H}$, it generates the encryption c_i as:

$$S_i := h^{b(i)}, \quad c_i := (c_1, c_2, c_3) := (\hat{g}_i, \hat{u}_i, c_1^{x_i} c_2^{y_i} S_i).$$

The rest remains unchanged: NIZK proofs of correct encryption are simulated (without knowledge of the witness), and encryptions c_i for $i \in \mathcal{C}_0$ are generated by sampling uniform randomness $r_i \leftarrow \mathbb{Z}_p$ and computing $c_i := (g^{r_i}, u^{r_i}, \text{pk}_i^{r_i} S_i)$. In the end, R outputs whatever the game outputs. Clearly, R perfectly simulates game $\mathbf{G}_3$ for the adversary if $\mathcal{D}_n$ is an M-DH tuple. On the other hand, if $\mathcal{D}_n$ is *not* an M-DH tuple, then R perfectly simulates game $\mathbf{G}_4$. Using Remark 1, we therefore get

$$|\Pr[\mathbf{G}_4 \Rightarrow 1] - \Pr[\mathbf{G}_3 \Rightarrow 1]| \leq \varepsilon_{\text{m-ddh}_n} \leq n \cdot \varepsilon_{\text{ddh}}.$$

GAME $\mathbf{G}_5$: In this game, we defer the generation of the secret key values x_i, y_i for all $i \in [n]$ until party i is corrupted. Recall that in the previous game, we reference the secret key $\text{sk}_i = (x_i, y_i)$ at two places: (i) During the key generation, and (ii) to generate the encryption c_i (note that the latter is only true for initially honest parties $i \in \mathcal{H}_0$). We change this as follows. First, in the setup phase, we generate the public parameter u by sampling $\delta_u \leftarrow \mathbb{Z}_p$ uniformly at random and then setting $u := g^{\delta_u}$. Then, in the key generation phase, for each $i \in [n]$, we generate the public key pk_i directly from its distribution, without referencing x_i and y_i , by sampling $\alpha_i \leftarrow \mathbb{Z}_p$ uniformly at random and then setting $\text{pk}_i := g^{\alpha_i}$. Finally, in the challenge phase, for each $i \in \mathcal{H}_0$, we generate the encryption $c_i[2]$ (i.e., the third component of the element $c_i \in \mathbb{G}^3$) by sampling $\tilde{\gamma}_i \leftarrow \mathbb{Z}_p$ uniformly at random and then setting $c_i[2] := g^{\tilde{\gamma}_i} h^{b(i)}$. Upon a corruption query $\text{CORR}(i)$ for $i \in \mathcal{H}_0$, we can determine $\text{sk}_i = (x_i, y_i)$ using the identities²⁰

$$g^{\alpha_i} = \text{pk}_i = g^{x_i} u^{y_i}, \quad g^{\tilde{\gamma}_i} = c_1^{x_i} c_2^{y_i}.$$

To this end, we use the values $\delta_u, r_i, r'_i \in \mathbb{Z}_p$ to solve for the tuple (x_i, y_i) as a solution to (considering the discrete logarithm values to base g of the above identities):

$$\alpha_i = x_i + \delta_u y_i, \quad \tilde{\gamma}_i = r_i x_i + \delta_u r'_i y_i.$$

We note that this equation is uniquely solvable in (x_i, y_i) , unless $r_i = r'_i$, which is easy to see.

We claim that the distribution of outputs in both games $\mathbf{G}_5$ and $\mathbf{G}_4$ is identical, unless $\delta_u = 0$ or $r_i = r'_i$ for some $i \in \mathcal{H}_0$. In the following, we condition on the event E that $\delta_u \neq 0$ and $r_i \neq r'_i$ for all $i \in \mathcal{H}_0$. First, it is clear that the distribution of public keys pk_i is identical in both games (and this is true even if the prescribed event E does not occur). Second, for each $i \in \mathcal{H}_0$, given the public key $\text{pk}_i =: g^{\alpha_i}$, there are p possible secret keys (x_i, y_i) , and each possible secret key yields a different value $c_1^{x_i} c_2^{y_i} \in \mathbb{G}$. This can be seen as follows:

$$c_1^{x_i} c_2^{y_i} = g^{r_i x_i} u^{r'_i y_i} = g^{r_i x_i + \delta_u r'_i y_i} = g^{r_i x_i + r'_i (\alpha_i - x_i)} = g^{r'_i \alpha_i + x_i (r_i - r'_i)},$$

¹⁹It is meant to be understood that R embeds u into the public parameters as the element “ u ”.

²⁰For parties $i \in \mathcal{C}_0$, we can simply output any of the p possible secret keys (x_i, y_i) satisfying the first identity.

and observe that the mapping $x_i \mapsto x_i(r_i - r'_i)$ defines an automorphism on $\mathbb{Z}_p$, since $r_i \neq r'_i$. In more detail, a different tuple (x_i, y_i) necessarily yields a different x_i , since $\delta_u \neq 0$ and $\alpha_i = x_i + \delta_u y_i$, and this in turn yields a different $x_i(r_i - r'_i)$. As a consequence, for each $i \in \mathcal{H}_0$, the element $c_i[2]$ (i.e., the third component of the encryption $c_i \in \mathbb{G}^3$) is uniformly random for given $\text{pk}_i, c_i[0], c_i[1]$. Further, since the set of elements $\{\text{pk}_i, r_i, r'_i\}$ is sampled independently among varying $i \in \mathcal{H}_0$, the elements $c_i[2]$ are also independent among varying $i \in \mathcal{H}_0$. Having said all that, our claim on the identical distributions of outputs in both games (conditioned on the event E) follows. Finally, we can easily bound the probability of event E occurring by

$$\Pr[\delta_u = 0] + \Pr[\exists i \in \mathcal{H}_0 : r_i = r'_i] = \frac{1}{p} + |\mathcal{H}_0| \cdot \Pr[r_i = r'_i] \leq \frac{1}{p} + n \cdot \frac{1}{p}.$$

Therefore, we get

$$|\Pr[\mathbf{G}_5 \Rightarrow 1] - \Pr[\mathbf{G}_4 \Rightarrow 1]| \leq \frac{n+1}{p}.$$

GAME $\mathbf{G}_6$: In this game, we defer the generation of the secret share $h^{b(i)}$ for all $i \in \mathcal{H}_0$ until party i is corrupted. Recall that in the previous game, we use $h^{b(i)}$ to generate the encryption $c_i[2]$ (i.e., the third component of the element $c_i \in \mathbb{G}^3$) by sampling $\tilde{\gamma}_i \leftarrow \mathbb{Z}_p$ uniformly at random and then setting $c_i[2] := g^{\tilde{\gamma}_i} h^{b(i)}$. From now on, for each $i \in \mathcal{H}_0$, we generate the encryption $c_i[2]$ by sampling $\gamma_i \leftarrow \mathbb{Z}_p$ uniformly at random and then setting $c_i[2] := g^{\gamma_i}$. Upon a corruption query $\text{CORR}(i)$ for $i \in \mathcal{H}_0$, we first define $h^{b(i)} := g^{\beta_i}$ for a uniformly sampled $\beta_i \leftarrow \mathbb{Z}_p$ and then determine $\text{sk}_i = (x_i, y_i)$ as a solution to

$$\alpha_i = x_i + \delta_u y_i, \quad \gamma_i = r_i x_i + \delta_u r'_i y_i + \beta_i.$$

We claim that the distribution of outputs in both games $\mathbf{G}_6$ and $\mathbf{G}_5$ is identical. First, since there are at most t corruptions, the values $b(1), \dots, b(n)$ are t -wise independent (for fixed $b(0) = s$). Therefore, the distributions of secret share outputs $\{h^{b(i)}\}_{i \in \mathcal{C}}$ is identical in both games. Second, recall that the values $\tilde{\gamma}_i \in \mathbb{Z}_p$ in game $\mathbf{G}_5$ and $\gamma_i \in \mathbb{Z}_p$ in game $\mathbf{G}_6$ are sampled uniformly random and independent from other values in their respective game. Therefore, the distributions of encryption outputs $\{(c_i[0], c_i[1], c_i[2])\}_{i \in \mathcal{H}_0}$ is identical in both games. To briefly recap: First, we can output the encryptions $c_i[2]$ as $c_i[2] := g^{\gamma_i}$ for a uniformly random sampled $\gamma_i \leftarrow \mathbb{Z}_p$, since the encryptions look uniformly random and independent as argued in game $\mathbf{G}_5$. Second, upon a corruption of party i , we can output the secret shares $h^{b(i)}$ as $h^{b(i)} := g^{\beta_i}$ for a uniformly random sampled $\beta_i \leftarrow \mathbb{Z}_p$, since the output of $c_i[2]$ does not give any information on $b(i)$ (and we do not use $b(i)$ anywhere else) and the first t values $\{b(i) \mid i \in \mathcal{C}\}$ therefore can be sampled uniformly at random.

At this point, we highlight the following interesting observation. Once the first t values β_i are sampled, the degree- t polynomial $b(T) \in \mathbb{Z}_p[T]$ with $b(0) = s$ is uniquely determined. As such, upon a hypothetical $(t+1)$ -th corruption query j , we would need to compute the corresponding β_j as a deterministic linear combination of $b(0)$ and the already-sampled t values $\{\beta_i\}_{i \in \mathcal{C}}$ (i.e., via Lagrange interpolation). But this would force us to reference the value $b(0)$, which we clearly want to avoid. Otherwise, we cannot change the secret $S = h^{b(0)}$ to a uniformly random secret $S' = h^{s'}$ in the final game; more concretely: we cannot provide an efficient reduction to DDH that interpolates between these two games. Having said that, we conclude with

$$\Pr[\mathbf{G}_6 \Rightarrow 1] = \Pr[\mathbf{G}_5 \Rightarrow 1].$$

From this sequence of games, we observe the following: (i) The game references $a(0) = s$ only when it computes the commitment element $\text{com}[0] = g^{a(0)}$, and (ii) the game references $b(0) = s$ only to compute the secret $S = h^{b(0)}$ (which might be given to the adversary in the distinguishing game). In the subsequent final game, we will use $b(0) := s'$ for a uniformly random sampled $s' \leftarrow \mathbb{Z}_p$ to compute the secret S' . Clearly, since s and s' are not referenced anywhere else, this change should be indistinguishable under DDH.

GAME $\mathbf{G}_7$: In this game, we change the full secret of the transcript. Namely, we first sample $s' \leftarrow \mathbb{Z}_p$ uniformly random and independent of s and then compute the final secret as $S' := h^{b(0)} := h^{s'}$. We bound the distinguishing advantage between games $\mathbf{G}_7$ and $\mathbf{G}_6$ via a straightforward reduction R to DDH. To this end, R gets a tuple $\mathcal{D} := (g, h, \hat{g}, \hat{h})$ as input and then simulates game $\mathbf{G}_6$ for A with the following changes. First, it samples t uniformly random values $a_1, \dots, a_t \leftarrow \mathbb{Z}_p$, defines $g_j := g^{a_j} \in \mathbb{G}$ for all $j \in [t]$, and then computes the polynomial commitment as

$$\forall i \in [n] : \text{com}[i] := \hat{g} \cdot \prod_{j \in [t]} g_j^{i^j}, \quad \text{com}[0] := \hat{g}.$$

Second, it generate the final secret as $S' := \hat{h}$. In the end, R outputs whatever the game outputs. Clearly, R perfectly simulates game $\mathbf{G}_6$ for the adversary if $\mathcal{D}$ is a DH tuple. On the other hand, if $\mathcal{D}$ is *not* a DH tuple, then R perfectly simulates game $\mathbf{G}_7$. We therefore get

$$|\Pr[\mathbf{G}_7 \Rightarrow 1] - \Pr[\mathbf{G}_6 \Rightarrow 1]| \leq \varepsilon_{\text{ddh}}.$$

Finally, since $S' \in \mathbb{G}$ is a uniformly random element and independent of the sampled bit $b \in \{0, 1\}$, it is clear that the final game is unconditionally hard. Therefore, we conclude with $\Pr[\mathbf{G}_7 \Rightarrow 1] = \frac{1}{2}$. $\square$

On Correctness and Public Verifiability. In the following, we give a brief high-level discussion of these properties. Formal proofs are deferred to Appendix D.1 and Appendix D.2, respectively.

For correctness, observe that an honestly generated transcript defines a degree- t polynomial f in the exponent by publishing group elements encoding $f(i)$. Each secret share is encrypted in a manner algebraically consistent with this commitment, and the accompanying NIZK proofs certify the correctness of these encryptions. Given its secret key, each party decrypts its ciphertext to obtain exactly the committed share in the exponent. Perfect completeness of the NIZK proof systems and of the low-degree test ensures that all verification checks accept for honestly generated transcripts. Finally, by correctness of Lagrange interpolation, any set of $t+1$ honest shares uniquely reconstructs the original secret.

Public verifiability follows from the soundness of the verification mechanisms. First, statistical soundness of the low-degree test guarantees that any accepted commitment defines—except with probability at most $1/p$ —a unique degree- t polynomial in the exponent, and therefore a uniquely determined secret. Second, computational soundness of the NIZK proof systems ensures that any ciphertext or decryption share that verifies satisfies the prescribed algebraic relations with respect to this polynomial. Encryption proofs enforce consistency with the committed evaluations, while decryption proofs enforce correctness of the corresponding decryptions. Consequently, except with negligible probability, no adversary can produce verifying shares that are inconsistent with the polynomial defined by the commitment. Thus, all verifying shares—whether generated honestly or maliciously—must agree on the same underlying polynomial. Reconstruction from any set of $t+1$ verifying shares therefore yields a unique secret, establishing public verifiability. Finally, the soundness of the NIZK systems follows from the Fiat–Shamir transform in the random-oracle model. Against any (possibly unbounded) adversary making at most q_h random oracle queries, the soundness error is bounded by q_h/p [PS00].

5 Aggregatable PVSS in Pairing Groups

In this section, we present our non-interactive, aggregatable PVSS scheme over asymmetric pairing groups *Illusia* and then give an adaptive security proof for it under the DDH assumption. We begin by giving the syntax and security definition of non-interactive, aggregatable PVSS schemes.

5.1 Formal Definitions

Let PGrGen be a pairing group generation algorithm that on input 1^λ outputs asymmetric pairing group parameters $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, p, g, \hat{g}, e)$, where $\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T$ are cyclic groups of prime order p , $g \in \mathbb{G}$, $\hat{g} \in \hat{\mathbb{G}}$ are generators, and $e : \mathbb{G} \times \hat{\mathbb{G}} \rightarrow \mathbb{G}_T$ is a bilinear pairing of source groups $\mathbb{G} \neq \hat{\mathbb{G}}$ (i.e., e is an efficiently computable non-degenerate bilinear map).

Computational Assumptions. In addition to the DDH assumption (in both base groups), we will also rely on the co-CDH assumption for bilinear groups, formally defined as follows.

Definition 6 (co-CDH Assumption). The co-CDH problem is (τ, ε) -hard for pairing group generation algorithm PGrGen if for any $\lambda \in \mathbb{N}$ and any algorithm A that runs in time at most τ ,

$$\Pr[A(\mathcal{G}, g^a, g^b, \hat{g}^a) = g^{ab} \mid \mathcal{G} := (\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, p, g, \hat{g}, e) \leftarrow \text{PGrGen}(1^\lambda), a, b \leftarrow \mathbb{Z}_p] \leq \varepsilon.$$

Syntax. We define a non-interactive, aggregatable (t, n) -threshold PVSS scheme. Our definition mostly follows that of a (plain) non-interactive (t, n) -threshold PVSS scheme as given in Section 4, with the following three differences. First, the secret sharing algorithm **Share** additionally takes the dealing party i 's secret key sk_i as input. Second, we add a deterministic aggregation algorithm **Agg** that allows to combine any finite number of transcripts into a single transcript. And third, we add a deterministic contributor identifier algorithm **ConID** which returns the list of dealing parties from an input transcript (potentially aggregated).

The first and third modifications are closely related: The first binds a transcript to its creator, while the third enables its efficient identification. In practice, this is typically done by having parties digitally sign their transcripts using signing keys from an additional digital signature scheme. For simplicity, we model this by letting the secret sharing algorithm take a party's secret key as input, which allows the transcript to be signed after its generation. The resulting signature then allows to identify the transcript's creator, referred to as the *contributor*. We capture this functionality by the auxiliary contributor identifier algorithm. Moreover, since an aggregated transcript may contain multiple contributions, the algorithm outputs the full list $\mathcal{I}$ of contributors.

Definition 7 (Non-Interactive, Aggregatable PVSS Scheme). Let $t < n$ be natural numbers. A non-interactive, aggregatable (t, n) -threshold PVSS scheme consists of algorithms with the following syntax:

- $\text{Setup}(1^\lambda) \rightarrow \text{par}$: The *setup algorithm* takes the security parameter 1^λ as input, and outputs the public parameters par . We assume par implicitly defines a set of *input secrets* $\mathcal{M}_{in}$ and a set of *output secrets* $\mathcal{M}_{out}$. Further, all other algorithms implicitly take par as input.
- $\text{KGen}() \rightarrow (\text{pk}, \text{sk})$: The *key generation algorithm* outputs a secret key sk and a public key pk .
- $\text{Share}(\mathcal{L}, s, i, \text{sk}_i) \rightarrow (\text{trx}, S)$: The *secret sharing algorithm* takes a set of n public keys $\mathcal{L} = \{\text{pk}_1, \dots, \text{pk}_n\}$, a secret $s \in \mathcal{M}_{in}$, an identity index $i \in [n]$, and a secret key sk_i as input, and outputs a transcript trx and a secret $S \in \mathcal{M}_{out}$.
- $\text{Agg}(\{\text{trx}_i\}_{i \in [k]}) \rightarrow \text{trx}$: The deterministic *aggregation algorithm* takes a finite number of $k \geq 1$ transcripts $\{\text{trx}_i\}_{i \in [k]}$ as input, and outputs a transcript trx .
- $\text{ConID}(\mathcal{L}, \text{trx}) \rightarrow \mathcal{I}$: The deterministic *contributor identifier algorithm* takes a set of n public keys $\mathcal{L} = \{\text{pk}_1, \dots, \text{pk}_n\}$ and a transcript trx as input, and outputs a list $\mathcal{I} \in [n]^k$ of $k \in \mathbb{N}$ contributor indices (allowing multiplicities).
- $\text{Ver}(\mathcal{L}, \text{trx}) \rightarrow 0/1$: The deterministic *verification algorithm* takes a set of n public keys $\mathcal{L} = \{\text{pk}_1, \dots, \text{pk}_n\}$ and a transcript trx as input, and outputs 0 (invalid) or 1 (valid).
- $\text{DecShare}(\text{trx}, i, \text{sk}_i) \rightarrow (S_i, \mu_i)$: The *share decryption algorithm* takes a transcript trx , an identity index $i \in [n]$, and a secret key sk_i as input, and outputs a secret share $S_i \in \mathcal{M}_{out}$ and a proof μ_i .

- $\text{DecShareVer}(\text{trx}, i, \text{pk}_i, S_i, \mu_i) \rightarrow 0/1$: The deterministic *share decryption verification algorithm* takes a transcript trx , an identity index $i \in [n]$, a public key pk_i , a secret share $S_i \in \mathcal{M}_{\text{out}}$, and a proof μ_i as input, and outputs 0 (invalid) or 1 (valid).
- $\text{Rec}(\mathcal{S}, \{S_i\}_{i \in \mathcal{S}}) \rightarrow S$: The deterministic *reconstruction algorithm* takes a set of $t + 1$ indices $\mathcal{S} \subseteq [n]$ and corresponding secret shares $\{S_i\}_{i \in \mathcal{S}}$ as input, and outputs a secret $S \in \mathcal{M}_{\text{out}}$.

For correctness, we require that for any $\lambda \in \mathbb{N}$, any $t, n \in \text{poly}(\lambda)$ with $t < n$, any $\text{par} \in \text{Setup}(1^\lambda)$, any $(\text{pk}_1, \text{sk}_1), \dots, (\text{pk}_n, \text{sk}_n) \in \text{KGen}()$, any $k \in \mathbb{N}$, any $\mathcal{I} := (i_1, \dots, i_k) \in [n]^k$, any $s_1, \dots, s_k \in \mathcal{M}_{\text{in}}$, any $(\text{trx}_j, S_j) \in \text{Share}(\mathcal{L}, s_j, i_j, \text{sk}_{i_j})$ for $j \in [k]$, any $\mathcal{S} \subseteq [n]$ with $|\mathcal{S}| = t + 1$, and any $i \in \mathcal{S}$, we have

$$\begin{aligned} \text{Ver}(\mathcal{L}, \text{trx}) &= 1, \quad \text{DecShareVer}(\text{trx}, i, \text{pk}_i, S'_i, \mu_i) = 1, \\ \text{Rec}(\mathcal{S}, \{S'_i\}_{i \in \mathcal{S}}) &= \prod_{j \in \mathcal{S}} S_j, \quad \text{ConID}(\mathcal{L}, \text{trx}) = \mathcal{I}, \end{aligned}$$

where $\text{trx} := \text{Agg}(\{\text{trx}_j\}_{j \in [k]})$ and $(S'_i, \mu_i) := \text{DecShare}(\text{trx}, i, \text{sk}_i)$. Note that, by setting $k = 1$, this definition recovers the correctness notion of plain non-interactive PVSS (cf. Section 4.1).

Public Verifiability. We define public verifiability of non-interactive, aggregatable PVSS schemes via the same security game PV (see Fig. 1) that is used for plain non-interactive PVSS schemes. That is, a non-interactive, aggregatable (t, n) -threshold PVSS scheme APVSS is (τ, ε) -publicly verifiable if for every $\lambda \in \mathbb{N}$ and every adversary $\mathbf{A}$ that runs in time at most $\tau = \tau(\lambda)$, we have $\Pr[\text{PV}_{\text{APVSS}}^{\mathbf{A}}(1^\lambda) \Rightarrow 1] \leq \varepsilon = \varepsilon(\lambda)$, where the security game PV is defined in Fig. 1. Syntactically, the adversary may output an aggregated transcript. Moreover, the ability to output a non-aggregated transcript subsumes the setting of plain non-interactive PVSS.

Security. We define the security of non-interactive, aggregatable PVSS schemes via the notion of unpredictability introduced in [BL23]. The respective security game Unpred is described in Fig. 4. Intuitively, this notion prohibits an adversary corrupting at most t parties from learning the secret of an aggregated transcript with at least one honest contribution, even if the adversary is allowed to contribute itself to the aggregate.

In more detail, let $\mathbf{A}$ be the adversary in this game. The setup phase is identical to that of the indistinguishability game for plain PVSS schemes defined in Fig. 2. In particular, adversarial public keys may be generated arbitrarily, and no proofs of possession of secret keys are required. As before, $\mathbf{A}$ has access to a corruption oracle CORR throughout the execution of the game. Additionally, $\mathbf{A}$ is given access to a transcript oracle TRX which, on input an honest party identifier $i \in \mathcal{H}$, returns a transcript generated on behalf of party i using its secret key sk_i and a uniformly randomly chosen input secret. The adversary may query TRX any finite number of times and at arbitrary points during the execution of the game. At the end of the game, $\mathbf{A}$ outputs a tuple $(\text{trx}^*, S^*, \mathcal{S}^*, \{(S_j^*, \mu_j^*)\}_{j \in \mathcal{S}^* \cap \mathcal{C}})$, where trx^* is a transcript, S^* is a candidate secret, $\mathcal{S}^* \subseteq [n]$ is a set of $t + 1$ parties, and for each corrupted party $j \in \mathcal{S}^* \cap \mathcal{C}$, (S_j^*, μ_j^*) denotes a purported decrypted share together with a proof for party j . Let $\mathcal{L} := \{\text{pk}_1, \dots, \text{pk}_n\}$ be the set of public keys. Let S be the secret obtained by running the reconstruction algorithm on input the adversary-provided decrypted shares $\{S_j^*\}_{j \in \mathcal{S}^* \cap \mathcal{C}}$ and the honestly decrypted shares $\{S_i := \text{DecShare}(\text{trx}^*, i, \text{sk}_i)\}_{i \in \mathcal{S}^* \cap \mathcal{H}}$, provided that all proofs μ_j^* verify. The adversary $\mathbf{A}$ wins the game if and only if all of the following conditions hold: (i) trx^* verifies with respect to the public key set $\mathcal{L}$, (ii) trx^* has at least one honest contributor, and (iii) $S^* = S$.

Definition 8 (Security of Non-interactive, Aggregatable PVSS Scheme). A non-interactive, aggregatable (t, n) -threshold PVSS scheme APVSS is (q_t, τ, ε) -secure if for every $\lambda \in \mathbb{N}$ and every adversary $\mathbf{A}$ that makes at most $q_t = q_t(\lambda)$ transcript oracle queries and runs in time at most $\tau = \tau(\lambda)$, we have $\Pr[\text{Unpred}_{\text{APVSS}}^{\mathbf{A}}(1^\lambda) \Rightarrow 1] \leq \varepsilon = \varepsilon(\lambda)$, where the security game Unpred is defined in Fig. 4.

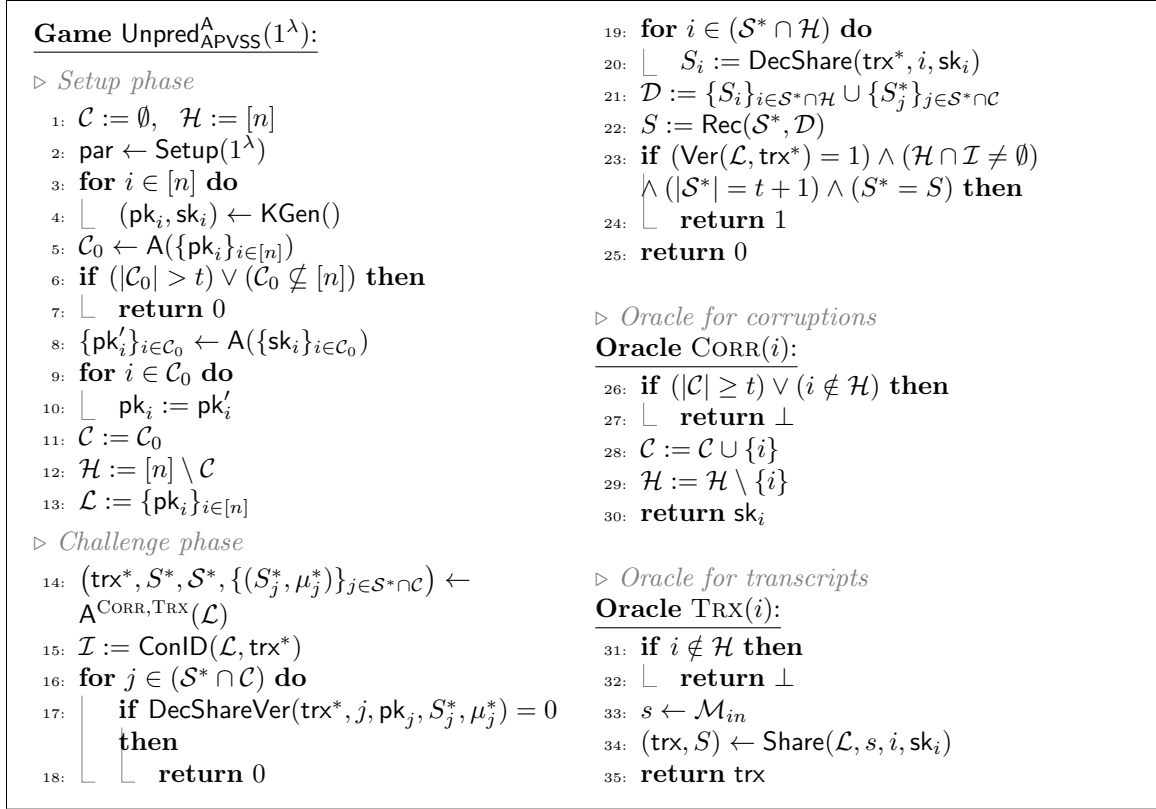


Figure 4: The unpredictability security game Unpred for non-interactive, aggregatable PVSS scheme APVSS against adversary A.

5.2 Homomorphic NIZK Arguments

In our construction, we will make use of a *homomorphic* NIZK argument system for linear relations, namely Groth-Sahai proofs [GS08]. We refer to Appendix B.1 for a formal definition of these argument systems.

The following lemma from [DHLW10] (Lemma 5.2) shows the homomorphism property of Groth-Sahai proofs for linear relations. For a proof, we refer the reader directly to the paper.

Lemma 2. *Groth-Sahai [GS08] is a homomorphic NIZK argument system for $\mathcal{R}_{\text{lin}}$, assuming DDH holds in both base groups output by the underlying pairing group generation algorithm.*

5.3 Our Scheme

In our construction, we will make explicit use of a digital signature scheme DS. The signature is used to bind a transcript to a particular party (i.e., the dealing party), and thus allows to implement the contributor identifier algorithm ConID. In the following, we will write $\langle m \rangle_i$ to denote the message-signature-identifier tuple (m, σ, i) , where σ is a signature on (m, i) signed by party i with its signing key from DS. For a formal definition of a digital signature scheme, we refer to Appendix B.

We give a verbal description of our non-interactive, aggregatable PVSS scheme Illusia next and present it formally as pseudocode in Fig. 5.²¹

²¹For better readability, we omit the public-secret keys of DS from the inputs of the algorithms of Illusia in Fig. 5.

Setup(1^λ): <pre> 1: $\mathcal{G} := (\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, p, g, \hat{g}, e) \leftarrow \text{PGrGen}(1^\lambda)$ 2: $h, u, v \leftarrow \mathbb{G}$ 3: $\text{p\bar{a}r} \leftarrow \text{DS.Setup}(1^\lambda)$ 4: return $\text{par} := (\mathcal{G}, h, u, v, \text{p\bar{a}r})$ </pre> KGen($\cdot$): <pre> 5: $x, y \leftarrow \mathbb{Z}_p$ 6: $\text{sk} := (x, y) \in \mathbb{Z}_p^2$ 7: $\text{pk} := g^x u^y \in \mathbb{G}$ 8: $(\text{pk}, \text{sk}) \leftarrow \text{DS.KGen}(\cdot)$ 9: return $(\text{pk}, \text{sk}), (\text{pk}, \text{sk})$ </pre> Share($\mathcal{L}, s, l, \text{sk}_l$): <pre> 10: $\{\text{pk}_1, \dots, \text{pk}_n\} := \mathcal{L}$ 11: $\{s_1, \dots, s_n\} \leftarrow \text{Shamir}(n, t, s)$ 12: $\text{com} := (g^s, g^{s_1}, \dots, g^{s_n})$ 13: for $i \in [n]$ do 14: $r_i \leftarrow \mathbb{Z}_p, c_i := (g^{r_i}, u^{r_i}, \text{pk}_i^{r_i} h^{s_i})$ 15: $\text{stm} := (\mathbf{B}_i, \mathbf{T}_i) \triangleright \text{Eq. (1)}$ 16: $\pi_i := \text{GS.NIZK.P}_1^H(\text{stm}, (s_i, r_i))$ 17: $r \leftarrow \mathbb{Z}_p, c_0 := (g^r, v^r h^s), \zeta := g^s$ 18: $\text{stm} := (\mathbf{B}_{0,l}, \mathbf{T}_{0,l}) \triangleright \text{Eq. (3)}$ 19: $\pi_0 := \text{NIZK.P}_0^H(\text{stm}, (s, r)) \triangleright \text{Fig. 10}$ 20: $\theta := (c_0, \pi_0)$ 21: $\sigma_l := \text{DS.Sign}(\text{sk}_l, (\zeta, \theta, l))$ 22: $\langle \zeta, \theta \rangle_l := ((\zeta, \theta), \sigma_l, l)$ 23: $\text{trx} := (\text{com}, \{c_i, \pi_i\}_{i \in [n]}, \langle \zeta, \theta \rangle_l)$ 24: return trx </pre> Agg($\{\text{trx}_l\}_{l \in [k]}$): <pre> 25: for $l \in [k]$ do 26: $(\text{com}_l, \{c_{i,l}, \pi_{i,l}\}_{i \in [n]}, \langle \zeta_l, \theta_l \rangle_{j_l}) := \text{trx}_l$ 27: $\text{com} := \prod_{l \in [k]} \text{com}_l$ 28: for $i \in [n]$ do 29: $c_i := \prod_{l \in [k]} c_{i,l}$ 30: $\pi_i := \prod_{l \in [k]} \pi_{i,l}$ 31: $\text{trx} := (\text{com}, \{c_i, \pi_i\}_{i \in [n]}, \{\langle \zeta_l, \theta_l \rangle_{j_l}\}_{l \in [k]})$ 32: return trx </pre> ConID($\mathcal{L}, \text{trx}$): <pre> 33: $\{\text{pk}_1, \dots, \text{pk}_n\} := \mathcal{L}$ 34: $(\text{com}, \{c_i, \pi_i\}_{i \in [n]}, \{\langle \zeta_l, \theta_l \rangle_{j_l}\}_{l \in [k]}) := \text{trx}$ </pre>	<pre> 35: $\mathcal{I} := (j_l)_{l \in [k]}$ 36: return $\mathcal{I}$ </pre> Ver($\mathcal{L}, \text{trx}$): <pre> 37: $\{\text{pk}_1, \dots, \text{pk}_n\} := \mathcal{L}$ 38: $(\text{com}, \{c_i, \pi_i\}_{i \in [n]}, \{\langle \zeta_l, \theta_l \rangle_{j_l}\}_{l \in [k]}) := \text{trx}$ 39: for $i \in [n]$ do 40: $\text{stm} := (\mathbf{B}_i, \mathbf{T}_i) \triangleright \text{Eq. (1)}$ 41: if $\text{GS.NIZK.V}_1^H(\text{stm}, \pi_i) = 0$ then 42: return 0 43: for $l \in [k]$ do 44: $(c_{0,l}, \pi_{0,l}) := \theta_l$ 45: $\text{stm} := (\mathbf{B}_{0,l}, \mathbf{T}_{0,l}) \triangleright \text{Eq. (3)}$ 46: if $(\text{NIZK.V}_0^H(\text{stm}, \pi_{0,l}) = 0)$ $\vee (\text{DS.Ver}(\text{pk}_l, \langle \zeta_l, \theta_l \rangle_l) = 0)$ then $\triangleright$ <i>Fig. 10</i> 47: return 0 48: $\zeta := \prod_{l \in [k]} \zeta_l$ 49: if $\text{com}[0] \neq \zeta$ then 50: return 0 51: return $\text{degTest}(\text{com}, n, t) \triangleright \text{Fig. 7}$ </pre> DecShare($\text{trx}, i, \text{sk}_i$): <pre> 52: $(\text{com}, \{c_i, \pi_i\}_{i \in [n]}, \mathcal{K})_{l \in [k]} := \text{trx}$ 53: $(c_1, c_2, c_3) := c_i$ 54: $(x_i, y_i) := \text{sk}_i$ 55: $S_i := c_3 / (c_1^{x_i} c_2^{y_i})$ 56: $\text{stm} := (\mathbf{B}'_i, \mathbf{T}'_i) \triangleright \text{Eq. (2)}$ 57: $\mu_i := \text{NIZK.P}_2^H(\text{stm}, \text{sk}_i) \triangleright \text{Fig. 9}$ 58: return (S_i, μ_i) </pre> DecShareVer($\text{trx}, i, \text{pk}_i, S_i, \mu_i$): <pre> 59: $(\text{com}, \{c_i, \pi_i\}_{i \in [n]}, \mathcal{K})_{l \in [k]} := \text{trx}$ 60: $(c_1, c_2, c_3) := c_i$ 61: $\text{stm} := (\mathbf{B}'_i, \mathbf{T}'_i) \triangleright \text{Eq. (2)}$ 62: if $\text{NIZK.V}_2^H(\text{stm}, \mu_i) = 0$ then $\triangleright \text{Fig. 9}$ 63: return 0 64: return 1 </pre> Rec($\mathcal{S}, \{S_i\}_{i \in \mathcal{S}}$): <pre> 65: for $i \in \mathcal{S}$ do 66: $\ell_{i,\mathcal{S}} := \prod_{k \in \mathcal{S} \setminus \{i\}} \frac{k}{k-i}$ 67: return $S := \prod_{i \in \mathcal{S}} S_i^{\ell_{i,\mathcal{S}}}$ </pre>
--	--

Figure 5: Our non-interactive, aggregatable PVSS scheme Illusia.

Setup. The public parameters par of the scheme consist of asymmetric pairing group parameters $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, p, g, \hat{g}, e)$ generated by PGrGen , three uniformly random group elements $h, u, v \in \mathbb{G}$, and

the system parameters of the digital signature scheme.²² It also implicitly defines the set of input secrets as $\mathbb{Z}_p$ and the set of output secrets as $\mathbb{G}$.

Key Generation. The same as in *Illusi*, but it also outputs a pair $(\bar{\mathbf{pk}}, \bar{\mathbf{sk}})$ generated by the key generation algorithm of the digital signature scheme DS.

Secret Sharing. The same as in *Illusi*, but the dealer $l \in [n]$ also computes a *signature of knowledge* $\langle \zeta, \theta \rangle_l := \langle g^s, \theta \rangle_l$ as described below. Concretely, The transcript $\text{trx} := (\text{com}, \{c_i, \pi_i\}_{i \in [n]}, \langle \zeta, \theta \rangle_l)$ consists of the following components:

- *Commitments and Encryptions:* The same as in *Illusi*.
- *NIZK Proofs:* The same as in *Illusi*, but the proofs $\{\pi_i\}_{i \in [n]}$ are homomorphic Groth-Sahai proofs now, labeled as $\text{GS.NIZK}_1^H = (\text{GS.NIZK.P}_1^H, \text{GS.NIZK.V}_1^H)$.
- *Signature of Knowledge:* This is a signed ElGamal encryption of the secret S under the public parameter $v \in \mathbb{G}$ along with a NIZK proof of correct encryption. Concretely, the dealer samples some uniform randomness $r \leftarrow \mathbb{Z}_p$ and encrypts the secret S towards v as $c_0 := (g^r, v^r h^s)$. Further, a proof π_0 of correct encryption is computed (derived from an efficient Σ -protocol). It proves a statement in $\mathcal{R}_{\text{lin}}$, i.e., the satisfiability of the following system of linear equations $(\mathbf{B}_{0,l}, \mathbf{T}_{0,l})$, using (s, r) as the witness:

$$\mathbf{B}_{0,l} = \begin{pmatrix} g & \mathbb{1} \\ \mathbb{1} & g \\ h & v \end{pmatrix}, \quad \mathbf{T}_{0,l} = \begin{pmatrix} \zeta \\ c_{0,1} \\ c_{0,2} \end{pmatrix}. \quad (3)$$

The signature of knowledge is then $\langle \zeta, \theta \rangle_l$, where $\theta := (c_0, \pi_0)$ and the dealer l signs the tuple (g^s, θ, l) using the secret key $\bar{\mathbf{sk}}_l$ from the digital signature scheme DS and $\zeta := g^s$. Further, for the NIZK proof π_0 , we use the notation $\pi_0 = \text{NIZK.P}_0^H(\text{stm}, (s, r))$, where $\text{stm} := (\mathbf{B}_{0,l}, \mathbf{T}_{0,l})$ as above (see Eq. (3)). We refer to Fig. 10 for a detailed description of the NIZK proof.

Aggregation. Given $k \geq 1$ transcripts $\{\text{trx}_l := (\text{com}_l, \{c_{i,l}, \pi_{i,l}\}_{i \in [n]}, \langle \zeta_l, \theta_l \rangle_{j_l})\}_{l \in [k]}$, an aggregator does as follows. It aggregates the commitments, encryptions, and NIZK proofs using their underlying homomorphic structure:

$$\text{com} := \prod_{l \in [k]} \text{com}_l, \quad \forall i \in [n]: c_i := \prod_{l \in [k]} c_{i,l}, \quad \pi_i := \prod_{l \in [k]} \pi_{i,l}.$$

The signatures of knowledge are plainly collected as $\mathcal{K} := \{\langle \zeta_l, \theta_l \rangle_{j_l}\}_{l \in [k]}$. The aggregated transcript is then output as $\text{trx} := (\text{com}, \{c_i, \pi_i\}_{i \in [n]}, \mathcal{K})$.

Contributors. Given a (potentially aggregated) transcript $\text{trx} := (\text{com}, \{c_i, \pi_i\}_{i \in [n]}, \{\langle \zeta_l, \theta_l \rangle_{j_l}\}_{l \in [k]})$, the contributor list is derived as $\mathcal{I} := (j_l)_{l \in [k]} \in [n]^k$ from the signatures of knowledge.

Verification. A given transcript $\text{trx} := (\text{com}, \{c_i, \pi_i\}_{i \in [n]}, \langle \zeta, \theta \rangle_j)$ is verified with respect to the list of n public keys $\mathcal{L} := \{\mathbf{pk}_1, \dots, \mathbf{pk}_n\}$ as follows. The same as in *Illusi*, but the verifier additionally checks the signature of knowledge. To this end, it first verifies the signature in $\langle \zeta, \theta \rangle_j$ using the public key $\bar{\mathbf{pk}}_j$ of the dealer j from DS. Then, it checks that the proof $\theta =: (c_0, \pi_0)$ verifies by invoking $\text{NIZK.V}_0^H(\text{stm}, \pi_0)$, where $\text{stm} := (\mathbf{B}_{0,j}, \mathbf{T}_{0,j})$ (see Eq. (3) above). Finally, it checks that $\zeta = \text{com}[0]$ holds. The transcript trx verifies if and only if all these checks are successful.

Verification of an aggregated transcript is performed in the same way, except that all $k \geq 1$ signatures of knowledge are verified individually, and instead of checking $\zeta_l = \text{com}[0]$, the verifier checks that $\prod_{l \in [k]} \zeta_l = \text{com}[0]$.²³

²²We also assume that it implicitly includes the random oracles from the NIZK proof systems.

²³Note that ζ_l encodes only a single party l 's secret g^{s_l} , whereas com already encodes their sum $g^{\sum_{l \in [k]} s_l}$, since all individual commitments have been aggregated.

Decryption, Decryption Share Verification, and Reconstruction. The same as in *Illusi*. We note that for these algorithms, the signatures of knowledge are essentially irrelevant.

Remark 3. We stress that homomorphic Groth–Sahai proofs are used exclusively for the proofs of correct encryption, which is precisely what enables aggregation of ciphertexts and proofs; all other proofs in *Illusia* are standard Fiat–Shamir–based NIZKs and do not require homomorphism.

5.4 Security Analysis

We give an adaptive security proof for *Illusia* under the DDH and co-CDH assumptions in asymmetric pairing groups. For a detailed proof, we refer to Appendix D.5. Further, we defer the proofs of correctness and public verifiability of our scheme to Appendices D.3 and D.4, respectively, as both properties follow from standard arguments.

Theorem 3. *If the DDH problem is $(\tau_{\text{ddh}}, \varepsilon_{\text{ddh}})$ -hard and the co-CDH problem is $(\tau_{\text{co-cdh}}, \varepsilon_{\text{co-cdh}})$ -hard for the asymmetric pairing group generation algorithm PGrGen ,²⁴ if GS.NIZK_1 is a $(\tau_{\text{snd}}, \varepsilon_{\text{snd}})$ -sound Groth–Sahai NIZK argument system, and if the digital signature scheme DS is $(q_s, \tau_{\text{uf}}, \varepsilon_{\text{uf}})$ -secure, then *Illusia* is a $(q_t, \tau_{\text{unpred}}, \varepsilon_{\text{unpred}})$ -secure non-interactive, aggregatable (t, n) -threshold PVSS scheme in the random oracle model against any adversary that makes at most q_h hash queries, where $\tau_{\text{unpred}} \approx \tau_{\text{ddh}} + \tau_{\text{co-cdh}} + \tau_{\text{uf}} + \tau_{\text{snd}}$ and $\varepsilon_{\text{unpred}} \leq \text{negl}(\lambda)$.*

Proof. We structure the proof as a sequence of hybrid games and bound the change in success probability between consecutive games.

GAME $\mathbf{G}_0$: This is the real unpredictability game $\text{Unpred}_{\text{APVSS}}^{\Lambda}(1^\lambda)$ as defined in Fig. 4. By definition, we have $\Pr[\mathbf{G}_0 \Rightarrow 1] = \varepsilon_{\text{unpred}}$.

GAME $\mathbf{G}_1$: In this game, we sample a uniformly random party index $i^* \leftarrow [n]$ and abort the game when $i^* \notin \mathcal{I} \cap \mathcal{H}$, where $\mathcal{I}$ denotes the contributor set of the adversary’s output transcript tr^* . Clearly, this choice is information-theoretically hidden from the adversary. Further, assuming the adversary’s forgery transcript tr^* is valid, it must hold $\mathcal{I} \cap \mathcal{H} \neq \emptyset$. Thus, the probability that our guess i^* is included in $\mathcal{I} \cap \mathcal{H}$ can be upper bounded by $1/n$. We thus get

$$\Pr[\mathbf{G}_1 \Rightarrow 1] \geq \frac{1}{n} \cdot \Pr[\mathbf{G}_0 \Rightarrow 1].$$

GAME $\mathbf{G}_2$: In this game, we rule out the possibility that the adversary produces a valid forged signatures for party i^* with respect to the underlying digital signature scheme DS . Concretely, we parse the signatures of knowledge $\{\langle \zeta_\ell, \theta_\ell \rangle_{j_\ell}\}_{\ell \in [k]}$ appearing in tr^* appearing in the forgery transcript tr^* . By game $\mathbf{G}_1$, we know that one of these tuples verifies under i^* ’s public key pk_{i^*} ; let $\langle \zeta, \theta \rangle_{i^*}$ be this tuple. However, if i^* never produced such a signature on (ζ, θ, i^*) during any transcript oracle query $\text{TRX}(i^*)$, then we abort the game. It is straightforward to give a direct reduction to the unforgeability of DS , and therefore omit it here. We thus get

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq \varepsilon_{\text{uf}}.$$

GAME $\mathbf{G}_3$: In this game, we sample a uniformly random transcript oracle query $\ell^* \leftarrow [q_t]$. By game $\mathbf{G}_2$, it is guaranteed that the signature of knowledge $\langle \zeta, \theta \rangle_{i^*}$ appearing in tr^* was output from an oracle query $\text{TRX}(i^*)$. At the end of this game, we check whether the ℓ^* -th valid query to TRX has output this signature of knowledge $\langle \zeta, \theta \rangle_{i^*}$. If not, then we abort the game. Clearly, this choice is information-theoretically hidden from the adversary. Further, assuming $\mathbf{G}_2$ returns 1, the probability that our guess ℓ^* is correct can be upper bounded by $1/q_t$. We thus get

$$\Pr[\mathbf{G}_3 \Rightarrow 1] \geq \frac{1}{q_t} \cdot \Pr[\mathbf{G}_2 \Rightarrow 1].$$

²⁴We require DDH to hold in both base groups. This is needed for zero-knowledge simulation of Groth–Sahai proofs.

GAME $\mathbf{G}_4$: In this game, we change how we compute the ElGamal encryption c_0 in the ℓ^* -th oracle query to TRX , corresponding to the signature of knowledge $\text{sok}^* := \langle \zeta, \theta \rangle_{i^*}$ of party i^* . By game $\mathbf{G}_3$, we know that sok^* appearing in trx^* was generated by TRX . Recall that $\theta = (c_0, \pi_0)$ where $c_0 = (g^r, v^r h^s)$ for $r, s \leftarrow \mathbb{Z}_p$. Instead of computing the tuple (c_0, π_0) honestly, in this game, we switch to a fake Elgamal encryption c_0 along with a simulated NIZK proof π_0 . That is, we sample $r, r' \leftarrow \mathbb{Z}_p$ and output $c_0 := (g^r, v^{r'} h^s)$. At the same time, we use the special honest-verifier zero-knowledge simulation to generate π_0 , which fails with probability at most q_h/p in the random oracle model. All other components of the transcript are generated exactly as before honestly. It is standard to give a reduction to DDH, and therefore omit it here. We thus get

$$|\Pr[\mathbf{G}_4 \Rightarrow 1] - \Pr[\mathbf{G}_3 \Rightarrow 1]| \leq \varepsilon_{\text{ddh}} + \frac{q_h}{p}.$$

GAME $\mathbf{G}_5$: In this game, we change the computation of the ElGamal encryption c_0 again. In the previous game, we sampled $r, r' \leftarrow \mathbb{Z}_p$ and output $c_0 := (g^r, v^{r'} h^s)$. In this game, we instead sample $r, \gamma \leftarrow \mathbb{Z}_p$ and output $c_0 := (g^r, g^\gamma)$. Conditioned on $v \neq 1$, this modification is indistinguishable, as r' is sampled independently and uniformly at random. Since v is sampled uniformly at random at the beginning of the game, the event $v = 1$ happens with probability at most $1/p$. We thus get

$$|\Pr[\mathbf{G}_5 \Rightarrow 1] - \Pr[\mathbf{G}_4 \Rightarrow 1]| \leq \frac{1}{p}.$$

GAME $\mathbf{G}_6$: In this game, we add an abort condition. Instead of testing the adversary's forgery secret S^* equals the value reconstructed from actual decryptions, we instead want to check if it equals a product of individual secrets extracted from the signatures of knowledge. Concretely, at setup, the game generates $v := g^\alpha \in \mathbb{G}$ with known discrete logarithm by sampling $\alpha \leftarrow \mathbb{Z}_p$ uniformly at random. At the end of the game, it receives $(\text{trx}^*, S^*, \mathcal{S}^*, \{(S_j^*, \mu_j^*)\}_{j \in \mathcal{S}^* \cap \mathcal{C}})$ from the adversary and reconstructs the secret $S_{\text{rec}} := \text{Rec}(\mathcal{S}^*, \mathcal{D})$ from the decrypted values $\mathcal{D}$ of honest and corrupt parties. At the same time, it parses signature of knowledge tuples in trx^* as $\mathcal{K}^* := \{\langle \zeta_\ell, \theta_\ell \rangle_{j_\ell} \}_{\ell \in [k]}$, where $\theta_\ell = (c_{0,\ell}, \pi_{0,\ell})$ and $c_{0,\ell} = (c_{0,\ell}[1], c_{0,\ell}[2])$. Now let $m \in [k]$ denote the number of copies of $\text{sok}^* := \langle \zeta, \theta \rangle_{i^*}$ corresponding to the special honest party i^* . Let $\mathcal{I}_{\text{rest}} := \mathcal{K}^* \setminus \{\text{sok}^*\}$ denote the set of the remaining signatures of knowledge. For each $\ell \in \mathcal{I}_{\text{rest}}$, we then define the extracted contribution $X_\ell := c_{0,\ell}[2]/(c_{0,\ell}[1])^\alpha$, and the signature-of-knowledge-implied secret $S_{\text{sok}} := (h^s)^m \prod_{\ell \in \mathcal{I}_{\text{rest}}} X_\ell$, where s denotes the secret exponent associated with the special honest party i^* . Then, assuming game $\mathbf{G}_5$ returns 1, the game aborts if $S_{\text{rec}} \neq S_{\text{sok}}$.

We can bound the probability of aborting via a reduction to the simulation-soundness of the proof system NIZK_0 , the soundness of the encryption and decryption proof systems GS.NIZK_1 and NIZK_2 , respectively, and the soundness of the degree test degTest . The former ensures that the adversarial signatures of knowledge appearing in trx^* are well formed, while the latter ensures that the encryptions and decryptions from the adversary are well formed. We omit the reductions here and refer to the detailed proof. We get

$$|\Pr[\mathbf{G}_6 \Rightarrow 1] - \Pr[\mathbf{G}_5 \Rightarrow 1]| \leq \frac{1}{p} + \frac{k(q_h + 1)}{p} + \frac{t \cdot q_h}{p} + n \cdot \varepsilon_{\text{snd}}.$$

GAME $\mathbf{G}_7$: In this game, we now change the winning condition. Instead of computing S_{rec} , we directly compute S_{sok} and check if $S^* = S_{\text{sok}}$. Unless game $\mathbf{G}_6$ aborts, both winning conditions are equivalent. We therefore have

$$\Pr[\mathbf{G}_7 \Rightarrow 1] = \Pr[\mathbf{G}_6 \Rightarrow 1].$$

The subsequent games $\mathbf{G}_8$ to $\mathbf{G}_{13}$ are identical to Games $\mathbf{G}_1$ to $\mathbf{G}_6$ in the security proof of Illusi (cf. Theorem 1). In these games, we further modify the transcript trx_{i^*} from the special honest party i^* . By the previous hybrids, the signature of knowledge sok^* is already generated without referencing

h^s . We now apply the same transformations to the remaining components of trx_{i^*} as in the proof of **Illusi**. This is possible since a transcript of **Illusia** consists of a transcript of **Illusi** together with an additional signature of knowledge. This sequence of games then guarantees the following: In the generation of trx_{i^*} , the value s is referenced only when computing the commitment $\text{com}[0] = g^s$ and when checking the winning condition $S^* = S_{\text{sok}}$ using h^s . We can easily build a reduction to co-CDH as follows.

Building a Reduction. Given a co-CDH instance $(g^a, g^b, \hat{g}^a)$, the reduction embeds g^a as $\text{com}[0]$ in trx_{i^*} and it sets the public parameter $h := g^b$. Using the adversary's output S^* , the reduction computes the extracted contribution X_ℓ for all $\ell \in \mathcal{I}_{\text{rest}}$ as defined in game **G**₆. It then computes its co-CDH solution as

$$X_{\text{co-cdh}} := \left(S^* / \prod_{\ell \in \mathcal{I}_{\text{rest}}} X_\ell \right)^{1/m},$$

where m denotes the number of copies of sok^* in trx^* as defined in game **G**₆. In particular, we have

$$\Pr[\mathbf{G}_{13} \Rightarrow 1] \leq \varepsilon_{\text{co-cdh}}.$$

□

6 Conclusion

In this work, we presented two adaptively secure non-interactive (t, n) -threshold PVSS schemes from well-established assumptions. Our first scheme **Illusi** (cf. Fig. 3) works over any pairing-free cyclic group in which the decisional Diffie-Hellman (DDH) problem is assumed to be hard. Our second scheme **Illusia** (cf. Fig. 5) works over any asymmetric pairing group in which the DDH (in both base groups) and the co-computational Diffie-Hellman (co-CDH) problems are assumed to be hard. Further, **Illusia** supports efficient aggregation of multiple PVSS transcripts into a single transcript (i.e., it provides aggregatability). Notably, both our schemes are highly efficient, non-interactive, and work in the established plain public key model. These properties along with their provable adaptive security make them attractive candidates as building block in higher-level distributed protocols that aim to minimize communication.

Future Research Directions. Our second scheme **Illusia** only works over asymmetric pairing groups. The reason being that our security proof has to assume the hardness of DDH. Designing a non-interactive, aggregatable PVSS scheme from standard assumptions that works over any pairing group is a fascinating open problem. Moreover, it would be interesting to improve the concrete efficiency of our schemes. We are confident that our work will serve as a foundation for future research in this field.

Acknowledgements

Yanbo Chen is supported by an NSERC discovery grant. Julian Loss is supported by the European Union, ERC-2023-StG-101116713, and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972. Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

References

- [ABLS25] Ittai Abraham, Renas Bacho, Julian Loss, and Gilad Stern. Nearly quadratic asynchronous distributed key generation. *Cryptology ePrint Archive*, Paper 2025/006, 2025. [2](#), [30](#)
- [AJM⁺21] Ittai Abraham, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. Reaching consensus for asynchronous distributed key generation. In Avery Miller, Keren Censor-Hillel, and Janne H. Korhonen, editors, *40th ACM PODC*, pages 363–373. ACM, July 2021. [2](#), [30](#)
- [Bag25] Karim Baghery. A unified framework for computational verifiable secret sharing. In *Public-Key Cryptography – PKC 2025: 28th IACR International Conference on Practice and Theory of Public-Key Cryptography, Røros, Norway, May 12–15, 2025, Proceedings, Part IV*, page 110–142, Berlin, Heidelberg, 2025. Springer-Verlag. [30](#)
- [BCP02] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Group Diffie-Hellman key exchange secure against dictionary attacks. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 497–514. Springer, Berlin, Heidelberg, December 2002. [8](#)
- [BK25] Renas Bacho and Alireza Kavousi. Sok: Dlog-based distributed key generation. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 614–632, May 2025. [2](#)
- [BL22] Renas Bacho and Julian Loss. On the adaptive security of the threshold BLS signature scheme. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 193–207. ACM Press, November 2022. [31](#)
- [BL23] Renas Bacho and Julian Loss. Adaptively secure (aggregatable) PVSS and application to distributed randomness beacons. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 1791–1804. ACM Press, November 2023. [2](#), [3](#), [4](#), [5](#), [6](#), [19](#)
- [BLL⁺24] Renas Bacho, Christoph Lenzen, Julian Loss, Simon Ochsenreither, and Dimitrios Papachristoudis. GRandLine: Adaptively secure DKG and randomness beacon with (log-)quadratic communication complexity. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 941–955. ACM Press, October 2024. [2](#), [3](#), [30](#)
- [BNPS03] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-rsa-inversion problems and the security of chaum’s blind signature scheme. *Journal of Cryptology*, 16:185–215, 2003. [2](#)
- [BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In Ueli M. Maurer, editor, *EUROCRYPT’96*, volume 1070 of *LNCS*, pages 399–416. Springer, Berlin, Heidelberg, May 1996. [5](#)
- [BSKN23] Adithya Bhat, Nibesh Shrestha, Aniket Kate, and Kartik Nayak. OptRand: Optimistically responsive reconfigurable distributed randomness. In *NDSS 2023*. The Internet Society, February 2023. [5](#), [10](#), [30](#)
- [CD17] Ignacio Cascudo and Bernardo David. SCRAPE: Scalable randomness attested by public entities. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *ACNS 2017*, volume 10355 of *LNCS*, pages 537–556. Springer, Cham, July 2017. [3](#), [6](#), [9](#), [10](#), [12](#), [30](#), [31](#), [32](#), [35](#), [36](#), [39](#), [41](#), [46](#)

- [CD20] Ignacio Cascudo and Bernardo David. Albatross: Publicly attestable batched randomness based on secret sharing. In *Advances in Cryptology – ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part III*, page 311–341, Berlin, Heidelberg, 2020. Springer-Verlag. [30](#)
- [CD24] Ignacio Cascudo and Bernardo David. Publicly verifiable secret sharing over class groups and applications to dkg and yoso. In *Advances in Cryptology – EUROCRYPT 2024: 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26–30, 2024, Proceedings, Part V*, page 216–248, Berlin, Heidelberg, 2024. Springer-Verlag. [31](#)
- [CDGK22] Ignacio Cascudo, Bernardo David, Lydia Garms, and Anders Konring. Yolo yoso: Fast and simple encryption and secret sharing in the yoso model. In *Advances in Cryptology – ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part I*, page 651–680, Berlin, Heidelberg, 2022. Springer-Verlag. [2](#)
- [CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *28th ACM STOC*, pages 639–648. ACM Press, May 1996. [2](#)
- [CGJ⁺99] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Adaptive security for threshold cryptosystems. In Michael J. Wiener, editor, *CRYPTO’99*, volume 1666 of *LNCS*, pages 98–115. Springer, Berlin, Heidelberg, August 1999. [2](#)
- [CMB23] Kevin Choi, Aathira Manoj, and Joseph Bonneau. SoK: Distributed randomness beacons. In *2023 IEEE Symposium on Security and Privacy*, pages 75–92. IEEE Computer Society Press, May 2023. [2](#)
- [CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Berlin, Heidelberg, April / May 2002. [5](#)
- [Dam92] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *CRYPTO’91*, volume 576 of *LNCS*, pages 445–456. Springer, Berlin, Heidelberg, August 1992. [4](#)
- [Dam10] Ivan Damgård. On σ -protocols. Technical Report CPT 2010 v.2, Department of Computer Science, Aarhus University, 2010. [7](#)
- [DGKR18] Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 66–98. Springer, Cham, April / May 2018. [2](#)
- [DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *51st FOCS*, pages 511–520. IEEE Computer Society Press, October 2010. [5](#), [20](#)
- [DKIR22] Sourav Das, Vinith Krishnan, Irene Miriam Isaac, and Ling Ren. Spurt: Scalable distributed randomness beacon with transparent setup. In *2022 IEEE Symposium on Security and Privacy*, pages 2502–2517. IEEE Computer Society Press, May 2022. [30](#)

- [DPTX25] Sourav Das, Benny Pinkas, Alin Tomescu, and Zhuolun Xiang. Distributed randomness using weighted VUFs. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part VII*, volume 15607 of *LNCS*, pages 314–344. Springer, Cham, May 2025. [30](#), [31](#)
- [DR85] Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. *J. ACM*, 32(1):191–204, January 1985. [31](#)
- [DXT⁺25] Sourav Das, Zhuolun Xiang, Alin Tomescu, Alexander Spiegelman, Benny Pinkas, and Ling Ren. Verifiable secret sharing simplified. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 633–651, 2025. [31](#)
- [FGL⁺25] Hanwen Feng, Yingzi Gao, Yuan Lu, Qiang Tang, and Jing Xu. Practical asynchronous distributed key reconfiguration and its applications. Cryptology ePrint Archive, Paper 2025/149, 2025. [31](#)
- [Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168. Springer, Berlin, Heidelberg, August 2005. [5](#)
- [FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Cham, August 2018. [2](#)
- [FLT24] Hanwen Feng, Zhenliang Lu, and Qiang Tang. Dragon: Decentralization at the cost of representation after arbitrary grouping and its applications to sub-cubic DKG and interactive consistency. In Ran Gelles, Dennis Olivetti, and Petr Kuznetsov, editors, *43rd ACM PODC*, pages 469–479. ACM, June 2024. [30](#)
- [FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO’ 86*, pages 186–194, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg. [7](#)
- [FS01] Pierre-Alain Fouque and Jacques Stern. One round threshold discrete-log key generation without private channels. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 300–316. Springer, Berlin, Heidelberg, February 2001. [31](#)
- [FT25] Hanwen Feng and Qiang Tang. Asymptotically optimal adaptive asynchronous common coin and dkg with silent setup. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology — CRYPTO 2025*, pages 642–675, Cham, 2025. Springer Nature Switzerland. [30](#)
- [GHK⁺21] Craig Gentry, Shai Halevi, Hugo Krawczyk, Bernardo Magri, Jesper Buus Nielsen, Tal Rabin, and Sophia Yakoubov. Yoso: You only speak once: Secure mpc with stateless ephemeral roles. In *Advances in Cryptology — CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part II*, page 64–93, Berlin, Heidelberg, 2021. Springer-Verlag. [2](#)
- [GJM⁺21] Kobi Gurkan, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. Aggregatable distributed key generation. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 147–176. Springer, Cham, October 2021. [2](#), [3](#), [5](#), [30](#)
- [Gro21] Jens Groth. Non-interactive distributed key generation and key resharing. Cryptology ePrint Archive, Paper 2021/339, 2021. [31](#)

- [GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Berlin, Heidelberg, April 2008. 5, 20
- [HMW18] Timo Hanke, Mahnush Movahedi, and Dominic Williams. Dfinity technology overview series, consensus system, 2018. 2
- [HV09] Somayeh Heidarvand and Jorge L. Villar. Public verifiability from pairings in secret sharing schemes. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC 2008*, volume 5381 of *LNCS*, pages 294–308. Springer, Berlin, Heidelberg, August 2009. 10, 30
- [KMM⁺24] Aniket Kate, Easwar Vivek Mangipudi, Pratyay Mukherjee, Hamza Saleem, and Sri Aravinda Krishnan Thyagarajan. Non-interactive vss using class groups and application to dkg. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, page 4286–4300, New York, NY, USA, 2024. Association for Computing Machinery. 31
- [KW03] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM CCS 2003*, pages 155–164. ACM Press, October 2003. 5
- [KWJ24] Alireza Kavousi, Zhipeng Wang, and Philipp Jovanovic. SoK: Public Randomness . In *2024 IEEE 9th European Symposium on Security and Privacy (EuroS&P)*, pages 216–234, Los Alamitos, CA, USA, July 2024. IEEE Computer Society. 2
- [PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. Cryptol.*, 13(3):361–396, January 2000. 17
- [RV05] Alexandre Ruiz and Jorge L. Villar. Publicly verifiable secret sharing from paillier’s cryptosystem. In Christopher Wulf, Stefan Lucks, and Po-Wah Yau, editors, *WEWoRC 2005 – Western European Workshop on Research in Cryptology*, pages 98–108, Bonn, 2005. Gesellschaft für Informatik e.V. 3, 6, 10, 30
- [Sch99] Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic. In Michael J. Wiener, editor, *CRYPTO’99*, volume 1666 of *LNCS*, pages 148–164. Springer, Berlin, Heidelberg, August 1999. 2, 3, 6, 30
- [Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979. 7
- [Sta96] Markus Stadler. Publicly verifiable secret sharing. In Ueli M. Maurer, editor, *EUROCRYPT’96*, volume 1070 of *LNCS*, pages 190–199. Springer, Berlin, Heidelberg, May 1996. 1, 30

Appendix

A Related Work

In his seminal work in 1996, Stadler [Sta96] presented the first construction of a non-interactive publicly verifiable secret sharing (PVSS) scheme. Since then, numerous refinements and constructions have been proposed. Following the blueprint of Stadler, the common approach taken by many of these schemes is as follows. A dealer samples a polynomial $f \in \mathbb{Z}_p[X]$ of degree at most t uniformly at random and commits to it com using Feldman commitments. It also provides encryptions c_i of the polynomial shares $f(i)$ towards party i 's public key pk_i along with non-interactive zero-knowledge proofs π_i showing correct encryption. The transcript is then $\text{trx} := (\text{com}, \{c_i, \pi_i\}_{i \in [n]})$, which can be verified by any third party knowing the list of public keys $\{\text{pk}_1, \dots, \text{pk}_n\}$. Stadler realized the proofs using the Fiat–Shamir heuristic in the random oracle model (ROM), achieving security under the non-standard double discrete logarithm (DDL) assumption. Later in 1999, Schoenmakers [Sch99] gave a more efficient variant, achieving security under the decisional Diffie–Hellman (DDH) assumption.²⁵ In 2005, Ruiz and Villar [RV05] proposed standard model constructions based on Paillier encryption, achieving security under the decisional composite residuosity (DCR) assumption. The authors were also the first to formalize the security notion of indistinguishability of secrets (for non-interactive PVSS schemes). In 2008, Heidaryvand and Villar [HV09] proposed alternative pairing-based constructions in the standard model, achieving security under the decisional bilinear square (DBS) assumption. In terms of concrete efficiency, a major drawback of these approaches is that a verifier must perform $O(nt)$ exponentiations to check correctness of the encrypted shares, yielding $O(n^2)$ computational cost.

In 2017, this issue was elegantly resolved by Cascudo and David [CD17] using linear error correction codes. Specifically, instead of committing to the coefficients of f (as in Feldman commitments), the dealer directly commits to the polynomial shares $f(i)$. Using linear error correction codes and their duals, a verifier can verify that these commitments correspond to a degree- t polynomial, with overwhelming probability. This obviates the need of quadratic many exponentiations and reduces the total verification cost to $O(n)$ exponentiations. The authors provide two constructions: One in the random oracle model using the Fiat–Shamir heuristic, achieving security under the DDH assumption; and one in the standard model using pairings, achieving security under the DBS assumption. Their work demonstrates a major advance and has inspired several follow-up works. Specifically, the works [GJM⁺21, DKIR22, BSKN23] design variants of [CD17] for their distributed randomness beacon protocols. To the best of our knowledge, the scheme by Gurkan et al. [GJM⁺21] is the first to explicitly use digital signatures to support secure aggregation of transcripts. Roughly, the authors augment the pairing-based variant of [CD17] with a *signature of knowledge* of the shared secret. Once sufficiently many transcripts are aggregated, these signatures can be used as a proof of honest contribution.²⁶ This idea was also explicitly used in [BSKN23] to design an efficient distributed randomness beacon, and in [AJM⁺21, BLL⁺24, FLT24, DPTX25, FT25, ABL25] to design an efficient distributed key generation (DKG) protocol. In 2020, Cascudo and David [CD20] presented an improved PVSS scheme building on their prior work [CD17]. More recently, Bagheri [Bag25] proposed a PVSS scheme with similar efficiency improvements. This latter work, however, relies on a new computational assumption called the “polynomial discrete logarithm” (PDL) assumption. Moreover, Bagheri observes that the scheme of Cascudo and David [CD20] appears to contain an issue in its security proof and, thus, also requires the PDL assumption for a corrected proof of security.

²⁵Schoenmakers also showed that the scheme achieves the weaker notion of *unpredictability* under the computational Diffie–Hellman (CDH) assumption. This notion allows the adversary to obtain partial information about the secret, with the only condition that it cannot fully recover it.

²⁶In the general case, parties collect $t + 1$ transcripts to ensure at least one honest contribution. Interestingly, in the works [BLL⁺24, ABL25], much fewer transcripts are sufficient to ensure honest contribution.

Following the generic approach of Fouque and Stern [FS01] in 2001, several works [Gro21, KMM⁺24, CD24, DPTX25] have presented non-interactive PVSS schemes for the purpose of non-interactive DKG protocols.²⁷ The generic template is as follows: First, each party broadcasts a PVSS transcript for a random secret using a broadcast channel. Then, every party locally verifies all received PVSS transcripts and discards those that are invalid. Finally, each party decrypts its own shares from the valid transcripts and locally aggregates them to obtain its DKG secret key share. This construction is modular and works also with non-aggregatable PVSS schemes. Finally, two recent works design *interactive* PVSS schemes [DXT⁺25, FGL⁺25]. In these schemes, the dealer privately sends each party its share together with a commitment to that share (e.g., a Feldman commitment). If the share is correct with respect to the commitment, the party returns a signature on the commitment; for any party from which no signature is received, the dealer instead opens the corresponding share. The resulting PVSS transcript then consists of the collection of signed commitments together with the opened shares for which no signature was obtained. However, transcripts produced in this way are not aggregatable. The work of Feng et al. [FGL⁺25] uses their interactive PVSS scheme as part of a DKG construction, which they prove to be “oracle-aided algebraic” adaptively secure [BL22] under the OMDL assumption. However, they do not establish any security guarantees for the interactive PVSS scheme in isolation, and we suspect that it would also need to rely on the OMDL assumption and satisfy only the weaker notion of unpredictability.

B More Preliminaries

In this section, we provide the preliminaries that were omitted from the main body.

Digital Signatures. A digital signature scheme equips a user with a public-secret key pair $(\mathbf{pk}, \mathbf{sk})$. The secret key $\mathbf{sk}$ allows the user to sign any message of its choice, while the public key $\mathbf{pk}$ allows any third party to verify the signature (on that message). Further, the security of a digital signature scheme (also called *unforgeability*) requires that an adversary cannot create a signature on a new message, even after adaptively obtaining signatures on messages of its choice.

Definition 9 (Digital Signature Scheme). A digital signature scheme consists of algorithms with the following syntax:

- $\text{Setup}(1^\lambda) \rightarrow \text{par}$: The *setup algorithm* takes the security parameter 1^λ as input, and outputs the public parameters par . All other algorithms implicitly take par as input.
- $\text{KGen}() \rightarrow (\mathbf{pk}, \mathbf{sk})$: The *key generation algorithm* outputs a secret key $\mathbf{sk}$ and a public key $\mathbf{pk}$.
- $\text{Sign}(m, \mathbf{sk}) \rightarrow \sigma$: The *signing algorithm* takes a message m and a secret key $\mathbf{sk}$ as input, and outputs a signature σ .
- $\text{Ver}(\mathbf{pk}, m, \sigma) \rightarrow 0/1$: The deterministic *verification algorithm* takes a public key $\mathbf{pk}$, a message m , and a signature σ as input, and outputs 0 (invalid) or 1 (valid).

Definition 10 (Security of Digital Signature Scheme). A digital signature scheme DS is (q_s, τ, ε) -secure if for every $\lambda \in \mathbb{N}$ and every adversary $\mathbf{A}$ that makes at most $q_s = q_s(\lambda)$ signing queries and runs in time at most $\tau = \tau(\lambda)$, we have $\Pr[\text{UFCMA}_{\text{DS}}^{\mathbf{A}}(1^\lambda) \Rightarrow 1] \leq \varepsilon = \varepsilon(\lambda)$, where the security game UFCMA is defined in Fig. 6.

Low-Degree Test from [CD17]. Our PVSS schemes employ the low-degree test from Cascudo and David [CD17] to verify the degree of the committed polynomial, as described in Fig. 7. Specifically, to check that the committed polynomial has degree at most t , a verifier multiplies the commitments in the exponent with a random word from the dual code and checks that the result is $\mathbb{1} \in \mathbb{G}$. This

²⁷We emphasize that these works assume the existence of a (synchronous) broadcast channel. In any concrete implementation, however, a broadcast channel necessarily requires interaction. In fact, without relying on a randomness beacon in the first place, any protocol that implements broadcast must run for at least $t + 1 \in O(n)$ rounds [DR85].

Game $\text{UFCMA}_{\text{DS}}^{\text{A}}(1^\lambda)$:	
$\triangleright$ <i>Setup phase</i> 1: $\mathcal{Q} := \emptyset$ 2: $\text{par} \leftarrow \text{Setup}(1^\lambda)$ 3: $(\text{pk}, \text{sk}) \leftarrow \text{KGen}()$ $\triangleright$ <i>Challenge phase</i> 4: $(m^*, \sigma^*) \leftarrow \text{A}^{\text{SIGN}}(\text{pk})$	5: if $(m^* \notin \mathcal{Q}) \wedge (\text{Ver}(\text{pk}, m^*, \sigma^*) = 1)$ then 6: $\perp$ return 1 7: return 0 $\triangleright$ <i>Oracle for signing</i> Oracle $\text{SIGN}(m)$: 8: $\mathcal{Q} := \mathcal{Q} \cup \{m\}$ 9: $\sigma \leftarrow \text{Sign}(m, \text{sk})$ 10: return σ

Figure 6: The unforgeability game UFCMA for digital signature scheme DS against adversary A.

degTest (com, n, t):	
1: if $ \text{com} \neq n + 1$ then 2: $\perp$ return 0 3: $d := n - t$ 4: $(f_0, \dots, f_d) \leftarrow \mathbb{Z}_p^{d+1}$ 5: $f(T) := \sum_{i=0}^d f_i T^i \in \mathbb{Z}_p[T]$ 6: $\ell_0 := 1 / \prod_{i \in [n]} (0 - i)$	7: for $i \in [n]$ do 8: $\ell_i := 1 / \left(i \cdot \prod_{j \neq i, j \in [n]} (i - j) \right)$ $\triangleright$ <i>Output phase</i> 9: $y := \text{com}_0^{\ell_0 f(0)} \prod_{i \in [n]} \left(\text{com}_i^{\ell_i f(i)} \right)$ 10: if $(y = 1)$ then 11: $\perp$ return 1 12: return 0

Figure 7: The low-degree check from Scrape [CD17].

verification procedure is information-theoretically sound, with error probability $\frac{1}{p}$, the order of the underlying cryptographic group.

B.1 Homomorphic NIZK Arguments

We define homomorphic NIZK arguments for homomorphic (e.g., linear) relations.

Definition 11 (Homomorphic Relations). A relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$ is *homomorphic* if $(\mathcal{X}, \cdot)$, $(\mathcal{W}, +)$ are algebraic groups and for any $(x, w), (x', w') \in \mathcal{R}$, we have $(x \cdot x', w + w') \in \mathcal{R}$.

Remark 4. A linear relation $\mathcal{R}_{\text{lin}}^{\mathbf{B}}$ (for fixed $\mathbf{B}$) is homomorphic in the following sense: If $(\mathbf{T}, \mathbf{x}), (\mathbf{T}', \mathbf{x}') \in \mathcal{R}_{\text{lin}}^{\mathbf{B}}$, then also $(\mathbf{T} \cdot \mathbf{T}', \mathbf{x} + \mathbf{x}') \in \mathcal{R}_{\text{lin}}^{\mathbf{B}}$.

Definition 12 (Homomorphic NIZK Argument for Homomorphic Relations). Let $\mathcal{R}$ be a homomorphic relation and let $\mathcal{L}$ be its associated language. A *homomorphic non-interactive zero-knowledge argument system* for (homomorphic relation) $\mathcal{R}$ consists of the following three algorithms:

- $\text{Setup}(1^\lambda) \rightarrow \text{crs}$: The *setup algorithm* takes the security parameter 1^λ as input, and outputs a common reference string crs . All other algorithms implicitly take crs input.
- $\text{P}(\text{stm}, \text{wit}) \rightarrow \pi$: The *proving algorithm* takes a statement stm and a witness wit as input, and outputs a proof π .
- $\text{Ver}(\text{stm}, \pi) \rightarrow 0/1$: The deterministic *verification algorithm* takes a statement stm and a proof π as input, and outputs 0 (invalid) or 1 (valid).

We require the following properties:

- *Completeness.* For any $\lambda \in \mathbb{N}$, any $\text{crs} \in \text{Setup}(1^\lambda)$, any $(\text{stm}, \text{wit}) \in \mathcal{R}$, and any $\pi \in \text{P}(\text{stm}, \text{wit})$, we have $\text{Ver}(\text{stm}, \pi) = 1$.

- *Homomorphism.* For any $\lambda \in \mathbb{N}$ and any $\text{crs} \in \text{Setup}(1^\lambda)$, P takes randomness from the group $(\mathcal{C}, +)$ and outputs proofs from the group $(\mathcal{P}, \cdot)$, such that for any $(\text{stm}, \text{wit}), (\text{stm}', \text{wit}') \in \mathcal{R}$ and any $\rho, \rho' \in \mathcal{C}$, we have

$$P(\text{stm}, \text{wit}; \rho) \cdot P(\text{stm}', \text{wit}'; \rho') = P(\text{stm} \cdot \text{stm}', \text{wit} + \text{wit}'; \rho + \rho').$$

- (τ, ε) -*Soundness.* For any $\lambda \in \mathbb{N}$ and any adversary A that runs in time at most $\tau = \tau(\lambda)$, we have

$$\Pr[\text{Ver}(\text{stm}, \pi) = 1 \wedge \text{stm} \notin \mathcal{L} \mid \text{crs} \leftarrow \text{Setup}(1^\lambda), (\text{stm}, \pi) \leftarrow A(\text{crs})] \leq \varepsilon.$$

- (τ, ε) -*Zero-Knowledge.* There exists a two-step simulator $(\text{Sim}_1, \text{Sim}_2)$ such that for any $\lambda \in \mathbb{N}$ and any $(\text{stm}, \text{wit}) \in \mathcal{R}$, the tuple (crs, π) output by the following two procedures are (τ, ε) -indistinguishable:

$$\{\text{crs} \leftarrow \text{Setup}(1^\lambda), \pi \leftarrow P(\text{stm}, \text{wit})\} \text{ and } \{(\text{crs}, \text{td}) \leftarrow \text{Sim}_1(1^\lambda), \pi \leftarrow \text{Sim}_2(\text{stm}, \text{td})\}.$$

C Deferred Figures

We present the deferred figures here.

Statement: $(g, u, h, \text{pk}, \text{com}, c_1, c_2, c_3) \in \mathbb{G}^8 \triangleright$ We omit the index i from $\text{pk}, \text{com}, c_1, c_2, c_3$. Witness: $(s, r) \in \mathbb{Z}_p^2$ The prover $\mathcal{P}$ wants to convince the verifier $\mathcal{V}$ that there exist $s, r \in \mathbb{Z}_p$ such that the following relations hold: $\text{com} = g^s, \quad c_1 = g^r, \quad c_2 = u^r, \quad c_3 = h^s \cdot \text{pk}^r.$	
<u>NIZK.P₁^H(stm, (s, r)):</u> 1: $(g, u, h, \text{pk}, \text{com}, c_1, c_2, c_3) := \text{stm}$ 2: $\alpha, \beta \leftarrow \mathbb{Z}_p$ 3: $A_1 := g^\alpha$ 4: $A_2 := g^\beta$ 5: $A_3 := u^\beta$ 6: $A_4 := h^\alpha \cdot \text{pk}^\beta$ 7: $e := H(\text{stm}, A_1, A_2, A_3, A_4) \in \mathbb{Z}_p$ 8: $z_s := \alpha + e \cdot s$ 9: $z_r := \beta + e \cdot r$ 10: return $\pi := (e, z_s, z_r)$	<u>NIZK.V₁^H(stm, π):</u> 1: $(g, u, h, \text{pk}, \text{com}, c_1, c_2, c_3) := \text{stm}$ 2: $(e, z_s, z_r) := \pi$ 3: $A_1 := g^{z_s} \cdot \text{com}^{-e}$ 4: $A_2 := g^{z_r} \cdot c_1^{-e}$ 5: $A_3 := u^{z_r} \cdot c_2^{-e}$ 6: $A_4 := h^{z_s} \cdot \text{pk}^{z_r} \cdot c_3^{-e}$ 7: $e' := H(\text{stm}, A_1, A_2, A_3, A_4) \in \mathbb{Z}_p$ 8: if $e' \neq e$ then 9: return 0 10: return 1

Figure 8: Fiat–Shamir NIZK proof $\text{NIZK}_1 := (\text{NIZK.P}_1^H, \text{NIZK.V}_1^H)$ using random oracle H .

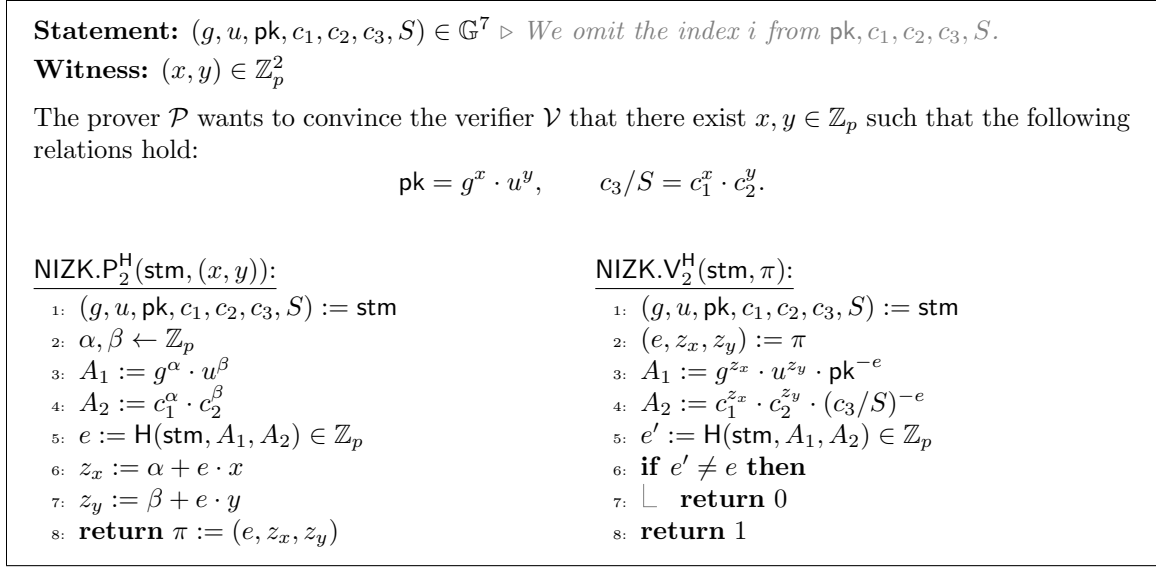


Figure 9: Fiat–Shamir NIZK proof $\text{NIZK}_2 := (\text{NIZK.P}_2^H, \text{NIZK.V}_2^H)$ using random oracle H .

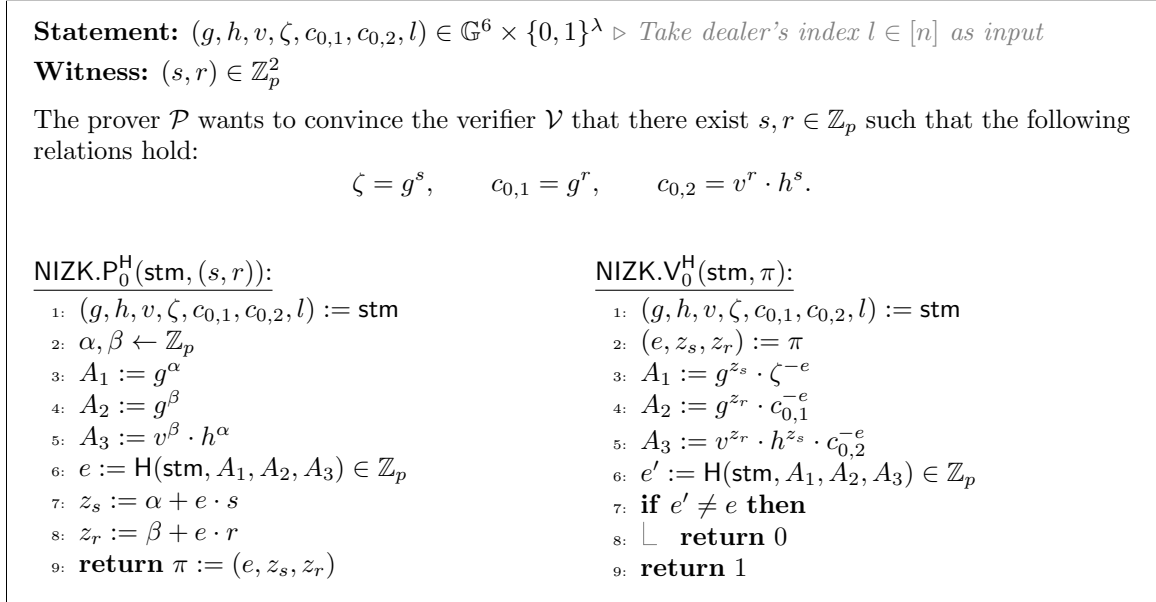


Figure 10: Fiat–Shamir NIZK proof $\text{NIZK}_0 := (\text{NIZK.P}_0^H, \text{NIZK.V}_0^H)$ using random oracle H .

D Deferred Proofs

D.1 Correctness of Illusi

We provide a formal proof of correctness for our non-interactive PVSS scheme `Illusi`.

Theorem 4 (Correctness of `Illusi`). *Let $\lambda \in \mathbb{N}$, let $t, n \in \text{poly}(\lambda)$ with $t < n$, let $\text{par} \leftarrow \text{Setup}(1^\lambda)$, and for each $i \in [n]$, let $(\text{pk}_i, \text{sk}_i) \leftarrow \text{KGen}()$ with $\text{sk}_i := (x_i, y_i) \in \mathbb{Z}_p^2$ and $\text{pk}_i := g^{x_i} u^{y_i}$. For any $s \in \mathcal{M}_{in} = \mathbb{Z}_p$, any $(\text{tr}_x, S) \leftarrow \text{Share}(\mathcal{L}, s)$ where $\mathcal{L} := \{\text{pk}_1, \dots, \text{pk}_n\}$, any $\mathcal{S} \subseteq [n]$ with $|\mathcal{S}| = t + 1$,*

and any $i \in \mathcal{S}$, if $(S_i, \mu_i) := \text{DecShare}(\text{trx}, i, \text{sk}_i)$, then

$$\text{Ver}(\mathcal{L}, \text{trx}) = 1, \quad \text{DecShareVer}(\text{trx}, i, \text{pk}_i, S_i, \mu_i) = 1, \quad \text{Rec}(\mathcal{S}, \{S_j\}_{j \in \mathcal{S}}) = S.$$

Proof. Let $a(T) \in \mathbb{Z}_p[T]$ be the polynomial sampled in **Share**, with $\deg(a) \leq t$ and $a(0) = s$. Thus, the secret shares are $s_i := a(i)$ for all $i \in [n]$, and the dealer outputs the commitment

$$\text{com} := (g^{a(0)}, g^{a(1)}, \dots, g^{a(n)}).$$

Transcript verification. By completeness of **degTest** [CD17], the degree check accepts since **com** is a commitment to evaluations of a polynomial of degree t . For each $i \in [n]$, the ciphertext

$$c_i := (g^{r_i}, u^{r_i}, h^{s_i} \text{pk}_i^{r_i})$$

is generated honestly, and the proof π_i is produced using witness (s_i, r_i) . Since the equalities

$$\text{com}[i] = g^{s_i}, \quad c_{i,1} = g^{r_i}, \quad c_{i,2} = u^{r_i}, \quad c_{i,3} = h^{s_i} \text{pk}_i^{r_i}$$

hold by construction, the statement satisfies the relation of **NIZK**₁. By perfect completeness of the proof system **NIZK**₁, we obtain $\text{NIZK.V}_1^H(\text{stm}_i, \pi_i) = 1$ for all i , and hence $\text{Ver}(\mathcal{L}, \text{trx}) = 1$.

Decryption share verification. Party i computes its decryption share as

$$S_i := \frac{c_{i,3}}{c_{i,1}^{x_i} c_{i,2}^{y_i}} = \frac{h^{s_i} \text{pk}_i^{r_i}}{(g^{r_i})^{x_i} (u^{r_i})^{y_i}} = \frac{h^{s_i} \text{pk}_i^{r_i}}{(g^{x_i} u^{y_i})^{r_i}} = h^{s_i}.$$

Therefore, S_i is exactly the correct Shamir share of the output secret in the exponent. Party i then generates μ_i using witness (x_i, y_i) . Since $\text{pk}_i = g^{x_i} u^{y_i}$ by key generation and

$$c_{i,3}/S_i = \text{pk}_i^{r_i} = c_{i,1}^{x_i} c_{i,2}^{y_i},$$

the statement satisfies the relation of **NIZK**₂. By perfect completeness of the proof system **NIZK**₂, we conclude that $\text{DecShareVer}(\text{trx}, i, \text{pk}_i, S_i, \mu_i) = 1$.

Reconstruction. Let $\mathcal{S} \subseteq [n]$ with $|\mathcal{S}| = t + 1$. By Lagrange interpolation, we have

$$a(0) = \sum_{j \in \mathcal{S}} \ell_{j,\mathcal{S}} a(j), \quad \ell_{j,\mathcal{S}} := \prod_{k \in \mathcal{S} \setminus \{j\}} \frac{k}{k - j}.$$

Using $S_j = h^{s_j} = h^{a(j)}$ for all $j \in \mathcal{S}$, the reconstruction algorithm therefore outputs

$$\prod_{j \in \mathcal{S}} S_j^{\ell_{j,\mathcal{S}}} = \prod_{j \in \mathcal{S}} (h^{a(j)})^{\ell_{j,\mathcal{S}}} = h^{\sum_{j \in \mathcal{S}} \ell_{j,\mathcal{S}} a(j)} = h^{a(0)} = h^s = S.$$

□

D.2 Public Verifiability of **Illusi**

We provide a formal proof of public verifiability for our non-interactive PVSS scheme **Illusi**.

Theorem 5 (Public Verifiability). *Let $\text{PVSS} = \text{Illusi}$ and consider the game $\text{PV}_{\text{PVSS}}^A(1^\lambda)$ from Fig. 1. Then for every PPT adversary $\mathbf{A}$, there exist PPT adversaries $\mathbf{B}_1, \mathbf{B}_2$ such that*

$$\Pr[\text{PV}_{\text{PVSS}}^{\mathbf{A}}(1^\lambda) = 1] \leq \frac{1}{p} + n \cdot \text{Adv}_{\text{NIZK}_1}^{\text{snd}, \mathbf{B}_1}(1^\lambda) + t \cdot \text{Adv}_{\text{NIZK}_2}^{\text{snd}, \mathbf{B}_2}(1^\lambda) \leq \frac{1 + q_h(n + t)}{p}.$$

Proof. Fix an execution of game $\text{PV}_{\text{PVSS}}^{\text{A}}(1^\lambda)$, and let

$$\text{trx} := (\text{com}, \{(c_i, \pi_i)\}_{i \in [n]}) \quad \text{with} \quad c_i = (c_{i,1}, c_{i,2}, c_{i,3}).$$

Let $\mathcal{C}$ and $\mathcal{H} = [n] \setminus \mathcal{C}$ denote the final corrupted and honest sets in the game. Further, let $\mathcal{S}_1, \mathcal{S}_2 \subseteq [n]$ be the sets output by **A**, each of size $t + 1$ (otherwise the game returns 0). We define the following bad events.

- Bad_{deg} is the event that $\text{degTest}(\text{com}, n, t) = 1$ but the soundness conclusion fails, i.e., there is no polynomial $a(T) \in \mathbb{Z}_p[T]$ of degree at most t such that $\text{com}[j] = g^{a(j)}$ for all $j \in \{0\} \cup [n]$. As the soundness error of the degree test is $1/p$ [CD17], we have

$$\Pr[\text{Bad}_{\text{deg}}] \leq \frac{1}{p}.$$

- Bad_1 is the event that at least one of the NIZK_1 proofs in the transcript is an accepting proof for a false statement:

$$\text{Bad}_1 := \left\{ \exists i \in [n] \text{ s.t. } \text{NIZK.V}_1^{\text{H}}(\text{stm}_i, \pi_i) = 1 \wedge \text{stm}_i \notin \mathcal{L}_1 \right\},$$

where $\text{stm}_i := (g, u, h, \text{pk}_i, \text{com}[i], c_{i,1}, c_{i,2}, c_{i,3})$.

- Bad_2 is the event that at least one corrupted decryption share proof checked by the game is an accepting proof for a false statement:

$$\text{Bad}_2 := \left\{ \exists i \in \mathcal{C} \cap (\mathcal{S}_1 \cup \mathcal{S}_2) \text{ s.t. } \text{NIZK.V}_2^{\text{H}}(\text{stm}'_i, \mu_i) = 1 \wedge \text{stm}'_i \notin \mathcal{L}_2 \right\},$$

where $\text{stm}'_i := (g, u, \text{pk}_i, c_{i,1}, c_{i,2}, c_{i,3}, S_i)$.

Lemma 6. *If **A** wins the public verifiability game, then at least one of the events Bad_{deg} , Bad_1 , or Bad_2 occurs. Equivalently, we have*

$$\{\text{PV}_{\text{PVSS}}^{\text{A}}(1^\lambda) = 1\} \subseteq \text{Bad}_{\text{deg}} \cup \text{Bad}_1 \cup \text{Bad}_2.$$

Proof. Assume towards contradiction that **A** wins but none of the bad events occurs. Since **A** wins, the game does not abort at any of its verification steps. In particular, we have

$$\text{Ver}(\mathcal{L}, \text{trx}) = 1, \quad \text{DecShareVer}(\text{trx}, i, \text{pk}_i, S_i, \mu_i) = 1 \quad \forall i \in \mathcal{C} \cap (\mathcal{S}_1 \cup \mathcal{S}_2).$$

From $\text{Ver}(\mathcal{L}, \text{trx}) = 1$, it follows that the low-degree test degTest accepts the commitment com . Because Bad_{deg} does not occur, there exists a polynomial $a(T) \in \mathbb{Z}_p[T]$ of degree at most t such that

$$\text{com}[j] = g^{a(j)} \quad \text{for all } j \in \{0\} \cup [n].$$

Moreover, this polynomial is unique, since $n + 1 > t$ and g generates a group of prime order. Next, again from $\text{Ver}(\mathcal{L}, \text{trx}) = 1$, for every $i \in [n]$ the proof π_i satisfies $\text{NIZK.V}_1^{\text{H}}(\text{stm}_i, \pi_i) = 1$. Since Bad_1 does not occur and NIZK_1 is sound, for each $i \in [n]$, there exist values $s_i, r_i \in \mathbb{Z}_p$ such that

$$\text{com}[i] = g^{s_i}, \quad c_{i,1} = g^{r_i}, \quad c_{i,2} = u^{r_i}, \quad c_{i,3} = h^{s_i} \text{pk}_i^{r_i}.$$

Comparing $\text{com}[i] = g^{s_i}$ with $\text{com}[i] = g^{a(i)}$ implies $s_i = a(i)$, and therefore $h^{s_i} = h^{a(i)}$. Now fix any index $i \in \mathcal{C} \cap (\mathcal{S}_1 \cup \mathcal{S}_2)$. Since decryption shares verify, $\text{DecShareVer}(\text{trx}, i, \text{pk}_i, S_i, \mu_i) = 1$, the proof μ_i satisfies $\text{NIZK.V}_2^{\text{H}}(\text{stm}'_i, \mu_i) = 1$. Because Bad_2 does not occur and NIZK_2 is sound, there exist $x, y \in \mathbb{Z}_p$ such that

$$\text{pk}_i = g^x u^y \quad \text{and} \quad c_{i,3}/S_i = c_{i,1}^x c_{i,2}^y.$$

Substituting $c_{i,1} = g^{r_i}$ and $c_{i,2} = u^{r_i}$ yields

$$c_{i,1}^x c_{i,2}^y = (g^x u^y)^{r_i} = \text{pk}_i^{r_i},$$

and therefore

$$S_i = \frac{c_{i,3}}{\text{pk}_i^{r_i}} = \frac{h^{s_i} \text{pk}_i^{r_i}}{\text{pk}_i^{r_i}} = h^{s_i} = h^{a(i)}.$$

Thus every corrupted share used in reconstruction that passes verification is fixed as $h^{a(i)}$. For honest parties $i \in \mathcal{H}$, the game computes $(S_i, \mu_i) \leftarrow \text{DecShare}(\text{trx}, i, \text{sk}_i)$, and by correctness of the decryption algorithm these shares also equal $h^{a(i)}$. Consequently, $S_i = h^{a(i)}$ for all $i \in \mathcal{S}_1 \cup \mathcal{S}_2$.

Finally, consider reconstruction from any set $\mathcal{S}$ of size $t+1$. Using the reconstruction algorithm, we obtain

$$\text{Rec}(\mathcal{S}, \{S_i\}_{i \in \mathcal{S}}) = \prod_{i \in \mathcal{S}} (h^{a(i)})^{\ell_{i,\mathcal{S}}} = h^{\sum_{i \in \mathcal{S}} \ell_{i,\mathcal{S}} a(i)} = h^{a(0)},$$

where the last equality follows from Lagrange interpolation and the fact that $\deg(a) \leq t$. Applying this argument to both $\mathcal{S}_1$ and $\mathcal{S}_2$ shows that $S = h^{a(0)} = S'$, contradicting the winning condition of the game. This completes the proof of the lemma. $\square$

Applying the union bound, we therefore obtain

$$\Pr[\text{PV}_{\text{VSS}}^{\text{A}}(1^\lambda) = 1] \leq \frac{1}{p} + \Pr[\text{Bad}_1] + \Pr[\text{Bad}_2]. \quad (4)$$

Reduction B_1 from Bad_1 to Soundness of NIZK_1 . We define a PPT adversary B_1 in the soundness game for NIZK_1 as follows:

- B_1 runs A internally and simulates the PV game *honestly*, including the corruption oracle. Concretely, it samples $\text{par} \leftarrow \text{Setup}(1^\lambda)$, generates $(\text{pk}_i, \text{sk}_i) \leftarrow \text{KGen}()$ for all $i \in [n]$, answers $\text{CORR}(i)$ queries with sk_i , and relaying A 's messages to the internal game simulation.
- When A outputs a transcript trx , B_1 checks $\text{Ver}(\mathcal{L}, \text{trx})$. If it fails, B_1 aborts.
- Otherwise, B_1 samples a uniform index $I \leftarrow [n]$ and outputs to its soundness challenger the pair (stm_I, π_I) , where $\text{stm}_I = (g, u, h, \text{pk}_I, \text{com}[I], c_{I,1}, c_{I,2}, c_{I,3})$.

Because $\text{Ver}(\mathcal{L}, \text{trx}) = 1$ implies $\text{NIZK.V}_1^{\text{H}}(\text{stm}_i, \pi_i) = 1$ for all i , we have $\text{NIZK.V}_1^{\text{H}}(\text{stm}_I, \pi_I) = 1$.

If event Bad_1 occurs, then there exists at least one index $i^* \in [n]$ such that $(\text{stm}_{i^*}, \pi_{i^*})$ is a valid proof for a false statement. With probability at least $1/n$, we have $I = i^*$. In that case, B_1 outputs an accepting proof for a false statement and thus wins the NIZK_1 soundness game. Therefore,

$$\text{Adv}_{\text{NIZK}_1}^{\text{snd}, \text{B}_1}(1^\lambda) \geq \frac{1}{n} \Pr[\text{Bad}_1] \iff \Pr[\text{Bad}_1] \leq n \cdot \text{Adv}_{\text{NIZK}_1}^{\text{snd}, \text{B}_1}(1^\lambda).$$

Reduction B_2 from Bad_2 to Soundness of NIZK_2 . We define a PPT adversary B_2 in the soundness game for NIZK_2 as follows:

- B_2 runs A internally and simulates the PV game *honestly*, including the corruption oracle. Concretely, it samples $\text{par} \leftarrow \text{Setup}(1^\lambda)$, generates $(\text{pk}_i, \text{sk}_i) \leftarrow \text{KGen}()$ for all $i \in [n]$, answers $\text{CORR}(i)$ queries with sk_i , and relays A 's messages to the internal game simulation.
- When A outputs a transcript trx , if $\text{Ver}(\mathcal{L}, \text{trx}) = 0$, then B_2 aborts.
- Otherwise, the game proceeds and for each honest $i \in \mathcal{H}$, the simulator computes decryption shares $(S_i, \mu_i) \leftarrow \text{DecShare}(\text{trx}, i, \text{sk}_i)$, and gives these to A .
- When A outputs sets $\mathcal{S}_1, \mathcal{S}_2$ together with corrupted decryption shares $\{(S_i, \mu_i)\}_{i \in \mathcal{C} \cap (\mathcal{S}_1 \cup \mathcal{S}_2)}$, B_2 sets

$$U := \mathcal{C} \cap (\mathcal{S}_1 \cup \mathcal{S}_2).$$

If $U = \emptyset$, then B_2 aborts. (In this case, the adversary cannot win the public verifiability game, since both reconstructions use only honestly generated shares.)

- Otherwise, B_2 samples a uniform index $I \leftarrow U$ and outputs to its soundness challenger the pair (stm'_I, μ_I) , where $\text{stm}'_I := (g, u, \text{pk}_I, c_{I,1}, c_{I,2}, c_{I,3}, S_I)$.

Since B_2 faithfully simulates the PV game, it reaches the output step only if the game does not abort in the verification phase. In particular, this implies that

$$\text{DecShareVer}(\text{trx}, i, \text{pk}_i, S_i, \mu_i) = 1 \quad \text{for all } i \in U.$$

By definition of DecShareVer , we therefore have $\text{NIZK}_2^H(\text{stm}'_I, \mu_I) = 1$. If event Bad_2 occurs, then there exists at least one index $i^* \in U$ such that $(\text{stm}'_{i^*}, \mu_{i^*})$ is a valid proof for a false statement. Since $|U| \leq |\mathcal{C}| \leq t$, we have

$$\Pr[I = i^* \mid \text{Bad}_2] \geq \frac{1}{t}.$$

In that case, B_2 outputs an accepting proof for a false statement and thus wins the NIZK_2 soundness game. Therefore,

$$\text{Adv}_{\text{NIZK}_2}^{\text{snd}, B_2}(1^\lambda) \geq \frac{1}{t} \Pr[\text{Bad}_2] \iff \Pr[\text{Bad}_2] \leq t \cdot \text{Adv}_{\text{NIZK}_2}^{\text{snd}, B_2}(1^\lambda).$$

Combining Bounds. Plugging all bounds into Equation (4), we find

$$\Pr[\text{PV}_{\text{PVSS}}^A(1^\lambda) = 1] \leq \frac{1}{p} + n \cdot \text{Adv}_{\text{NIZK}_1}^{\text{snd}, B_1}(1^\lambda) + t \cdot \text{Adv}_{\text{NIZK}_2}^{\text{snd}, B_2}(1^\lambda).$$

□

Remark 5 (Soundness of the NIZK proofs). The proof systems NIZK_1 and NIZK_2 used in **Illusia** are obtained via the Fiat–Shamir transform applied to standard Σ -protocols for linear relations in the exponent. In particular, the underlying interactive protocols are perfectly complete and satisfy special soundness with challenge space $\mathbb{Z}_p$: from two accepting transcripts with the same first message and different challenges, a valid witness can be efficiently extracted.

As a consequence, the interactive protocols are information-theoretically sound with soundness error $1/p$. After applying the Fiat–Shamir transform in the random oracle model, the resulting non-interactive proof systems remain sound against all PPT adversaries. Concretely, any adversary that outputs an accepting proof for a false statement can be used to either guess the random oracle challenge or to produce two accepting transcripts with different challenges, thereby violating special soundness. Hence, for an adversary making at most q_h queries to the random oracle, the soundness error of both NIZK_1 and NIZK_2 is bounded by q_h/p . For public verifiability, we only rely on this plain soundness property, namely that no false statement can admit an accepting proof except with negligible probability. We do not require knowledge soundness or extractability of the witness.

D.3 Correctness of Illusia

We provide a formal proof of correctness for our non-interactive, aggregatable PVSS scheme **Illusia**. Note that the case $k = 1$ immediately yields correctness for single transcripts.

Theorem 7 (Correctness of **Illusia**). *Let $\lambda \in \mathbb{N}$, let $t, n \in \text{poly}(\lambda)$ with $t < n$, let $\text{par} \leftarrow \text{Setup}(1^\lambda)$, and for each $i \in [n]$ let $((\text{pk}_i, \text{sk}_i), (\text{pk}_i, \bar{\text{sk}}_i)) \leftarrow \text{KGen}()$ with $\text{sk}_i = (x_i, y_i) \in \mathbb{Z}_p^2$ and $\text{pk}_i = g^{x_i} u^{y_i}$, where $(\text{pk}_i, \bar{\text{sk}}_i)$ are keys for DS. Furthermore, let $\mathcal{L} := \{\text{pk}_1, \dots, \text{pk}_n\}$.*

Fix any $k \in \mathbb{N}$ and any contributor list $\mathcal{I} := (i_1, \dots, i_k) \in [n]^k$. For any secrets $s_1, \dots, s_k \in \mathbb{Z}_p$, and for any honestly generated transcripts $(\text{trx}_j, S_j) \leftarrow \text{Share}(\mathcal{L}, s_j, i_j, \text{sk}_{i_j})$ for all $j \in [k]$, let $\text{trx} := \text{Agg}(\{\text{trx}_j\}_{j \in [k]})$. Then, for any set $\mathcal{S} \subseteq [n]$ with $|\mathcal{S}| = t + 1$ and any $i \in \mathcal{S}$, if $(S'_i, \mu_i) := \text{DecShare}(\text{trx}, i, \text{sk}_i)$, we have

$$\text{Ver}(\mathcal{L}, \text{trx}) = 1, \quad \text{DecShareVer}(\text{trx}, i, \text{pk}_i, S'_i, \mu_i) = 1, \quad \text{ConID}(\mathcal{L}, \text{trx}) = \mathcal{I},$$

and moreover

$$\text{Rec}(\mathcal{S}, \{S'_i\}_{i \in \mathcal{S}}) = S, \quad \text{where } S := \prod_{j \in [k]} h^{s_j} \in \mathbb{G}.$$

Proof. For each $j \in [k]$, let $a_j(T) \in \mathbb{Z}_p[T]$ denote the polynomial sampled in $\text{Share}(\mathcal{L}, s_j, i_j, \text{sk}_{i_j})$, with $\deg(a_j) \leq t$ and $a_j(0) = s_j$. Thus, the Shamir shares are $s_{\ell,j} := a_j(\ell)$ for all $\ell \in [n]$, and the commitment of transcript trx_j is

$$\text{com}_j := (g^{a_j(0)}, g^{a_j(1)}, \dots, g^{a_j(n)}).$$

For each $\ell \in [n]$, the encryption component in trx_j is generated honestly as

$$c_{\ell,j} := (g^{\rho_{\ell,j}}, u^{\rho_{\ell,j}}, h^{s_{\ell,j}} \text{pk}_{\ell}^{\rho_{\ell,j}})$$

for uniformly random $\rho_{\ell,j} \leftarrow \mathbb{Z}_p$, accompanied by a Groth-Sahai proof $\pi_{\ell,j}$ for statement $\text{stm}_{\ell,j} = (\mathbf{B}_{\ell}, \mathbf{T}_{\ell,j})$ as in Eq. (1). Furthermore, the dealer i_j forms the signature of knowledge $\langle \zeta_j, \theta_j \rangle_{i_j}$ with $\zeta_j = g^{s_j}$ and $\theta_j = (c_{0,j}, \pi_{0,j})$, where $c_{0,j} := (g^{\rho_j}, v^{\rho_j} h^{s_j})$ for uniformly random $\rho_j \leftarrow \mathbb{Z}_p$, and $\pi_{0,j}$ proving statement $\text{stm}_{0,j} = (\mathbf{B}_0, \mathbf{T}_{0,j})$ as in Eq. (3) using witness (s_j, ρ_j) . Let $S_j := h^{s_j} \in \mathbb{G}$ denote the output secret corresponding to s_j . The aggregation algorithm outputs

$$\text{trx} = (\text{com}, \{c_{\ell}, \pi_{\ell}\}_{\ell \in [n]}, \{\langle \zeta_j, \theta_j \rangle_{i_j}\}_{j \in [k]})$$

where

$$\text{com} := \prod_{j \in [k]} \text{com}_j, \quad c_{\ell} := \prod_{j \in [k]} c_{\ell,j}, \quad \pi_{\ell} := \prod_{j \in [k]} \pi_{\ell,j}.$$

Contributor identification. By definition of ConID , the algorithm parses the aggregated transcript trx , extracts the indices from the k signature of knowledge tuples, and returns

$$\text{ConID}(\mathcal{L}, \text{trx}) = (i_j)_{j \in [k]} = (i_1, \dots, i_k) = \mathcal{I},$$

as required.

Transcript verification. Consider the polynomial $a(T) := \sum_{j \in [k]} a_j(T) \in \mathbb{Z}_p[T]$ of degree $\deg(a) \leq t$. Then, for every $\ell \in \{0, 1, \dots, n\}$, we have

$$\text{com}[\ell] = \prod_{j \in [k]} g^{a_j(\ell)} = g^{\sum_{j \in [k]} a_j(\ell)} = g^{a(\ell)}.$$

Hence, com is a commitment to evaluations of a degree- t polynomial, so by correctness of degTest [CD17] the degree check accepts. Next fix any $\ell \in [n]$, and let

$$\rho_{\ell} := \sum_{j \in [k]} \rho_{\ell,j} \in \mathbb{Z}_p, \quad s_{\ell} := \sum_{j \in [k]} s_{\ell,j} = \sum_{j \in [k]} a_j(\ell) = a(\ell).$$

Then, we have

$$c_{\ell} = \prod_{j \in [k]} (g^{\rho_{\ell,j}}, u^{\rho_{\ell,j}}, h^{s_{\ell,j}} \text{pk}_{\ell}^{\rho_{\ell,j}}) = (g^{\rho_{\ell}}, u^{\rho_{\ell}}, h^{s_{\ell}} \text{pk}_{\ell}^{\rho_{\ell}}),$$

and we also have $\text{com}[\ell] = g^{s_{\ell}}$ from above. Therefore, the aggregated statement $\text{stm}_{\ell} = (\mathbf{B}_{\ell}, \mathbf{T}_{\ell})$ satisfies the relation of GS.NIZK_1 with witness (s_{ℓ}, ρ_{ℓ}) . By homomorphism and perfect completeness of the Groth-Sahai proof system, the aggregated proof $\pi_{\ell} = \prod_{j \in [k]} \pi_{\ell,j}$ verifies, that is, we have $\text{GS.NIZK.V}_1^H(\text{stm}_{\ell}, \pi_{\ell}) = 1$ for all $\ell \in [n]$.

Finally, the verifier checks each signature of knowledge tuple. Fix any $j \in [k]$. By correctness of the digital signature scheme DS, since $\langle \zeta_j, \theta_j \rangle_{i_j}$ is honestly generated under $\mathbf{sk}_{i_j}$, we have

$$\text{DS.Ver}(\bar{\mathbf{pk}}_{i_j}, \langle \zeta_j, \theta_j \rangle_{i_j}) = 1.$$

Moreover, the NIZK proof $\pi_{0,j}$ is generated with witness (s_j, ρ_j) for the statement

$$\mathbf{stm}_{0,j} = (g, h, v, \zeta_j, c_{0,j,1}, c_{0,j,2}),$$

where $\zeta_j = g^{s_j}$ and $c_{0,j} = (g^{\rho_j}, v^{\rho_j} h^{s_j})$. By perfect completeness of NIZK_0 , we thus have $\text{NIZK.V}_0^H(\mathbf{stm}_{0,j}, \pi_{0,j}) = 1$. In addition, the verifier checks the aggregate consistency condition

$$\prod_{j \in [k]} \zeta_j = \text{com}[0].$$

By construction,

$$\prod_{j \in [k]} \zeta_j = \prod_{j \in [k]} g^{s_j} = g^{\sum_{j \in [k]} s_j} = g^{a(0)} = \text{com}[0],$$

where the last equality follows from $\text{com}[0] = \prod_{j \in [k]} g^{a_j(0)}$ and $a_j(0) = s_j$. Thus, all checks in Ver succeed and we obtain $\text{Ver}(\mathcal{L}, \text{trx}) = 1$.

Decryption share verification. Let $\mathcal{S} \subseteq [n]$ with $|\mathcal{S}| = t + 1$ and fix any $i \in \mathcal{S}$. Write $\mathbf{sk}_i = (x_i, y_i)$. From the aggregated ciphertext for index i , we have the representation

$$c_i = (g^{\rho_i}, u^{\rho_i}, h^{s_i} \mathbf{pk}_i^{\rho_i}) \quad \text{with} \quad \rho_i = \sum_{j \in [k]} \rho_{i,j}, \quad s_i = \sum_{j \in [k]} s_{i,j} = a(i).$$

Thus, party i computes

$$S'_i := \frac{c_{i,3}}{c_{i,1}^{x_i} c_{i,2}^{y_i}} = \frac{h^{s_i} \mathbf{pk}_i^{\rho_i}}{(g^{\rho_i})^{x_i} (u^{\rho_i})^{y_i}} = \frac{h^{s_i} \mathbf{pk}_i^{\rho_i}}{(g^{x_i} u^{y_i})^{\rho_i}} = h^{s_i} = h^{a(i)}.$$

Party i then produces a NIZK proof μ_i for the statement $\mathbf{stm}'_i = (\mathbf{B}'_i, \mathbf{T}'_i)$ from Eq. (2) using witness (x_i, y_i) . Since $\mathbf{pk}_i = g^{x_i} u^{y_i}$ and

$$c_{i,3}/S'_i = \mathbf{pk}_i^{\rho_i} = c_{i,1}^{x_i} c_{i,2}^{y_i},$$

the statement $\mathbf{stm}'_i$ is satisfied. By perfect completeness of the proof system NIZK_2 , we therefore get $\text{NIZK.V}_2^H(\mathbf{stm}'_i, \mu_i) = 1$, and hence $\text{DecShareVer}(\text{trx}, i, \mathbf{pk}_i, S'_i, \mu_i) = 1$.

Reconstruction. Define $a(T) := \sum_{j \in [k]} a_j(T)$ as above, and note that for every $i \in [n]$, we have shown $S'_i = h^{a(i)}$. Let $\mathcal{S} \subseteq [n]$ with $|\mathcal{S}| = t + 1$. By Lagrange interpolation, we have

$$a(0) = \sum_{i \in \mathcal{S}} \ell_{i,\mathcal{S}} a(i), \quad \ell_{i,\mathcal{S}} := \prod_{k \in \mathcal{S} \setminus \{i\}} \frac{k}{k - i}.$$

Thus, the reconstruction algorithm outputs

$$\text{Rec}(\mathcal{S}, \{S'_i\}_{i \in \mathcal{S}}) = \prod_{i \in \mathcal{S}} (S'_i)^{\ell_{i,\mathcal{S}}} = \prod_{i \in \mathcal{S}} (h^{a(i)})^{\ell_{i,\mathcal{S}}} = h^{\sum_{i \in \mathcal{S}} \ell_{i,\mathcal{S}} a(i)} = h^{a(0)}.$$

Finally, since $a(0) = \sum_{j \in [k]} a_j(0) = \sum_{j \in [k]} s_j$, we obtain

$$\text{Rec}(\mathcal{S}, \{S'_i\}_{i \in \mathcal{S}}) = h^{\sum_{j \in [k]} s_j} = \prod_{j \in [k]} h^{s_j} = \prod_{j \in [k]} S_j,$$

concluding the proof. $\square$

D.4 Public Verifiability of Illusia

We provide a formal proof of public verifiability for our non-interactive, aggregatable PVSS scheme *Illusia*.

Theorem 8 (Public Verifiability). *Let $\text{APVSS} = \text{Illusia}$ and consider the game $\text{PV}_{\text{APVSS}}^{\text{A}}(1^\lambda)$ from Fig. 1. Then for every PPT adversary A , there exist PPT adversaries $\text{B}_1, \text{B}_2, \text{B}_0$ such that*

$$\begin{aligned} \Pr[\text{PV}_{\text{APVSS}}^{\text{A}}(1^\lambda) = 1] &\leq \frac{1}{p} + n \cdot \text{Adv}_{\text{GS.NIZK}_1}^{\text{snd}, \text{B}_1}(1^\lambda) \\ &\quad + t \cdot \text{Adv}_{\text{NIZK}_2}^{\text{snd}, \text{B}_2}(1^\lambda) + k \cdot \text{Adv}_{\text{NIZK}_0}^{\text{snd}, \text{B}_0}(1^\lambda) = \text{negl}(\lambda), \end{aligned}$$

where k denotes the number of signature of knowledge tuples contained in the transcript output by A , which is polynomially bounded in λ since A is PPT.

Proof. Fix an execution of game $\text{PV}_{\text{APVSS}}^{\text{A}}(1^\lambda)$. Let the transcript output by A be

$$\text{tr} := (\text{com}, \{(c_i, \pi_i)\}_{i \in [n]}, \mathcal{K}), \quad c_i = (c_{i,1}, c_{i,2}, c_{i,3}), \quad \mathcal{K} = \{(\zeta_\ell, \theta_\ell)_{j_\ell}\}_{\ell \in [k]},$$

where $\theta_\ell = (c_{0,\ell}, \pi_{0,\ell})$ and $c_{0,\ell} = (c_{0,\ell,1}, c_{0,\ell,2})$. Let $\mathcal{C}$ and $\mathcal{H} = [n] \setminus \mathcal{C}$ denote the final corrupted and honest sets in the game. Further, let $\mathcal{S}_1, \mathcal{S}_2 \subseteq [n]$ be the sets output by A , each of size $t + 1$ (otherwise the game returns 0). We define the following bad events.

- Bad_{deg} is the event that $\text{degTest}(\text{com}, n, t) = 1$ but the soundness conclusion fails, i.e., there is no polynomial $a(T) \in \mathbb{Z}_p[T]$ of degree at most t such that $\text{com}[j] = g^{a(j)}$ for all $j \in \{0\} \cup [n]$. As the soundness error of the degree test is $1/p$ [CD17], we have

$$\Pr[\text{Bad}_{\text{deg}}] \leq \frac{1}{p}.$$

- Bad_1 is the event that at least one of the encryption proofs π_i in the transcript is an accepting proof for a false statement:

$$\text{Bad}_1 := \left\{ \exists i \in [n] \text{ s.t. } \text{GS.NIZK.V}_1^{\text{H}}(\text{stm}_i, \pi_i) = 1 \wedge \text{stm}_i \notin \mathcal{L}_1 \right\},$$

where $\text{stm}_i := (g, u, h, \text{pk}_i, \text{com}[i], c_{i,1}, c_{i,2}, c_{i,3})$ is the statement from Eq. (1).

- Bad_2 is the event that at least one corrupted decryption share proof checked by the game is an accepting proof for a false statement:

$$\text{Bad}_2 := \left\{ \exists i \in \mathcal{C} \cap (\mathcal{S}_1 \cup \mathcal{S}_2) \text{ s.t. } \text{NIZK.V}_2^{\text{H}}(\text{stm}'_i, \mu_i) = 1 \wedge \text{stm}'_i \notin \mathcal{L}_2 \right\},$$

where $\text{stm}'_i := (g, u, \text{pk}_i, c_{i,1}, c_{i,2}, c_{i,3}, S_i)$ encodes the statement from Eq. (2).

- Bad_0 is the event that at least one of the proofs $\pi_{0,\ell}$ occurring in $\mathcal{K}$ is an accepting proof for a false statement:

$$\text{Bad}_0 := \left\{ \exists \ell \in [k] \text{ s.t. } \text{NIZK.V}_0^{\text{H}}(\text{stm}_{0,\ell}, \pi_{0,\ell}) = 1 \wedge \text{stm}_{0,\ell} \notin \mathcal{L}_0 \right\},$$

where $\text{stm}_{0,\ell}$ encodes the statement from Eq. (3) for the ℓ -th signature-of-knowledge tuple, i.e.,

$$\text{stm}_{0,\ell} := (g, h, v, \zeta_\ell, c_{0,\ell,1}, c_{0,\ell,2}).$$

Lemma 9. *If A wins the public verifiability game, then at least one of the events Bad_{deg} , Bad_1 , Bad_2 , or Bad_0 occurs. Equivalently,*

$$\{\text{PV}_{\text{APVSS}}^{\text{A}}(1^\lambda) = 1\} \subseteq \text{Bad}_{\text{deg}} \cup \text{Bad}_1 \cup \text{Bad}_2 \cup \text{Bad}_0.$$

Proof. Assume towards contradiction that $\mathbf{A}$ wins but none of the bad events occurs. Since $\mathbf{A}$ wins, the game does not abort at any verification step. In particular,

$$\text{Ver}(\mathcal{L}, \text{trx}) = 1 \quad \text{and} \quad \text{DecShareVer}(\text{trx}, i, \text{pk}_i, S_i, \mu_i) = 1 \quad \text{for all } i \in \mathcal{C} \cap (\mathcal{S}_1 \cup \mathcal{S}_2).$$

From $\text{Ver}(\mathcal{L}, \text{trx}) = 1$, the degree test degTest accepts the commitment com . Because Bad_{deg} does not occur, there exists a polynomial $a(T) \in \mathbb{Z}_p[T]$ of degree at most t such that

$$\text{com}[j] = g^{a(j)} \quad \text{for all } j \in \{0\} \cup [n].$$

Moreover, this polynomial is unique, since $n + 1 > t$ and g generates a group of prime order. Next, again from $\text{Ver}(\mathcal{L}, \text{trx}) = 1$, for every $i \in [n]$, the proof π_i satisfies $\text{GS.NIZK.V}_1^H(\text{stm}_i, \pi_i) = 1$. Since Bad_1 does not occur and GS.NIZK_1 is sound, for each $i \in [n]$, there exist values $s_i, r_i \in \mathbb{Z}_p$ such that

$$\text{com}[i] = g^{s_i}, \quad c_{i,1} = g^{r_i}, \quad c_{i,2} = u^{r_i}, \quad c_{i,3} = h^{s_i} \text{pk}_i^{r_i}.$$

Comparing $\text{com}[i] = g^{s_i}$ with $\text{com}[i] = g^{a(i)}$ implies $s_i = a(i)$ and hence $h^{s_i} = h^{a(i)}$.

Fix any index $i \in \mathcal{C} \cap (\mathcal{S}_1 \cup \mathcal{S}_2)$. Since decryption shares verify, $\text{DecShareVer}(\text{trx}, i, \text{pk}_i, S_i, \mu_i) = 1$ and therefore $\text{NIZK.V}_2^H(\text{stm}'_i, \mu_i) = 1$. Because Bad_2 does not occur and NIZK_2 is sound, there exist $x, y \in \mathbb{Z}_p$ such that

$$\text{pk}_i = g^x u^y \quad \text{and} \quad c_{i,3}/S_i = c_{i,1}^x c_{i,2}^y.$$

Substituting $c_{i,1} = g^{r_i}$ and $c_{i,2} = u^{r_i}$ yields

$$c_{i,1}^x c_{i,2}^y = (g^x u^y)^{r_i} = \text{pk}_i^{r_i},$$

and therefore

$$S_i = \frac{c_{i,3}}{\text{pk}_i^{r_i}} = \frac{h^{s_i} \text{pk}_i^{r_i}}{\text{pk}_i^{r_i}} = h^{s_i} = h^{a(i)}.$$

Thus, every corrupted share used in reconstruction that passes verification is fixed as $h^{a(i)}$. For honest parties $i \in \mathcal{H}$, the game computes $(S_i, \mu_i) \leftarrow \text{DecShare}(\text{trx}, i, \text{sk}_i)$, and by correctness of the decryption algorithm these shares also equal $h^{a(i)}$. Consequently, $S_i = h^{a(i)}$ for all $i \in \mathcal{S}_1 \cup \mathcal{S}_2$.

It remains to use the additional checks in Ver for the tuples in $\mathcal{K}$. Since $\text{Ver}(\mathcal{L}, \text{trx}) = 1$, for each $\ell \in [k]$, the verifier accepts the tuple $\langle \zeta_\ell, \theta_\ell \rangle_{j_\ell}$ and in particular verifies $\text{NIZK.V}_0^H(\text{stm}_{0,\ell}, \pi_{0,\ell}) = 1$ for the statement $\text{stm}_{0,\ell} = (g, h, v, \zeta_\ell, c_{0,\ell,1}, c_{0,\ell,2})$. Furthermore, it checks the aggregate consistency condition

$$\prod_{\ell \in [k]} \zeta_\ell = \text{com}[0].$$

Because Bad_0 does not occur and NIZK_0 is sound, for every $\ell \in [k]$, there exist values $s^{(\ell)}, r^{(\ell)} \in \mathbb{Z}_p$ such that

$$\zeta_\ell = g^{s^{(\ell)}} \quad \text{and} \quad c_{0,\ell} = (g^{r^{(\ell)}} v^{r^{(\ell)}} h^{s^{(\ell)}}).$$

Combining this with $\prod_{\ell \in [k]} \zeta_\ell = \text{com}[0] = g^{a(0)}$ yields

$$g^{\sum_{\ell \in [k]} s^{(\ell)}} = g^{a(0)} \implies \sum_{\ell \in [k]} s^{(\ell)} = a(0),$$

where the last implication holds since $\mathbb{G}$ has prime order p and g is a generator. Thus, the tuples in $\mathcal{K}$ are consistent with the aggregate committed secret exponent $a(0)$.

Finally, consider reconstruction from any set $\mathcal{S}$ of size $t + 1$. Using the reconstruction algorithm, we obtain

$$\text{Rec}(\mathcal{S}, \{S_i\}_{i \in \mathcal{S}}) = \prod_{i \in \mathcal{S}} (h^{a(i)})^{\ell_{i,\mathcal{S}}} = h^{\sum_{i \in \mathcal{S}} \ell_{i,\mathcal{S}} a(i)} = h^{a(0)},$$

where the last equality follows from Lagrange interpolation and $\deg(a) \leq t$. Applying this argument to both $\mathcal{S}_1$ and $\mathcal{S}_2$ shows that $S = S' = h^{a(0)}$, contradicting the winning condition of the game. This completes the proof of the lemma. $\square$

Applying the union bound, we obtain

$$\Pr[\text{PV}_{\text{APVSS}}^{\text{A}}(1^\lambda) = 1] \leq \Pr[\text{Bad}_{\text{deg}}] + \Pr[\text{Bad}_1] + \Pr[\text{Bad}_2] + \Pr[\text{Bad}_0]. \quad (5)$$

Reduction B_1 from Bad_1 to Soundness of GS.NIZK_1 . We define a PPT adversary B_1 in the soundness game for GS.NIZK_1 as follows:

- B_1 runs A internally and simulates the PV game *honestly*, including the corruption oracle. Concretely, it samples $\text{par} \leftarrow \text{Setup}(1^\lambda)$, generates $(\text{pk}_i, \text{sk}_i) \leftarrow \text{KGen}()$ for all $i \in [n]$, answers $\text{CORR}(i)$ queries with sk_i , and relays A 's messages to the internal game simulation.
- When A outputs a transcript trx , B_1 checks $\text{Ver}(\mathcal{L}, \text{trx})$. If it fails, B_1 aborts.
- Otherwise, B_1 samples a uniform index $I \leftarrow [n]$ and outputs to its soundness challenger the pair (stm_I, π_I) , where $\text{stm}_I = (g, u, h, \text{pk}_I, \text{com}[I], c_{I,1}, c_{I,2}, c_{I,3})$.

Because $\text{Ver}(\mathcal{L}, \text{trx}) = 1$ implies $\text{GS.NIZK.V}_1^H(\text{stm}_i, \pi_i) = 1$ for all i , we have $\text{GS.NIZK.V}_1^H(\text{stm}_I, \pi_I) = 1$. If event Bad_1 occurs, then there exists at least one index $i^* \in [n]$ such that $(\text{stm}_{i^*}, \pi_{i^*})$ is a valid proof for a false statement. With probability at least $1/n$, we have $I = i^*$. In that case, B_1 outputs an accepting proof for a false statement and thus wins the GS.NIZK_1 soundness game. Therefore,

$$\text{Adv}_{\text{GS.NIZK}_1}^{\text{snd}, B_1}(1^\lambda) \geq \frac{1}{n} \Pr[\text{Bad}_1] \iff \Pr[\text{Bad}_1] \leq n \cdot \text{Adv}_{\text{GS.NIZK}_1}^{\text{snd}, B_1}(1^\lambda).$$

Reduction B_2 from Bad_2 to Soundness of NIZK_2 . We define a PPT adversary B_2 in the soundness game for NIZK_2 as follows:

- B_2 runs A internally and simulates the PV game *honestly*, including the corruption oracle. Concretely, it samples $\text{par} \leftarrow \text{Setup}(1^\lambda)$, generates $(\text{pk}_i, \text{sk}_i) \leftarrow \text{KGen}()$ for all $i \in [n]$, answers $\text{CORR}(i)$ queries with sk_i , and relays A 's messages to the internal game simulation.
- When A outputs a transcript trx , if $\text{Ver}(\mathcal{L}, \text{trx}) = 0$, then B_2 aborts.
- Otherwise, the game proceeds and for each honest $i \in \mathcal{H}$, the simulator computes decryption shares $(S_i, \mu_i) \leftarrow \text{DecShare}(\text{trx}, i, \text{sk}_i)$, and gives these to A .
- When A outputs sets $\mathcal{S}_1, \mathcal{S}_2$ together with corrupted decryption shares $\{(S_i, \mu_i)\}_{i \in \mathcal{C} \cap (\mathcal{S}_1 \cup \mathcal{S}_2)}$, B_2 sets

$$U := \mathcal{C} \cap (\mathcal{S}_1 \cup \mathcal{S}_2).$$

If $U = \emptyset$, then B_2 aborts. (In this case, the adversary cannot win the public verifiability game, since both reconstructions use only honestly generated shares.)

- Otherwise, B_2 samples a uniform index $I \leftarrow U$ and outputs to its soundness challenger the pair (stm'_I, μ_I) , where $\text{stm}'_I := (g, u, \text{pk}_I, c_{I,1}, c_{I,2}, c_{I,3}, S_I)$.

Since B_2 faithfully simulates the PV game, it reaches the output step only if the game does not abort in the verification phase. In particular, this implies that

$$\text{DecShareVer}(\text{trx}, i, \text{pk}_i, S_i, \mu_i) = 1 \quad \text{for all } i \in U.$$

By definition of DecShareVer , we therefore have $\text{NIZK.V}_2^H(\text{stm}'_I, \mu_I) = 1$. If event Bad_2 occurs, then there exists at least one index $i^* \in U$ such that $(\text{stm}'_{i^*}, \mu_{i^*})$ is a valid proof for a false statement. Since $|U| \leq |\mathcal{C}| \leq t$, we have

$$\Pr[I = i^* \mid \text{Bad}_2] \geq \frac{1}{t}.$$

In that case, B_2 outputs an accepting proof for a false statement and thus wins the NIZK_2 soundness game. Therefore,

$$\text{Adv}_{\text{NIZK}_2}^{\text{snd}, B_2}(1^\lambda) \geq \frac{1}{t} \Pr[\text{Bad}_2] \iff \Pr[\text{Bad}_2] \leq t \cdot \text{Adv}_{\text{NIZK}_2}^{\text{snd}, B_2}(1^\lambda).$$

Reduction B_0 from Bad_0 to Soundness of NIZK_0 . We define a PPT adversary B_0 in the soundness game for NIZK_0 as follows:

- B_0 runs A internally and simulates the PV game *honestly*, including the corruption oracle. Concretely, it samples $\text{par} \leftarrow \text{Setup}(1^\lambda)$, generates $(\text{pk}_i, \text{sk}_i) \leftarrow \text{KGen}()$ for all $i \in [n]$, answers $\text{CORR}(i)$ queries with sk_i , and relays A 's messages to the internal game simulation.
- When A outputs a transcript trx , if $\text{Ver}(\mathcal{L}, \text{trx}) = 0$, then B_0 aborts.
- Otherwise, B_0 parses $\mathcal{K} = \{\langle \zeta_\ell, \theta_\ell \rangle_{j_\ell}\}_{\ell \in [k]}$ from trx , samples a uniform index $I \leftarrow [k]$, and outputs to its soundness challenger the pair $(\text{stm}_{0,I}, \pi_{0,I})$, where $\text{stm}_{0,I} := (g, h, v, \zeta_I, c_{0,I,1}, c_{0,I,2})$.

Because $\text{Ver}(\mathcal{L}, \text{trx}) = 1$ implies $\text{NIZK.V}_0^H(\text{stm}_{0,\ell}, \pi_{0,\ell}) = 1$ for all $\ell \in [k]$, we have $\text{NIZK.V}_0^H(\text{stm}_{0,I}, \pi_{0,I}) = 1$. If event Bad_0 occurs, then there exists at least one index $\ell^* \in [k]$ such that $(\text{stm}_{0,\ell^*}, \pi_{0,\ell^*})$ is a valid proof for a false statement. With probability at least $1/k$, we have $I = \ell^*$. In that case, B_0 outputs an accepting proof for a false statement and thus wins the NIZK_0 soundness game. Therefore,

$$\text{Adv}_{\text{NIZK}_0}^{\text{snd}, B_0}(1^\lambda) \geq \frac{1}{k} \Pr[\text{Bad}_0] \iff \Pr[\text{Bad}_0] \leq k \cdot \text{Adv}_{\text{NIZK}_0}^{\text{snd}, B_0}(1^\lambda).$$

Combining Bounds. Plugging all bounds into Equation (5), we find

$$\begin{aligned} \Pr[\text{PV}_{\text{APVSS}}^A(1^\lambda) = 1] &\leq \frac{1}{p} + n \cdot \text{Adv}_{\text{GS.NIZK}_1}^{\text{snd}, B_1}(1^\lambda) \\ &\quad + t \cdot \text{Adv}_{\text{NIZK}_2}^{\text{snd}, B_2}(1^\lambda) + k \cdot \text{Adv}_{\text{NIZK}_0}^{\text{snd}, B_0}(1^\lambda). \end{aligned}$$

□

Remark 6 (Soundness of the NIZK proofs). The public verifiability proof for *Illusia* relies on the soundness of three non-interactive zero-knowledge proof systems: NIZK_0 , NIZK_2 , and GS.NIZK_1 . The Fiat–Shamir–based proof systems NIZK_0 and NIZK_2 are sound against any (potentially unbounded) adversary making at most q_h random oracle queries, with soundness error at most q_h/p . The proof system GS.NIZK_1 is instantiated using Groth–Sahai proofs. In our setting of Type-III asymmetric pairing groups, soundness of Groth–Sahai proofs is computational. Concretely, soundness of GS.NIZK_1 reduces to the SXDH assumption, i.e., the hardness of the DDH problem in both base groups $\mathbb{G}$ and $\hat{\mathbb{G}}$. As a consequence, the soundness error of GS.NIZK_1 is negligible in the security parameter λ . We emphasize that for public verifiability we only rely on the plain soundness property of the above proof systems, namely that no false statement admits an accepting proof except with the stated probability bounds. We do not require knowledge soundness or witness extractability.

D.5 Unpredictability of *Illusia*

Theorem 10. *If the DDH problem is $(\tau_{\text{ddh}}, \varepsilon_{\text{ddh}})$ -hard and the co-CDH problem is $(\tau_{\text{co-cdh}}, \varepsilon_{\text{co-cdh}})$ -hard for the asymmetric pairing group generation algorithm PGrGen ,²⁸ if GS.NIZK_1 is a $(\tau_{\text{snd}}, \varepsilon_{\text{snd}})$ -sound Groth–Sahai NIZK argument system, and if the digital signature scheme DS is $(q_s, \tau_{\text{uf}}, \varepsilon_{\text{uf}})$ -secure, then *Illusia* is a $(q_t, \tau_{\text{unpred}}, \varepsilon_{\text{unpred}})$ -secure non-interactive, aggregatable (t, n) -threshold PVSS scheme in the random oracle model against any adversary that makes at most q_h hash queries, where $\tau_{\text{unpred}} \approx \tau_{\text{ddh}} + \tau_{\text{co-cdh}} + \tau_{\text{uf}} + \tau_{\text{snd}}$ and $\varepsilon_{\text{unpred}} \leq \text{negl}(\lambda)$.*

Proof. We structure the proof as a sequence of hybrid games and bound the change in success probability between consecutive games.

GAME $\mathbf{G}_0$: This is the real unpredictability game $\text{Unpred}_{\text{APVSS}}^A(1^\lambda)$ as defined in Fig. 4. By definition, we have

$$\Pr[\mathbf{G}_0 \Rightarrow 1] = \varepsilon_{\text{unpred}}.$$

²⁸We require DDH to hold in both base groups. This is needed for zero-knowledge simulation of Groth–Sahai proofs.

GAME $\mathbf{G}_1$: In this game, we sample a uniformly random party index $i^* \leftarrow [n]$ and abort the game when $i^* \notin \mathcal{I} \cap \mathcal{H}$, where $\mathcal{I}$ denotes the contributor set of the adversary's output transcript trx^* . Clearly, this choice is information-theoretically hidden from the adversary. Further, assuming the adversary's forgery transcript trx^* is valid, it must hold $\mathcal{I} \cap \mathcal{H} \neq \emptyset$. Thus, the probability that our guess i^* is included in $\mathcal{I} \cap \mathcal{H}$ can be upper bounded by $1/n$. We thus get

$$\Pr[\mathbf{G}_1 \Rightarrow 1] \geq \frac{1}{n} \cdot \Pr[\mathbf{G}_0 \Rightarrow 1].$$

GAME $\mathbf{G}_2$: In this game, we rule out the possibility that the adversary produces a valid forged signatures for party i^* with respect to the underlying digital signature scheme DS. Concretely, we parse the signatures of knowledge $\{\langle \zeta_\ell, \theta_\ell \rangle_{j_\ell}\}_{\ell \in [k]}$ appearing in trx^* appearing in the forgery transcript trx^* . By game $\mathbf{G}_1$, we know that one of these tuples verifies under i^* 's public key pk_{i^*} ; let $\langle \zeta, \theta \rangle_{i^*}$ be this tuple. However, if i^* never produced such a signature on (ζ, θ, i^*) during any transcript oracle query $\text{TRX}(i^*)$, then we abort the game.

To bound the probability of abort, we give a straightforward reduction to the unforgeability of DS. Let UF be the unforgeability challenger. We embed the unforgeability challenge key as the public key pk_{i^*} corresponding to index i^* . We simulate the signatures in the signature of knowledge for all transcript oracle queries using the signing oracle provided by UF. Everything else is executed exactly as in game $\mathbf{G}_2$. It is clear that this perfectly simulates $\mathbf{G}_2$ until the abort occurs. In that case, we know that the message-signature tuple $\langle \zeta, \theta \rangle_{i^*}$ appearing in the adversary's forgery transcript trx^* was never generated using the signing oracle provided by UF (otherwise, the abort would not have occurred). We therefore submit $\langle \zeta, \theta \rangle_{i^*}$ to the unforgeability challenger UF. Clearly, a successful abort in $\mathbf{G}_2$ yields a valid signature forgery. Thus, we obtain

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq \varepsilon_{\text{uf}}.$$

GAME $\mathbf{G}_3$: In this game, we sample a uniformly random transcript oracle query $\ell^* \leftarrow [q_t]$. By game $\mathbf{G}_2$, it is guaranteed that the signature of knowledge $\langle \zeta, \theta \rangle_{i^*}$ appearing in trx^* was output from an oracle query $\text{TRX}(i^*)$. At the end of this game, we check whether the ℓ^* -th valid query to TRX has output this signature of knowledge $\langle \zeta, \theta \rangle_{i^*}$. If not, then we abort the game. Clearly, this choice is information-theoretically hidden from the adversary. Further, assuming $\mathbf{G}_2$ returns 1, the probability that our guess ℓ^* is correct can be upper bounded by $1/q_t$. We thus get

$$\Pr[\mathbf{G}_3 \Rightarrow 1] \geq \frac{1}{q_t} \cdot \Pr[\mathbf{G}_2 \Rightarrow 1].$$

GAME $\mathbf{G}_4$: This game is identical to the previous game, except we now use a simulated NIZK proof $\bar{\pi}_0$, corresponding to the signature of knowledge $\text{sok}^* := \langle \zeta, \theta \rangle_{i^*}$ of party i^* . Recall that $\theta = (c_0, \pi_0)$ where $c_0 = (g^r, v^r h^s)$ for $r, s \leftarrow \mathbb{Z}_p$ and π_0 is a NIZK proof of correct ElGamal encryption c_0 . Concretely, we use the special honest-verifier zero-knowledge (SHVZK) simulation to compute $\bar{\pi}_0$. Clearly, the games $\mathbf{G}_4$ and $\mathbf{G}_3$ are indistinguishable until we generate a commitment $\text{cmt} \in \mathbb{G}^3$ for the simulated proof $\bar{\pi}_0$ that has already been queried to the random oracle H (which underlies the non-interactive proof systems). However, since the commitment is distributed uniformly over a space of size at least p and hidden from $\mathcal{A}$ (until we output the NIZK proof), the probability that the SHVZK simulation fails is at most q_h/p . Thus, we get

$$|\Pr[\mathbf{G}_4 \Rightarrow 1] - \Pr[\mathbf{G}_3 \Rightarrow 1]| \leq \frac{q_h}{p}.$$

GAME $\mathbf{G}_5$: In this game, we change how we compute the corresponding ElGamal encryption c_0 . Instead of computing it honestly, in this game, we switch to a fake ElGamal encryption c_0 . That is, we sample $r, r' \leftarrow \mathbb{Z}_p$ and output $c_0 := (g^r, v^{r'} h^s)$. Everything else is generated exactly as in the

previous game. We bound the distinguishing advantage between games $\mathbf{G}_5$ and $\mathbf{G}_4$ via a reduction R to DDH. To this end, R gets a tuple $\mathcal{D} := (g, v, \hat{g}_1, \hat{v}_1)$ as input²⁹ and then simulates game $\mathbf{G}_4$ for A with the following change. It generates the special encryption c_0 of party i^* as:

$$c_0 := (\hat{g}_i, \hat{v}_i h^s).$$

The rest remains unchanged. In the end, R outputs whatever the game outputs. Clearly, R perfectly simulates game $\mathbf{G}_4$ for the adversary if $\mathcal{D}$ is a DH tuple. On the other hand, if $\mathcal{D}$ is *not* an DH tuple, then R perfectly simulates game $\mathbf{G}_5$. We therefore get

$$|\Pr[\mathbf{G}_5 \Rightarrow 1] - \Pr[\mathbf{G}_4 \Rightarrow 1]| \leq \varepsilon_{\text{ddh}}.$$

GAME $\mathbf{G}_6$: In this game, we change the computation of the ElGamal encryption c_0 again. In the previous game, we sampled $r, r' \leftarrow \mathbb{Z}_p$ and output $c_0 := (g^r, v^{r'} h^s)$. In this game, we instead sample $r, \gamma \leftarrow \mathbb{Z}_p$ and output $c_0 := (g^r, g^\gamma)$. Conditioned on $v \neq 1$, this modification is indistinguishable, as r' is sampled independently and uniformly at random. Since v is sampled uniformly at random at the beginning of the game, the event $v = 1$ happens with probability at most $1/p$. We thus get

$$|\Pr[\mathbf{G}_6 \Rightarrow 1] - \Pr[\mathbf{G}_5 \Rightarrow 1]| \leq \frac{1}{p}.$$

GAME $\mathbf{G}_7$: In this game, we add an abort condition. Instead of testing the adversary's forgery secret S^* equals the value reconstructed from actual decryptions, we instead want to check if it equals a product of individual secrets extracted from the signatures of knowledge. Concretely, at setup, the game generates $v := g^\alpha \in \mathbb{G}$ with known discrete logarithm by sampling $\alpha \leftarrow \mathbb{Z}_p$ uniformly at random. At the end of the game, it receives $(\text{trx}^*, S^*, \mathcal{S}^*, \{(S_j^*, \mu_j^*)\}_{j \in \mathcal{S}^* \cap \mathcal{C}})$ from the adversary and reconstructs the secret $S_{\text{rec}} := \text{Rec}(\mathcal{S}^*, \mathcal{D})$ from the decrypted values $\mathcal{D}$ of honest and corrupt parties. At the same time, it parses signature of knowledge tuples in trx^* as $\mathcal{K}^* := \{\langle \zeta_\ell, \theta_\ell \rangle_{j_\ell}\}_{\ell \in [k]}$, where $\theta_\ell = (c_{0,\ell}, \pi_{0,\ell})$ and $c_{0,\ell} = (c_{0,\ell}[1], c_{0,\ell}[2])$. Now let $m \in [k]$ denote the number of copies of $\text{sok}^* := \langle \zeta, \theta \rangle_{i^*}$ corresponding to the special honest party i^* . Let $\mathcal{I}_{\text{rest}} := \mathcal{K}^* \setminus \{\text{sok}^*\}$ denote the set of the remaining signatures of knowledge. For each $\ell \in \mathcal{I}_{\text{rest}}$, we then define the extracted contribution $X_\ell := c_{0,\ell}[2] / (c_{0,\ell}[1])^\alpha$, and the signature-of-knowledge-implied secret $S_{\text{sok}} := (h^s)^m \prod_{\ell \in \mathcal{I}_{\text{rest}}} X_\ell$, where s denotes the secret exponent associated with the special honest party i^* . Then, assuming game $\mathbf{G}_5$ returns 1, the game aborts if the event $\text{Bad} := S_{\text{rec}} \neq S_{\text{sok}}$ occurs.

We can bound the probability of aborting via a reduction to the simulation-soundness of the proof system NIZK_0 , the soundness of the encryption and decryption proof systems GS.NIZK_1 and NIZK_2 , respectively, and the soundness of the degree test degTest . The former ensures that the adversarial signatures of knowledge appearing in trx^* are well formed, while the latter ensures that the encryptions and decryptions from the adversary are well formed.

To this end, we define further bad events as follows.

- Bad_{deg} is the event that $\text{degTest}(\text{com}, n, t) = 1$ but there is no polynomial $a(T) \in \mathbb{Z}_p[T]$ of degree at most t such that

$$\text{com}[j] = g^{a(j)} \quad \text{for all } j \in \{0\} \cup [n].$$

As the soundness error of degTest is $1/p$ [CD17], we have

$$\Pr[\text{Bad}_{\text{deg}}] \leq \frac{1}{p}.$$

- Bad_1 is the event that at least one encryption proof π_i is accepting for a false statement:

$$\text{Bad}_1 := \left\{ \exists i \in [n] \text{ s.t. } \text{GS.NIZK.V}_1^H(\text{stm}_i, \pi_i) = 1 \wedge \text{stm}_i \notin \mathcal{L}_1 \right\},$$

where $\text{stm}_i := (g, u, h, \text{pk}_i, \text{com}[i], c_{i,1}, c_{i,2}, c_{i,3})$ is the statement from Eq. (1).

²⁹It is meant to be understood that R embeds v into the public parameters as the element “ v ”.

- Bad_2 is the event that at least one corrupted decryption share proof is accepting for a false statement:

$$\text{Bad}_2 := \left\{ \exists i \in \mathcal{C} \cap \mathcal{S}^* \text{ s.t. } \text{NIZK.V}_2^H(\text{stm}'_i, \mu_i^*) = 1 \wedge \text{stm}'_i \notin \mathcal{L}_2 \right\},$$

where $\text{stm}'_i := (g, u, \text{pk}_i, c_{i,1}, c_{i,2}, c_{i,3}, S_i^*)$ encodes the statement from Eq. (2).

- Bad_0 is the event that at least one signature of knowledge proof $\pi_{0,\ell}$ is accepting for a false statement:

$$\text{Bad}_0 := \left\{ \exists \ell \in \mathcal{I}_{\text{rest}} \text{ s.t. } \text{NIZK.V}_0^H(\text{stm}_{0,\ell}, \pi_{0,\ell}) = 1 \wedge \text{stm}_{0,\ell} \notin \mathcal{L}_0 \right\},$$

where $\text{stm}_{0,\ell} := (g, h, v, \zeta_\ell, c_{0,\ell,1}, c_{0,\ell,2})$ encodes the statement from Eq. (3).

Lemma 11 (Structural implication). *If Bad occurs, then at least one of the events Bad_{deg} , Bad_1 , Bad_2 , or Bad_0 occurs. Equivalently, we have*

$$\text{Bad} \subseteq \text{Bad}_{\text{deg}} \cup \text{Bad}_1 \cup \text{Bad}_2 \cup \text{Bad}_0.$$

Proof. Assume towards contradiction that Bad occurs but none of Bad_{deg} , Bad_1 , Bad_2 , Bad_0 occurs.

Since Bad occurs, the game reaches the end without aborting earlier, so in particular

$$\text{Ver}(\mathcal{L}, \text{trx}^*) = 1 \quad \text{and} \quad \text{DecShareVer}(\text{trx}^*, i, \text{pk}_i, S_i^*, \mu_i^*) = 1 \quad \text{for all } i \in \mathcal{C} \cap \mathcal{S}^*,$$

and the reconstruction S_{rec} is defined.

Step 1 (degree-test soundness). From $\text{Ver}(\mathcal{L}, \text{trx}^*) = 1$, the degree test accepts com . Since Bad_{deg} does not occur, there exists a polynomial $a(T) \in \mathbb{Z}_p[T]$ with $\deg(a) \leq t$ such that

$$\text{com}[j] = g^{a(j)} \quad \text{for all } j \in \{0\} \cup [n].$$

Uniqueness holds since $n+1 > t$ and g generates a group of prime order.

Step 2 (encryption proof soundness). Again from $\text{Ver}(\mathcal{L}, \text{trx}^*) = 1$, for all $i \in [n]$, we have that $\text{GS.NIZK.V}_1^H(\text{stm}_i, \pi_i) = 1$. Since Bad_1 does not occur and GS.NIZK_1 is sound, for each $i \in [n]$, there exist $s_i, r_i \in \mathbb{Z}_p$ such that

$$\text{com}[i] = g^{s_i}, \quad c_{i,1} = g^{r_i}, \quad c_{i,2} = u^{r_i}, \quad c_{i,3} = h^{s_i} \text{pk}_i^{r_i}.$$

Comparing $\text{com}[i] = g^{s_i}$ with $\text{com}[i] = g^{a(i)}$ yields $s_i = a(i)$, and hence

$$c_{i,3} = h^{a(i)} \text{pk}_i^{r_i}.$$

Step 3 (decryption proof soundness). Fix any $i \in \mathcal{C} \cap \mathcal{S}^*$. Since $\text{DecShareVer}(\text{trx}^*, i, \text{pk}_i, S_i^*, \mu_i^*) = 1$, we have $\text{NIZK.V}_2^H(\text{stm}'_i, \mu_i^*) = 1$. Since Bad_2 does not occur and NIZK_2 is sound, there exist $x, y \in \mathbb{Z}_p$ such that

$$\text{pk}_i = g^x u^y \quad \text{and} \quad c_{i,3}/S_i^* = c_{i,1}^x c_{i,2}^y.$$

Substituting $c_{i,1} = g^{r_i}$ and $c_{i,2} = u^{r_i}$ yields

$$c_{i,1}^x c_{i,2}^y = (g^x u^y)^{r_i} = \text{pk}_i^{r_i},$$

and hence

$$S_i^* = \frac{c_{i,3}}{\text{pk}_i^{r_i}} = \frac{h^{a(i)} \text{pk}_i^{r_i}}{\text{pk}_i^{r_i}} = h^{a(i)}.$$

For any honest $i \in \mathcal{S}^* \setminus \mathcal{C}$, the game computes S_i via **DecShare**, and by correctness also obtains $S_i = h^{a(i)}$. Therefore, for all $i \in \mathcal{S}^*$, the shares used in reconstruction satisfy $S_i = h^{a(i)}$, and thus

$$S_{\text{rec}} = \text{Rec}(\mathcal{S}^*, \{S_i\}_{i \in \mathcal{S}^*}) = \prod_{i \in \mathcal{S}^*} (h^{a(i)})^{\ell_{i, \mathcal{S}^*}} = h^{a(0)}.$$

That is, we have $S_{\text{rec}} = h^{a(0)}$. We now compute S_{sok} and establish that the two values are identical. *Step 4 (signature-of-knowledge proof soundness + aggregate ζ -check).* Since $\text{Ver}(\mathcal{L}, \text{trx}^*) = 1$, the verifier checks every tuple $\langle \zeta_\ell, \theta_\ell \rangle_{j_\ell}$ and in particular verifies

$$\text{NIZK.V}_0^H(\text{stm}_{0, \ell}, \pi_{0, \ell}) = 1 \quad \text{for all } \ell \in [k].$$

It also checks the aggregate consistency condition

$$\prod_{\ell \in [k]} \zeta_\ell = \text{com}[0].$$

Since **Bad**₀ does not occur and **NIZK**₀ is simulation-sound, for each $\ell \in \mathcal{I}_{\text{rest}}$, there exist $s^{(\ell)}, r^{(\ell)} \in \mathbb{Z}_p$ such that

$$\zeta_\ell = g^{s^{(\ell)}} \quad \text{and} \quad c_{0, \ell} = (g^{r^{(\ell)}}, v^{r^{(\ell)}} h^{s^{(\ell)}}).$$

Combining with $\prod_{\ell \in [k]} \zeta_\ell = \text{com}[0] = g^{a(0)}$ yields

$$\begin{aligned} g^{a(0)} &= \prod_{\ell \in [k]} \zeta_\ell = (\zeta^*)^m \prod_{\ell \in \mathcal{I}_{\text{rest}}} \zeta_\ell = (g^s)^m \cdot g^{\sum_{\ell \in \mathcal{I}_{\text{rest}}} s^{(\ell)}} \\ \implies sm + \sum_{\ell \in \mathcal{I}_{\text{rest}}} s^{(\ell)} &= a(0), \end{aligned}$$

where $\zeta^* = g^s$ denotes the component of i^* 's signature of knowledge and m denotes the number of copies of **sok**^{*} appearing in **trx**^{*}.

Step 5 (correctness of extraction). Recall $v = g^\alpha$ and the extractor defines for each $\ell \in \mathcal{I}_{\text{rest}}$:

$$X_\ell := \frac{c_{0, \ell, 2}}{(c_{0, \ell, 1})^\alpha} = \frac{v^{r^{(\ell)}} h^{s^{(\ell)}}}{(g^{r^{(\ell)}})^\alpha} = \frac{(g^\alpha)^{r^{(\ell)}} h^{s^{(\ell)}}}{g^{\alpha r^{(\ell)}}} = h^{s^{(\ell)}}.$$

Step 6 (combining everything). Therefore the signature-of-knowledge-implied secret is equal

$$S_{\text{sok}} = (h^s)^m \prod_{\ell \in \mathcal{I}_{\text{rest}}} X_\ell = h^{sm} \prod_{\ell \in \mathcal{I}_{\text{rest}}} h^{s^{(\ell)}} = h^{sm + \sum_{\ell \in \mathcal{I}_{\text{rest}}} s^{(\ell)}} = h^{a(0)}.$$

Conclusion. We have shown $S_{\text{rec}} = h^{a(0)}$ and $S_{\text{sok}} = h^{a(0)}$, hence $S_{\text{rec}} = S_{\text{sok}}$, contradicting **Bad**. $\square$

By the union bound and Lemma 11, we get

$$\Pr[\text{Bad}] \leq \Pr[\text{Bad}_{\text{deg}}] + \Pr[\text{Bad}_1] + \Pr[\text{Bad}_2] + \Pr[\text{Bad}_0]. \quad (6)$$

Reduction B₁ from Bad₁ to soundness of GS.NIZK₁. We define a PPT adversary B₁ in the soundness game for GS.NIZK₁ as follows.

- B₁ runs A internally and simulates game **G**₇ for A, including all abort conditions and all oracle answers.
- Eventually, A outputs a purported forgery $(\text{trx}^*, S^*, \mathcal{S}^*, \{(S_j^*, \mu_j^*)\}_{j \in \mathcal{S}^* \cap \mathcal{C}})$. If $\text{Ver}(\mathcal{L}, \text{trx}^*) = 0$, then B₁ aborts.

- Otherwise, B_1 parses $\text{trx}^* = (\text{com}^*, \{(c_i^*, \pi_i^*)\}_{i \in [n]}, \mathcal{K}^*)$ and samples a uniformly random index $I \leftarrow [n]$. It outputs to its GS.NIZK_1 soundness challenger the pair $(\text{stm}_I^*, \pi_I^*)$, where $\text{stm}_I^* := (g, u, h, \text{pk}_I, \text{com}^*[I], c_{I,1}^*, c_{I,2}^*, c_{I,3}^*)$.

Since $\text{Ver}(\mathcal{L}, \text{trx}^*) = 1$ implies that $\text{GS.NIZK.V}_1^H(\text{stm}_i^*, \pi_i^*) = 1$ for all $i \in [n]$, we have that $\text{GS.NIZK.V}_1^H(\text{stm}_I^*, \pi_I^*) = 1$. If event Bad_1 occurs in Game $\mathbf{G}_7$, then there exists an index $i^* \in [n]$ such that $(\text{stm}_{i^*}^*, \pi_{i^*}^*)$ is accepting but $\text{stm}_{i^*}^* \notin \mathcal{L}_1$. With probability at least $1/n$ we have $I = i^*$, in which case B_1 wins the soundness game for GS.NIZK_1 . Therefore,

$$\Pr[\text{Bad}_1] \leq n \cdot \text{Adv}_{\text{GS.NIZK}_1}^{\text{snd}, B_1}(1^\lambda).$$

Reduction B_2 from Bad_2 to soundness of NIZK_2 . We define a PPT adversary B_2 in the soundness game for NIZK_2 as follows.

- B_2 runs A internally and simulates Game $\mathbf{G}_7$ for A , including all abort conditions and all oracle answers.
- Eventually, A outputs a purported forgery $(\text{trx}^*, S^*, \mathcal{S}^*, \{(S_j^*, \mu_j^*)\}_{j \in \mathcal{S}^* \cap \mathcal{C}})$.
- If $\text{Ver}(\mathcal{L}, \text{trx}^*) = 0$, then B_2 aborts.
- Otherwise, B_2 parses $\text{trx}^* = (\text{com}^*, \{(c_i^*, \pi_i^*)\}_{i \in [n]}, \mathcal{K}^*)$ and defines $U := \mathcal{C} \cap \mathcal{S}^*$.
- If $U = \emptyset$, then B_2 aborts.
- Otherwise, B_2 samples a uniformly random index $I \leftarrow U$ and outputs to its NIZK_2 soundness challenger the pair (stm'_I, μ'_I) , where $\text{stm}'_I := (g, u, \text{pk}_I, c_{I,1}^*, c_{I,2}^*, c_{I,3}^*, S_I^*)$.

Since $\text{Ver}(\mathcal{L}, \text{trx}^*) = 1$ implies that

$$\text{DecShareVer}(\text{trx}^*, I, \text{pk}_I, S_I^*, \mu'_I) = 1,$$

we have $\text{NIZK.V}_2^H(\text{stm}'_I, \mu'_I) = 1$. If event Bad_2 occurs in Game $\mathbf{G}_7$, then there exists an index $j^* \in U$ such that

$$(\text{stm}'_{j^*}, \mu'_{j^*})$$

is accepting but $\text{stm}'_{j^*} \notin \mathcal{L}_2$.

Since $|U| \leq |\mathcal{C}| \leq t$, with probability at least $1/t$ we have $I = j^*$. In this case, B_2 outputs an accepting proof for a false statement and thus wins the soundness game for NIZK_2 . Therefore, we get

$$\Pr[\text{Bad}_2] \leq t \cdot \text{Adv}_{\text{NIZK}_2}^{\text{snd}, B_2}(1^\lambda).$$

Reduction B_0 from Bad_0 to simulation-soundness of NIZK_0 . This is where we crucially use simulation-soundness: in earlier hybrids (in particular, in game $\mathbf{G}_4$) the challenger produces a *simulated* NIZK_0 proof for the signature of knowledge sok^* of the special honest party i^* . Therefore, ruling out an accepting NIZK_0 proof for a false statement in the adversary's output transcript requires simulation-soundness of NIZK_0 .

We define a PPT adversary B_0 in the *simulation-soundness* game for NIZK_0 as follows.

- B_0 receives from its simulation-soundness challenger the public parameters for NIZK_0 , as well as oracle access to a simulator Sim_0 that, on input a statement, returns a simulated proof.
- B_0 runs A internally and simulates game $\mathbf{G}_7$ for A , including all abort conditions and all oracle answers, with the following modification: whenever the simulation in game $\mathbf{G}_4$ requires producing the special proof π_0^* inside the special tuple $\text{sok}^* = \langle \zeta^*, \theta^* \rangle_{i^*}$, for the corresponding statement stm_0^* , B_0 queries its simulator oracle and sets

$$\pi_0^* := \text{Sim}_0(\text{stm}_0^*).$$

- Eventually, A outputs a purported forgery $(\text{trx}^*, S^*, \mathcal{S}^*, \{(S_j^*, \mu_j^*)\}_{j \in \mathcal{S}^* \cap \mathcal{C}})$. If $\text{Ver}(\mathcal{L}, \text{trx}^*) = 0$, then B_0 aborts.

- Otherwise, B_0 parses the set of signatures of knowledge in trx^* as

$$\mathcal{K}^* = \{\langle \zeta_\ell, \theta_\ell \rangle_{j_\ell}\}_{\ell \in [k]}, \quad \theta_\ell = (c_{0,\ell}, \pi_{0,\ell}),$$

and forms $\mathcal{I}_{\text{rest}} := \mathcal{K}^* \setminus \{\text{sok}^*\}$, denoting the set of the remaining signatures of knowledge (as in game $\mathbf{G}_7$).

- If $\mathcal{I}_{\text{rest}} = \emptyset$, then B_0 aborts. (Indeed, in this case Bad_0 cannot occur by definition.)
- Otherwise, B_0 samples a uniformly random index $I \leftarrow \mathcal{I}_{\text{rest}}$ and outputs to its simulation-soundness challenger the pair

$$(\text{stm}_{0,I}, \pi_{0,I}), \quad \text{where} \quad \text{stm}_{0,I} := (g, h, v, \zeta_I, c_{0,I,1}, c_{0,I,2}).$$

Since $\text{Ver}(\mathcal{L}, \text{trx}^*) = 1$ implies that $\text{NIZK.V}_0^H(\text{stm}_{0,\ell}, \pi_{0,\ell}) = 1$ for all tuples in $\mathcal{K}^*$, we have in particular that $\text{NIZK.V}_0^H(\text{stm}_{0,I}, \pi_{0,I}) = 1$.

Now suppose event Bad_0 occurs in Game $\mathbf{G}_7$. By definition of event Bad_0 , there exists an index $j^* \in \mathcal{I}_{\text{rest}}$ such that $(\text{stm}_{0,j^*}, \pi_{0,j^*})$ is accepting but $\text{stm}_{0,j^*} \notin \mathcal{L}_0$. Since B_0 samples I uniformly from $\mathcal{I}_{\text{rest}}$, we have

$$\Pr[I = j^* \mid \text{Bad}_0] \geq \frac{1}{|\mathcal{I}_{\text{rest}}|} \geq \frac{1}{k}.$$

Moreover, because $I \in \mathcal{I}_{\text{rest}}$ excludes the special signature of knowledge sok^* , the pair $(\text{stm}_{0,I}, \pi_{0,I})$ is *never* equal to the unique statement/proof pair $(\text{stm}_0^*, \pi_0^*)$ for which B_0 queried the simulator oracle. Hence, conditioned on $I = j^*$, B_0 outputs a *fresh* accepting proof for a false statement, after having obtained at least one simulated proof, and therefore wins the simulation-soundness game.

Therefore,

$$\Pr[\text{Bad}_0] \leq k \cdot \text{Adv}_{\text{NIZK}_0}^{\text{ss}, B_0}(1^\lambda).$$

Combining bounds. Plugging $\Pr[\text{Bad}_{\text{deg}}] \leq 1/p$ and the bounds above into (6) yields

$$\Pr[\text{Bad}] \leq \frac{1}{p} + n \cdot \text{Adv}_{\text{GS.NIZK}_1}^{\text{snd}, B_1}(1^\lambda) + t \cdot \text{Adv}_{\text{NIZK}_2}^{\text{snd}, B_2}(1^\lambda) + k \cdot \text{Adv}_{\text{NIZK}_0}^{\text{ss}, B_0}(1^\lambda).$$

In particular, if the right-hand side is negligible in λ , then the probability of Bad (equivalently, $\mathbf{G}_7$ aborting) happening is negligible. Using known bounds on (simulation-)soundness error of these NIZKs, we get

$$|\Pr[\mathbf{G}_7 \Rightarrow 1] - \Pr[\mathbf{G}_6 \Rightarrow 1]| \leq \Pr[\text{Bad}] \leq \frac{1}{p} + \frac{k(q_h + 1)}{p} + \frac{t \cdot q_h}{p} + n \cdot \varepsilon_{\text{snd}}.$$

GAME $\mathbf{G}_8$: In this game, we now change the winning condition. Instead of computing S_{rec} , we directly compute S_{sok} and check if $S^* = S_{\text{sok}}$. Unless game $\mathbf{G}_7$ aborts, both winning conditions are equivalent. We therefore have

$$\Pr[\mathbf{G}_8 \Rightarrow 1] = \Pr[\mathbf{G}_7 \Rightarrow 1].$$

The subsequent games $\mathbf{G}_9$ to $\mathbf{G}_{14}$ are identical to Games $\mathbf{G}_1$ to $\mathbf{G}_6$ in the security proof of *Illusi* (cf. Theorem 1). In these games, we further modify the transcript trx_{i^*} from the special honest party i^* . By the previous hybrids, the signature of knowledge sok^* is already generated without referencing h^s . We now apply the same transformations to the remaining components of trx_{i^*} as in the proof of *Illusi*. This is possible since a transcript of *Illusia* consists of a transcript of *Illusi* together with an additional signature of knowledge. This sequence of games then guarantees the following: In the generation of trx_{i^*} , the value s is referenced only when computing the commitment $\text{com}[0] = g^s$ and when checking the winning condition $S^* = S_{\text{sok}}$ using h^s . We can easily build a reduction to co-CDH as follows.

Building a Reduction. We construct an efficient reduction R that uses an adversary A winning Game $\mathbf{G}_{14}$ to solve the co-CDH problem. On input $(g^a, g^b, \hat{g}^a)$, R proceeds as follows.

1. R sets up the public parameters exactly as in game $\mathbf{G}_{14}$, except that $h := g^b$.
2. R sets up the special honest party i^* 's transcript tr_{i^*} exactly as in game $\mathbf{G}_{14}$, except that it generates $\text{com}[0] := g^a$.
3. Eventually, A outputs $(\text{tr}^*, S^*, \mathcal{S}^*, \{(S_j^*, \mu_j^*)\}_{j \in S^* \cap \mathcal{C}})$. Assuming the game returns 1, R proceeds as follows.
4. R parses tr^* , determines the multiset $\mathcal{K}^* = \{\langle \zeta_\ell, \theta_\ell \rangle_{j_\ell}\}_{\ell \in [k]}$, counts m (the number of copies of the special tuple sok^*), and computes for every $\ell \in \mathcal{I}_{\text{rest}}$ the extractions

$$X_\ell := \frac{c_{0,\ell,2}}{(c_{0,\ell,1})^\alpha},$$

where $v = g^\alpha$ for an $\alpha \leftarrow \mathbb{Z}_p$ as in game $\mathbf{G}_{14}$.

5. Using the forgery S^* , R now computes

$$X_{\text{co-cdh}} := \left(S^* / \prod_{\ell \in \mathcal{I}_{\text{rest}}} X_\ell \right)^{1/m},$$

and returns $X_{\text{co-cdh}}$ to its co-CDH challenger.

Correctness. If A wins game $\mathbf{G}_{14}$, then

$$S^* = S_{\text{sok}} = (h^a)^m \cdot \prod_{\ell \in \mathcal{I}_{\text{rest}}} X_\ell.$$

Therefore,

$$X_{\text{co-cdh}} = (h^{am})^{1/m} = h^a = (g^b)^a = g^{ab}.$$

As a result, if A wins game $\mathbf{G}_{14}$ with some probability, then R wins the co-CDH game with the same probability. We therefore get

$$\Pr[\mathbf{G}_{14} \Rightarrow 1] \leq \varepsilon_{\text{co-cdh}}.$$

□