Pre-Constructed Publicly Verifiable Secret Sharing and Applications

Karim Baghery, Noah Knapen, Georgio Nicolas, and Mahdi Rahimi

COSIC, KU Leuven, Leuven, Belgium
karim.baghery@kuleuven.be, noah.knapen@gmail.com,
georgio.nicolas@esat.kuleuven.be, mahdi.rahimi@kuleuven.be

March 30, 2025

Abstract. Conventional Publicly Verifiable Secret Sharing (PVSS) protocols allow a dealer to share a secret among n parties without interaction, ensuring that any $t+1$ parties (where $t+1 \leq n$) can recover the secret, while anyone can publicly verify the validity of both the individual shares and the reconstructed secret. PVSS schemes are shown to be a key tool in a wide range of practical applications. In this paper, we introduce Pre-constructed PVSS (PPVSS), an extension of standard PVSS schemes, highlighting its enhanced utility and efficiency in various protocols. Unlike standard PVSS, PPVSS requires the dealer to publish a commitment or encryption of the main secret and incorporates a novel secret reconstruction method. We show that these refinements make PPVSS more practical and versatile than conventional PVSS schemes. To build a PPVSS scheme, we first point out that the well-known PVSS scheme by Schoenmakers (CRYPTO'99) and its pairing-based variant presented by Heidarvand and Villar (SAC'08) can be seen as special cases of PPVSS, where the dealer also publishes a commitment to the main secret. However, these protocols are not practical for many applications due to efficiency limitations and are less flexible compared to a standard PPVSS scheme. To address this, we propose a general strategy for transforming a Shamir-based PVSS scheme into a PPVSS scheme. Using this strategy, we construct two practical PPVSS schemes in both the Random Oracle (RO) and plain models, grounded in state-of-the-art PVSS designs. Leveraging the new RO-based PPVSS scheme, we revisit some applications and present more efficient variants. Notably, we propose a new universally verifiable e-voting protocol that improves on the alternative scheme by Schoenmakers (CRYPTO'99), reducing the verification complexity with m voters from $O(n^2m)$ to $O(nm)$ exponentiations–a previously unattainable goal with standard PVSS schemes. Our implementation results demonstrate that both our proposed PPVSS schemes and the new universally verifiable e-voting protocol significantly outperform existing alternatives in terms of efficiency.

Keywords: Publicly Verifiable Secret Sharing · PVSS · Pre-Constructed Publicly Verifiable Secret Sharing · Universally Verifiable E-Voting

1 Introduction

Secret sharing schemes allow a dealer to divide a secret into shares and distribute them among n participants, such that any subset of $t+1$ participants can collaborate to reconstruct the secret, while any subset of t or fewer participants learn nothing about it. This $(t+1, n)$-threshold structure ensures resilience and confidentiality in settings where trust is distributed among multiple parties. Secret sharing protocols have found applications in secure multi-party computation, threshold cryptography, and decentralized systems such as blockchain.

Verifiable Secret Sharing. In the well-known secret sharing protocol proposed by Shamir [23], a dealer shares a secret f_0 among n parties using a secret degree-t polynomial $f(X)$. To do this, the dealer sets $f(0) := f_0$, and each party P_i privately receives $f(i)$ as their individual share of the main secret $f_0 = f(0)$. However, basic secret sharing schemes, while providing security guarantees, assume that both the dealer and participants are honest. This assumption is problematic in adversarial environments, where malicious actors could compromise the protocol by distributing incorrect shares or refusing to participate honestly. To address this issue, *Verifiable Secret Sharing* (VSS) schemes were developed, allowing parties to verify the correctness of their shares [1, 2, 4, 12, 15, 21]. Feldman's VSS scheme [15] is a pioneering work in this area, utilizing homomorphic commitments to enable verification of the shares in Shamir's scheme, ensuring that they are consistent with a unique secret polynomial. Feldman's approach guarantees that the shares are correct, but it does not hide the secret polynomial's coefficients, limiting its privacy guarantees and making it suitable only for sharing high-entropy secrets. Pedersen's VSS [21] extended this idea by using his homomorphic commitment and incorporating additional randomness, providing both verifiability and secrecy of the shared values. A recent work by Atapoor, Baghery, Cozzo, and Pedersen [1] presented a highly efficient VSS scheme based on Shamir's protocol, using hash-based (i.e., non-homomorphic) commitments.

Publicly Verifiable Secret Sharing. In VSS schemes, the share verification phase might require interaction between the dealer and the parties, and verification can only be done by the shareholders. As cryptographic protocols evolved, the need arose for schemes that allow external parties to verify the correctness of the process. *Publicly VSS* (PVSS) was proposed to meet this demand, allowing also external auditors to verify the shares and the reconstructed secret [3, 7, 8, 17, 18, 22]. In PVSS schemes, a dealer publishes shares encrypted under the public keys of parties along with a Non-Interactive Zero-Knowledge (NIZK) proof, enabling anyone to verify the correctness of the shares. Schoenmakers' PVSS scheme [22] is a well-known example of such a scheme, built on Feldman's VSS in the Random Oracle (RO) model. His protocol has been particularly useful for applications requiring transparency, such as universally verifiable e-voting, robust public-key infrastructure, threshold software key escrow, and many more [22]. However, while Schoenmakers' PVSS scheme guarantees public verifiability, it suffers from inefficiencies, particularly in the verification phase, where $O(n^2)$ exponentiations are required. This computational overhead

makes Schoenmakers' scheme unsuitable for applications requiring scalability, such as large-scale elections or randomness beacons in decentralized protocols.

Despite its foundational role, the complexity of Schoenmakers' PVSS scheme has driven the development of more efficient alternatives. Several works have sought to reduce the computational and communication costs associated with PVSS. Heidarvand and Villar [19] proposed a pairing-based variant of Schoenmakers' PVSS scheme in the plain model, but it still requires $O(nt)$ exponentiations and $O(n)$ pairing operations in the verification phase, which are computationally expensive. In ACNS'17, Cascudo and David [7] improved upon Schoenmakers' work [22] and its pairing-based variant [19] by designing two PVSS schemes that reduce the complexity of share verification to $O(n)$ exponentiations, making the schemes more suitable for large-scale applications. This improvement in verification was achieved through a coding-theory technique, employing a random codeword from the dual code of the Reed-Solomon code used for secret sharing. The authors [7] also used their proposed PVSS schemes to construct the SCRAIP protocol, a publicly verifiable randomness beacon protocol in the honest-majority setting. Several follow-up works [3, 8, 10] have proposed new PVSS schemes that are concretely more efficient than those in [7]. For a detailed comparison, we refer to Table 4 in the appendix. We highlight that PVSS schemes are mainly categorized into two classes, each suited to different applications: the schemes where the secret is g^{f_0}, and those where it is f_0, with g as the generator of a cyclic group of prime order. The PVSS schemes proposed in [3, 7, 8, 10, 22] belong to the first category, where each party obtains a share g^{f_i} and, in the reconstruction phase, the parties collectively reconstruct g^{f_0}. Conversely, the schemes in [9, 18] belong to the second category.

Our Contributions. Our contributions can be summarized as follows:

Pre-Constructed Publicly Verifiable Secret Sharing. As the first contribution of this paper, we define Pre-Constructed PVSS (PPVSS) as an extension of standard PVSS schemes. PPVSS offers all the capabilities of standard PVSS but also requires the dealer to pre-construct and publish a commitment/encryption of the main secret. This subtle modification allows us to have a novel secret reconstruction approach specific to PPVSS schemes. Beyond a novel reconstruction approach, by enabling the dealer to publish a commitment to the main secret and prove its well-formedness alongside other shares in the same round, a PPVSS scheme provides enhanced functionality compared to PVSS schemes. Using this new reconstruction approach, which we call *optimistic reconstruction*, the dealer can open a single commitment to the (pre-constructed) secret during the reconstruction phase, rather than involving at least $t + 1$ shareholders to reconstruct the same secret or opening the entire secret polynomial, as done in some current PVSS-based protocols [8]. Opening the entire polynomial by the dealer incurs a communication cost of at least $O(t\lambda)$ bits and a computation cost of $O(t)$ for each verifier, where λ and t are the security and threshold parameters, respectively. In some applications of PVSS schemes, each player usually acts as the dealer once. Since we assume the majority of parties are honest, most dealers can use the proposed optimistic reconstruction approach. If reconstruction of a

malicious dealer's secret is needed and the dealer does not open the commitment, any $t+1$ honest parties can use their shares along with the classic reconstruction approach–here referred to as the *pessimistic reconstruction*–to reconstruct the secret. Our analysis show that a PPVSS scheme can be more versatile and broadly applicable than standard PVSS schemes.

Schoenmakers' PVSS is a Special PPVSS. We show that Schoenmakers' PVSS and its pairing-based variant [19,22], which are built on Feldman VSS [15], can be seen as a special type of PPVSS. In addition to encrypting the shares, the dealer also publishes a commitment to the main secret. However, his scheme requires $O(n^2)$ exponentiations during verification and is not as flexible as our defined and proposed PPVSS schemes. The reason for this reduced flexibility is that, following Feldman VSS [14], in Schoenmakers' (P)PVSS scheme [22], the commitment to the main secret has to be based on the same group generator used in other commitments and cannot be based on the public key of a specific party. Due to this limitation, while constructing threshold binding ElGamal based on his proposed (P)PVSS scheme [22, Section 6.1], Schoenmakers had to separately encrypt $g^{f(0)}$ under the public key pk_0 of the target party and use a Chaum-Pedersen protocol [11] to prove that the commitment $C_0 = g^{f(0)}$ commits to the same value encrypted under the target public key pk_0. This approach incurs additional computational and communication costs. As we demonstrate later, in our proposed PPVSS schemes, this can be achieved for free, by simply replacing the commitment key with the target public key pk_0.

In [7], Cascudo and David presented two PVSS protocols that improve Schoenmakers' scheme and its paring-based variant [19, 22], by reducing the number of exponentiations required for verification from $O(n^2)$ to $O(n)$. All the follow-up works, like [3, 7, 8, 10, 22], also present a better efficiency compared to Schoenmakers' PVSS and its paring-based variant [19, 22]. However, all the follow-up PVSS schemes with $O(n)$ verification complexity lack the pre-constructability feature of Schoenmakers' original design. Since the goal of PVSS is typically to allow public verifiability of the shares and the dealer only needs to encrypt the shares under each participant's public key. The main secret is not shared with any individual, and thus the pre-constructability found in Schoenmakers' scheme was inherited from Feldman VSS and not carried forward into the follow-up and more efficient PVSS designs. Our analysis show that this property is crucial for some applications like universally verifiable e-voting and robust public-key infrastructure [22].

General Strategy for Building PPVSS. As the next contribution of this paper, we introduce a general strategy to convert a Shamir-based PVSS scheme into an efficient PPVSS scheme with minimal computational and communication overhead. Our approach is inspired by the design of the Shamir secret sharing scheme. We note that in Shamir's scheme, the individual shares $f(i)$ and the main secret $f(0)$ are all distinct points on the same secret polynomial $f(X)$. In a standard Shamir-based PVSS scheme, given the parties' public keys $\{\mathsf{pk}_i\}_{i=1}^n$ and a secret degree-t polynomial $f(X)$, a dealer generates a Non-Interactive Zero-Knowledge (NIZK) proof for $C_i = \mathsf{Enc}(f(i), \mathsf{pk}_i)$ for $i = 1, \ldots, n$, where

$n \geq 2t + 1$ and Enc is a public key encryption algorithm. We show that, with minimal modifications, the dealer can extend the NIZK proof scheme to the case $i = 0, 1, \ldots, n$, where $f(0)$ denotes the main secret and pk_0 is a commitment key or the public key of a designated party. Notably, in terms of efficiency, the computational and communication overhead in the resulting PPVSS scheme is negligible compared to the original PVSS scheme, increasing by only a factor of $1/n$ for both the dealer and the verifier.

Practical PPVSS Schemes. Using the proposed strategy, we build two practical PPVSS schemes based on the PVSS schemes proposed in [3,7,8]. Our focus is on these practical PVSS schemes, where the secret is g^{f_0}, however, we believe that our strategy is general enough to be used with other Shamir-based PVSS schemes as well, including the ones that the secret is f_0 [9]. Our initial PPVSS scheme can be seen as a more efficient alternative to Schoenmakers' PPVSS scheme [22], as both are constructed in the RO model. While the second proposed PPVSS scheme is constructed in the plain model and can be seen as a more efficient alternative to the paring-based variant of Schoenmakers' PPVSS scheme [22], proposed by Heidarvand and Villar [19]. In terms of efficiency, the new PPVSS schemes are more efficient than the (special) PPVSS schemes proposed in [19, 22], and notably reducing the verification complexity from $O(n^2)$ to $O(n)$. Compared to the original PVSS schemes [3, 7, 8], the resulting PPVSS schemes have very close efficiency, while presenting new functionalities, and enabling a more efficient reconstruction approach. Table 1 provides a summary of performance metrics for the proposed PPVSS schemes.

More Efficient Universally Verifiable E-Voting Protocol. Schoenmakers demonstrated how his (P)PVSS scheme could be applied to various use cases, including a universally verifiable e-voting protocol [22]. In Schoenmakers' voting protocol, the computational cost for the verification step is approximately $m(nt+4.5n)$ exponentiations, where m and n are the number of voters and talliers, respectively, and $t = \lfloor \frac{n-1}{2} \rfloor$ is the threshold. In some cases, such as boardroom elections, each participant may act as both a voter and a tallier, meaning $m = n$. In such cases, the verification phase requires $O(n^3)$ exponentiations.

Table 1. A comparison of new PPVSS schemes with those of Schoenmakers [22] and its paring-based variant [19]. BC: Dealer's Broadcast size, Dow.: Download size by a verifier, Opt. Reconst.: Optimistic Reconstruction, Pes. Reconst.: Pessimistic Reconstruction, PDL: Polynomial Discrete Logarithm, DDH: Decisional Diffie-Hellman, DBS: Decisional Bilinear Square, RO: Random Oracle, Plain: Plain Model, n: Number of parties, t: threshold parameter ($t \approx n/2$), $P_{\mathbb{G}}$: Pairing Operation, $E_{\mathbb{G}}$: Exponentiation in group $\mathbb{G}$, $M_{\mathbb{G}}$: Multiplication in group $\mathbb{G}$, $\mathcal{PE}$: degree-t Polynomial Evaluation, $|\mathbb{G}|$: $\mathbb{G}$ element size, $|\mathbb{Z}_q|$: $\mathbb{Z}_q$ element size.

PPVSS & Security	Share	Broadcast & Dow	Verification	Opt. Reconst.	Pes. Reconst.
Schoenmakers [22] (DDH, RO)	$4.5n$ $E_{\mathbb{G}}$ $1n$ $\mathcal{PE}$	BC: $1.5n\|\mathbb{G}\| + n\|\mathbb{Z}_q\|$ Dow: $2.5n\|\mathbb{G}\| + n\|\mathbb{Z}_q\|$	$nt + 4n$ $E_{\mathbb{G}}$ + $nt + 2n$ $M_{\mathbb{G}}$	1 $E_{\mathbb{G}}$	$5t$ $E_{\mathbb{G}}$ + t $M_{\mathbb{G}}$
Sec. 4.2 (PDL, DDH, RO)	$2n$ $E_{\mathbb{G}}$ $2n$ $\mathcal{PE}$	BC: $n\|\mathbb{G}\| + 0.5n\|\mathbb{Z}_q\|$ Dow: $2n\|\mathbb{G}\| + 0.5n\|\mathbb{Z}_q\|$	$2n$ $E_{\mathbb{G}}$ + n $\mathcal{PE}$ $+n$ $M_{\mathbb{G}}$	1 $E_{\mathbb{G}}$	$5t$ $E_{\mathbb{G}}$ + t $M_{\mathbb{G}}$
Hei-Vil [19] (DBS, Plain)	$1.5n$ $E_{\mathbb{G}}$ $1n$ $\mathcal{PE}$	BC: $1.5n\|\mathbb{G}\|$ Dow: $2.5n\|\mathbb{G}\|$	nt $E_{\mathbb{G}}$ + $2n$ $P_{\mathbb{G}}$ + nt $M_{\mathbb{G}}$	1 $E_{\mathbb{G}}$	$2t$ $P_{\mathbb{G}}$ + t $E_{\mathbb{G}}$ + t $M_{\mathbb{G}}$
Sec. 4.3 (DBS, Plain)	$2n$ $E_{\mathbb{G}}$ $1n$ $\mathcal{PE}$	BC: $2n\|\mathbb{G}\|$ Dow: $3n\|\mathbb{G}\|$	n $E_{\mathbb{G}}$ + $2n$ $P_{\mathbb{G}}$ + n $M_{\mathbb{G}}$	1 $E_{\mathbb{G}}$	$2t$ $P_{\mathbb{G}}$ + t $E_{\mathbb{G}}$ + t $M_{\mathbb{G}}$

As an additional contribution of this paper, leveraging our proposed RO-based PPVSS scheme, we revisit the universally verifiable e-voting protocol presented by Schoenmakers [22] and introduce an improved version. The new e-voting protocol outperforms Schoenmakers' construction in several ways, notably reducing the verification phase to $O(mn)$ exponentiations rather than $O(mn^2)$. This improvement makes the new e-voting protocol viable for elections with a large value of n. To assess the performance of the new e-voting protocol, we present a prototype Rust implementation of both the original and our revisited protocols. Detailed performance evaluations and comparisons are provided in Section 5.3. For $(n, t) = (512, 255)$, we demonstrate that the new e-voting scheme requires $90\times$ less time for a single ballot verification. Additionally, for the same parameter sizes, the improved e-voting protocol requires 40% less time for ballot casting and 40% less (broadcast) communication by each voter.

Outline. In Sec. 2, we present some preliminaries. In Sec. 3, we discuss the definitions for PPVSS and introduce a general strategy to build a PPVSS scheme. In Sec. 4, we present two practical PPVSS schemes along with some implementation results. In Sec. 5, we use the new RO-based PPVSS scheme and revisit Shoenmakers' e-voting protocol and evaluate its performance with a prototype implementation. Finally, in Sec. 6, we conclude the paper.

2 Preliminaries

Notation. We denote by λ the security parameter. A function f is called *negligible in* X, written $\mathsf{negl}(X)$, if for any constant c, there exists some X_0 such that $f(X) < X^{-c}$ for all $X > X_0$. When a function is negligible in the security parameter λ, we refer to it simply as negligible. The symbol $\leftarrow$ denotes uniform sampling from a set Ξ, for example, $x \leftarrow \Xi$. Throughout this paper, p and q represent two large primes such that q divides $p-1$. We define $\mathbb{G}$ as the unique subgroup of $\mathbb{Z}_p^\star$ with order q, and let g be a generator of the cyclic group $\mathbb{G}$ of prime order q. The group $\mathbb{G}$ is chosen such that computing discrete logarithms (DL) within $\mathbb{G}$, i.e., finding $\log_g h$ for some $h \in \mathbb{G}$, is computationally hard.

2.1 Sigma Protocols

Next, we recall the definition of sigma protocols (Σ-protocols). Here the algorithms are Probabilistic Polynomial-Time (PPT), unless mentioned. Let $X = X(\lambda)$ and $W = W(\lambda)$ be sets. Let R be a relation on $X \times W$ that defines a language $L = \{x \in X : \exists w \in W, R(x, w) = 1\}$. Given $x \in L$, an element $w \in W$ such that $R(x, w) = 1$ is called a witness. Let relation generator $\mathcal{R}$ be a PPT algorithm such that $\mathcal{R}(1^\lambda)$ outputs pairs (x, w) such that $R(x, w) = 1$.

A sigma-protocol (Σ-protocol) for the relation R is a 3-round interactive protocol between two PPT algorithms: a prover P and a verifier V. P holds a witness w for $x \in L$ and V is given x. In first round, P sends a commitment value a to V, and then in second round, V answers with a randomly sample challenge value d. A Σ-protocol is called public-coin if the challenge value d is a public

and random string rather than something derived in a complex or private way. Finally, P answers with a response z, and V verifies the proof and outputs `true` or `false`. The triple $\mathsf{trans} := (a, d, z)$ is called a transcript of the Σ-protocol. A Σ-protocol is supposed to satisfy *Completeness*, *Honest Verifier Zero-Knowledge* (HVZK), and *Special Soundness* defined below.

Definition 1 (Completeness). *A Σ-protocol with parties (P, V) is* complete for $\mathcal{R}$*, if for all $(x, w) \in R$, the honest V will always accept the honest P.*

Definition 2 (HVZK). *A Σ-protocol with parties (P, V) satisfies* HVZK for $\mathcal{R}$*, if there exists a PPT algorithm $\mathcal{S}$ that given $x \in X$, can simulate the* trans *of the scheme, s.t. for all $x \in L$, $(x, w) \in R$,*

$$\mathsf{trans}(P(x, w) \leftrightarrow V(x)) \approx \mathsf{trans}(\mathcal{S}(x))$$

where $\mathsf{trans}(P(\cdot) \leftrightarrow V(\cdot))$ indicates the transcript of the Σ-protocol with (P, V), and $\approx$ denotes the indistinguishability of transcripts.

Definition 3 (Special Soundness). *A Σ-protocol with parties (P, V) is* special sound for $\mathcal{R}$*, if there exists a PPT extractor $\mathcal{E}$, such that for any $x \in L$, given two valid transcripts (a, d, z) and (a, d', z') for the same message a but $d \neq d'$, then $\mathcal{E}(a, d, z, d', z')$ outputs a witness w for the relation R.*

In the Random Oracle (RO) model, using Fiat-Shamir transform [16], a public-coin, complete, HVZK, and special soundness Σ-protocol can be turned into a Non-Interactive Zero-Knowledge (NIZK) proof or argument of knowledge.

Chaum-Pedersen Protocol for DL Equality. Let $\mathbb{G}$ be a group with hard DL, and g, h be two group elements, where g is the group generator. Let a prover aim to convince a verifier that for the public statement g, h, a, b, he knows a witness x which holds in the following relation,

$$R_{DLEQ} = \{(g, h, a, b), x \mid a = g^x \wedge b = h^x\}. \tag{1}$$

This relation is known as DL EQuality (DLEQ). In [11], Chaum and Pedersen introduced an efficient NIZK proof of knowledge for DLEQ, as summarized in Fig. 1. This protocol is widely employed in various cryptographic protocols, including threshold decryption, e-voting systems, PVSS schemes, etc.

Prover: Given the statement $(g, h, a, b) \in \mathbb{G}$ and the witness value $x \in \mathbb{Z}_q$, proceed as follows and output a proof π.

1. Sample $r \leftarrow\$ \mathbb{Z}_q$ uniformly at random; and set $c_1 = g^r$ and $c_2 = h^r$.
2. Set $d \leftarrow \mathcal{H}(a, b, c_1, c_2)$, where $\mathcal{H}$ is a random oracle.
3. Set $z = r + d \cdot x \mod q$; and Return $\pi := (d, z)$

Verifier: Given the statement $(g, h, a, b) \in \mathbb{G}$ and the proof $\pi = (d, z)$, checks if $d = \mathcal{H}(a, b, \frac{g^z}{a^d}, \frac{h^z}{b^d})$ and outputs `true` or `false`.

Fig. 1. Chaum-Pedersen NIZK proof of knowledge for DLEQ [11].

NIZK Proof of Binary Vote. In the ballot casting phase of the e-voting protocol proposed by Schoenmakers [22], a voter V casts a binary vote $v \in \{0,1\}$ by running the sharing phase for the (P)PVSS scheme, using a random secret $s \in \mathbb{Z}_q$, and computing the value $U = G^{s+v}$. In addition, the voter generates a proof π_U to show that $v \in \{0,1\}$ without revealing any information on v. The proof π_U refers to the commitment $C_0 = g^s$, which is published as part of the (P)PVSS transcript, and proves that:

$$\log_G U = \log_g C_0 \ \vee \ \log_G U = 1 + \log_g C_0.$$

The proof π_U is constructed using the technique of [13], and its protocol can be summarized as follows:

1. The prover sets $a_v = g^w$ and $b_v = G^w$ for random $w \in_R \mathbb{Z}_q$. The prover also sets $a_{1-v} = g^{r_{1-v}} C_0^{d_{1-v}}$ and $b_{1-v} = G^{r_{1-v}} (U/G^{1-v})^{d_{1-v}}$, for random $r_{1-v}, d_{1-v} \in_R \mathbb{Z}_q$. The prover sends a_0, b_0, a_1, b_1 in this order to the verifier.
2. The verifier sends a random challenge $c \in_R \mathbb{Z}_q$ to the prover.
3. The prover sets $d_v = c - d_{1-v} \pmod q$ and $r_v = w - s d_v \pmod q$, and sends d_0, r_0, d_1, r_1 in this order to the verifier.
4. The verifier checks that $c = d_0 + d_1 \pmod q$ and that

$$a_0 = g^{r_0} C_0^{d_0}, \quad b_0 = G^{r_0} U^{d_0}, \quad a_1 = g^{r_1} C_0^{d_1}, \quad \text{and} \quad b_1 = G^{r_1} \left(\tfrac{U}{G}\right)^{d_1}.$$

The protocol is proven to satisfy completeness, special soundness and honest verifier zero-knowledge; hence, its non-interactive version achieves ZK and releases no information on v in the random oracle model. This protocol is also used in our revised e-voting protocol.

2.2 Bilinear Groups and Computational Assumptions

We present one of our proposed PPVSS schemes over a symmetric bilinear group, which are defined as below.

Definition 4 (Bilinear Group). *A bilinear group is a tuple $(q, \mathbb{G}, \mathbb{G}_T, e)$, where:*

- *$\mathbb{G}$ and $\mathbb{G}_T$ are groups of prime order q,*
- *$e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map with the following properties:*
  - ***Bilinearity:*** *$e(g^\alpha, g^\beta) = e(g,g)^{\alpha\beta}$ for every $g \in \mathbb{G}$ and $\alpha, \beta \in \mathbb{Z}_q$.*
  - ***Non-degeneration:*** *$e(g,g) \neq 1$ unless $g = 1$.*
  - ***Efficiency:*** *There are efficient algorithms for computing group operations in $\mathbb{G}$, $\mathbb{G}_T$, and evaluating $e(x,y)$ for $x, y \in \mathbb{G}$.*

The security of Cascudo and David's paring-based PVSS scheme [7] is proven under the Decisional Bilinear Square (DBS) assumption [19], which was shown to be related to the Decisional Bilinear Diffie-Hellman assumption.

Definition 5 (Decisional Bilinear Square (DBS) Assumption [19]). *Let $(q, \mathbb{G}, \mathbb{G}_T, e)$ be a bilinear group. For a generator $g \in \mathbb{G}$, random values $\mu, \nu, s \leftarrow \mathbb{Z}_q$, and given $u = g^\mu$ and $v = g^\nu$, the following probability distributions are computationally indistinguishable:*

$$D_0 = (g, u, v, T_0 = e(u,u)^\nu) \ \approx \ D_1 = (g, u, v, T_1 = e(u,u)^s)$$

Polynomial Discrete Problem. In [3], Baghery generalize Discrete Logarithm (DL) relation over polynomials and introduce the Polynomial Discrete Logarithm (PDL) relation denoted as R_{PDL}, which is defined as follows,

$$R_{PDL} = \{(g, x_i, F_i), f(X) \mid F_i = g^{f(x_i)} \text{ for } i = 1, 2, \ldots, n\}.$$

Here, $f(X) \in \mathbb{Z}_q[X]_t$ is a (at most) degree $t \leq n-1$ witness polynomial with coefficients from $\mathbb{Z}_q$, and $\{x_i\}_{i=1}^n$ are n *distinct* elements from $\mathbb{Z}_q$. The R_{PDL} relation is based on the PDL problem, defined as follows, which is shown to be reducible to the DL problem [3].

Definition 6 (Polynomial Discrete Logarithm Problem [3]). *Let $\mathbb{G}$ be a finite cyclic group of order q generated by g. Given $F_1, \ldots, F_n$ from $\mathbb{G}$ and distinct elements $x_1, \ldots, x_n$ from $\mathbb{Z}_q$, find a polynomial $f(X) \in \mathbb{Z}_q[X]_t$ of (at most) degree t, where $0 \leq t \leq n-1$, such that $F_i = g^{f(x_i)}$ for all $i = 1, \ldots, n$.*

In other words, an algorithm $\mathcal{A}$ has advantage ϵ in solving PDL in $\mathbb{G}$ if

$$\Pr[\mathcal{A}(x_1, \ldots, x_n, g, g^{f(x_1)}, \ldots, g^{f(x_n)}) = f(X)] \geq \epsilon$$

where $f(X) \in \mathbb{Z}_q[X]_t$ is (at most) a degree-t polynomial with $0 \leq t \leq n-1$, and the probability is over the random choice of generator $g \in \mathbb{G}^*$ and the distinct choice of $x_1, \ldots, x_n$ in $\mathbb{Z}_q$.

Baghery [3] also presented a Non-Interactive Zero-Knowledge (NIZK) Proof-of-Knowledge (PoK) scheme π_{PDL}, that allows a prover to prove knowledge of a witness for R_{PDL} relation. The verifiability property of $\mathbf{\Pi_S}$ PVSS scheme [3, 8] is proven under the PDL assumption.

2.3 Shamir Secret Sharing

A $(t+1, n)$-Shamir secret sharing scheme [23] allows n parties to individually hold a share x_i of a common secret f_0, such that any subset of t parties or less are not able to learn any information about the secret f_0, while any subset of at least $t+1$ parties are able to efficiently reconstruct the common secret f_0. In more detail, this is achieved via polynomial interpolation over the field $\mathbb{Z}_q$. A common polynomial $f(x) \in \mathbb{Z}_q[x]_t$ is chosen, such that the secret f_0 is set to be its constant term, namely $f_0 = f(0)$. Each party P_i for $i \in \{1, \cdots, n\}$ is assigned the secret share $f_i = f(i)$. Then any subset $Q \subseteq \{1, \ldots, n\}$ of at least $t+1$ parties can reconstruct the secret x_0 via Lagrange interpolation by computing $f_0 = f(0) = \sum_{i \in Q} f_i \cdot L_{0,i}^Q$, where

$$L_{0,i}^Q := \prod_{j \in Q \setminus \{i\}} \frac{j}{j-i} \pmod q.$$

are the Lagrange basis polynomials evaluated at 0.

2.4 Coding Theory and Dual Codes

We define a $[n, t+1, d]$ code RS to be a linear error-correcting code over $\mathbb{Z}_q$ of length n, dimension $t+1$, and minimum distance d. Its dual code $\mathsf{RS}^\intercal$ is the vector space that consists of all vectors $\mathbf{v}^\intercal \in \mathbb{Z}_q^n$ such that $\langle \mathbf{c}, \mathbf{v}^\intercal \rangle = 0$ for all $\mathbf{c}$ in RS. The dual code $\mathsf{RS}^\intercal$ of an $[n, t+1, d]$ code RS is an $[n, n-t-1, d^\intercal]$ code (for some $d^\intercal$). In this work, we will use the following basic linear algebra fact.

Lemma 1 ([7]). *If $\mathbf{c}' \in \mathbb{Z}_q^n \setminus \mathsf{RS}$, and $\mathbf{v}^\intercal$ is chosen uniformly at random in $\mathsf{RS}^\intercal$, then the probability that $\langle \mathbf{c}', \mathbf{v}^\intercal \rangle = 0$ is exactly $1/q$.*

Proof. By linearity, a $\mathbf{v}^\intercal \in \mathsf{RS}^\intercal$ is orthogonal to $\mathbf{c}'$ if and only if it is also orthogonal to every vector in the code D spanned by $\mathbf{c}'$ and RS, i.e., if and only if $\mathbf{v}^\intercal \in \mathsf{D}^\intercal$. Since $\mathbf{c}' \notin \mathsf{RS}$, the dimension of D is $t+2$, and hence the space $\mathsf{D}^\intercal$ has dimension $n-t-2$. Therefore, if $\mathbf{v}^\intercal$ is chosen uniformly at random in $\mathsf{RS}^\intercal$, the probability that $\langle \mathbf{c}', \mathbf{v}^\intercal \rangle = 0$ is

$$\frac{\#(\mathsf{D}^\intercal)}{\#(\mathsf{RS}^\intercal)} = \frac{q^{n-t-2}}{q^{n-t-1}} = \frac{1}{q}.$$

Moreover, in this work, we will always assume $n < q$, and we will use Reed-Solomon codes RS of the following form:

$$\mathsf{RS} = \{(f(1), f(2), \ldots, f(n)) : f(X) \in \mathbb{Z}_q[X], \deg(f(X)) \leq t\},$$

where $f(X)$ ranges over all polynomials in $\mathbb{Z}_q[X]$ of degree at most t. This is an $[n, t+1, n-t]$ code. Its dual $\mathsf{RS}^\intercal$ is an $[n, n-t-1, t+2]$ code, which can be defined as follows:

$$\mathsf{RS}^\intercal = \{(d_1 p(1), d_2 p(2), \ldots, d_n p(n)) : p(X) \in \mathbb{Z}_q[X], \deg(p(X)) \leq n-t\},$$

where the coefficients d_i are given by $d_i = \prod_{\substack{1 \leq j \leq n \\ j \neq i}} \frac{1}{i-j}$.

3 Pre-Constructed Publicly Verifiable Secret Sharing

The definition of PVSS requires that given a secret f_0 and public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$ for a set of n parties $P_1, \ldots, P_n$, a dealer D proceeds as follows. First, the dealer creates shares $\{f_i\}_{i=1}^n$, and then encrypts each f_i under the corresponding pk_i, obtaining ciphertexts c_i for $i = 1, \ldots, n$. Finally, it publishes $\{c_i\}_{i=1}^n$ along with a proof π_{PVSS}, proving that the encrypted shares are well-formed. For instance, in the case of PVSS based on (n, t)-Shamir secret sharing, the proof guarantees that the encrypted shares are valid shares generated by the Shamir scheme. Specifically, any $t+1$ of n decrypted shares can be used to reconstruct the secret f_0, while any t decrypted shares reveal no information about the secret. As can be seen, the dealer in a PVSS scheme does not need to publish a commitment to (or an encryption of) the secret f_0. This is why in practical PVSS schemes [3, 7, 8, 10], the dealer does not publish a commitment

to f_0. Consequently, in current cryptographic protocols that use practical PVSS schemes [3,7,8,10] as a subroutine, if the main secret f_0 is needed, either at least $t+1$ shareholders collaborate to reconstruct it, or the dealer has to publish at least $O(t)$ field elements, and each party has to do $O(n)$ computation to compute f_0. For instance, in the SCRAPE randomness generation protocol [7], the dealer commits to all n shares $\{f_i\}_{i=1}^n$ separately and later opens them to verify and reconstruct f_0. This adds $O(n\lambda)$ bits communication and $O(n)$ computation cost to this final protocol, where λ is the security parameter and n is the number of parties. In the case of ALBATROSS [8], an improved version of SCRAPE, the dealer opens the secret polynomial itself, which again adds $O(t\lambda)$ bits communication and $O(n)$ computation costs to the final protocol, where $t \approx n/2$ is the threshold parameter. Given the revealed polynomial, a verifier needs to evaluate a degree-t polynomial n times and recompute n exponentiations to check if the commitments are valid.

Above all, as shown in [22, Sections 5 and 6], some applications of PVSS schemes, such as universally verifiable e-voting and threshold binding ElGamal, require the commitment (i.e., $C_0 = g^{f_0}$) or encryption of the main secret f_0 to be published by the dealer. Obviously, this feature exceeds the requirements of a standard PVSS scheme and is absent in [3,7,8,10]. The only exception is the less efficient PVSS scheme proposed by Schoenmakers [22] and its paring-based variant [19]. In Schoenmakers' scheme, the dealer publishes $C_0 = g^{f_0}$ in the proof π_{PVSS}, as a commitment to the first coefficient of the underlying secret polynomial. One may note that is occurred due to the following reasons. First, in Shamir secret sharing, the secret is typically embedded in $f(0) = f_0$. Consequently, in the Feldman VSS where the dealer publishes commitments to the coefficient of the secret polynomial, $C_0 = g^{f(0)} = g^{f_0}$ serves as the commitment to the first coefficient. However, if the secret would be embedded differently in Shamir secret sharing, for instance by setting $f(-1) = f_0$, Feldman VSS would still work, but the commitment to the main secret, $g^{f(-1)}$, would not be published. One might argue that given $g^{f(i)}$ for $i = 1, \ldots, n$, it is possible to publicly compute $g^{f(j)}$ at an arbitrary point j, such as the index of main secret, using Lagrange interpolation in the exponent. However, this would require $O(n)$ exponentiations. More importantly, in packed Shamir secret sharing and packed PVSS schemes [8], where the l secrets $f_0, f_{-1}, \ldots, f_{-(l-1)}$ are embedded to different points, e.g., $f(0), f(-1), \ldots, f(-(l-1)))$, the secret is not solely $f(0)$. Thus, commitments to the secrets would not necessarily be published in a standard PVSS scheme.

In applications where the dealer needs to publish the encryption of $f_0 = f(0)$ under a specific public key pk_0, such as threshold binding ElGamal [22, Section 6.1], in addition to proof π_{PVSS}, the dealer publishes $y_{f_0} = \mathsf{pk}_0^{f_0}$ along with a DLEQ proof π_{DLEQ} (using Fig. 1) to show that $y_{f_0} = \mathsf{pk}_0^{f_0} \ \wedge \ C_0 = g^{f_0}$. This again adds extra communication and computation costs to the final protocol.

To address the concerns outlined above and develop more efficient, versatile, and broadly applicable PVSS schemes, we introduce Pre-Constructed PVSS (PPVSS) and present practical constructions. We start by defining the syntax for

PPVSS schemes, adapting it from the standard PVSS syntax with some extensions. Next, we propose a general strategy to efficiently convert a Shamir-based PVSS scheme into a PPVSS protocol.

3.1 Definitions

Next, we introduce our definition for PPVSS schemes, which can be viewed as a natural extension of the standard PVSS definitions adapted from [3,7,8,22].

Definition 7. *A Pre-constructed PVSS (PPVSS) protocol consists of four algorithms of (Initial, Share, Verify, Reconstruct) as follows:*

1. $\mathsf{Initial}(1^\lambda) \rightarrow (\{\mathsf{pk}_i, \mathsf{sk}_i\}_{i=1}^n, h_0$ *or* $(\mathsf{pk}_0, \mathsf{sk}_0))$*: Given* 1^λ*, in this phase, each party* $\{P_i\}_{i=1}^n$ *registers public key* pk_i *and withholds the corresponding secret key* sk_i*. The parties and dealer also agree on a commitment key* h_0 *or a specific public key* pk_0 *with the secret key* sk_0*, which will be used in the sharing phase while committing to or encrypting the main secret.*
2. $\mathsf{Share}(n, t, f_0, \{\mathsf{pk}_i\}_{i=1}^n, h_0$ *or* $\mathsf{pk}_0) \rightarrow (\{y_i\}_{i=0}^n, \pi_{PPVSS})$*: Given secret* f_0*, it first secret shares* f_0 *and obtains the shares* $\{f_i\}_{i=1}^n$*. Next, it encrypts the share* f_i *under* pk_i *and obtains ciphertexts* $\{y_i\}_{i=1}^n$*. It also commits to (resp. encrypts)* f_0 *using* h_0 *(resp.* pk_0*) and obtains* y_0*. Then, it generates a publicly verifiable NIZK proof* π_{PPVSS} *that* y_0 *is indeed commitment to (or encryption of) main secret* f_0 *under* h_0 *(resp.* pk_0*), and* $\{y_i\}_{i=1}^n$ *are encryptions of a valid sharing of* f_0*. Finally, returns* $(y_0, y_1, \ldots, y_n, \pi_{PPVSS})$*.*
3. $\mathsf{Verify}(n, t, \{\mathsf{pk}_i, y_i\}_{i=0}^n, \pi_{PPVSS}) \rightarrow$ `true/false`*: In this phase, anyone (not necessarily a participant in the protocol) can verify non-interactively the values* $\{y_i\}_{i=0}^n$ *using* $(\{\mathsf{pk}_i\}_{i=0}^n, \pi_{PPVSS})$*, and output either* `true/false`*.*
4. Reconstruct*: This phase can be done in two ways:*
   (a) *(Optimistic)* $\mathsf{Reconstruct}^{\mathsf{opt}}((h_0, y_0, f_0, r_0)$ *or* $(\mathsf{pk}_0, y_0, \mathsf{sk}_0)) \rightarrow \{f_0, \mathsf{false}\}$*:*
      - *Given the commitment key* h_0*, the commitment* y_0 *and the opening values* (f_0, r_0)*, a verifier checks if* (f_0, r_0) *are valid opening for* y_0*. If so, the verifier returns* f_0*; otherwise it returns* `false`*.*
      - *Given the public key* pk_0*, the ciphertext* y_0 *and the secret key* sk_0*, the algorithm decrypts* y_0 *and returns* f_0*;*

   (b) *(Pessimistic)* $\mathsf{Reconstruct}^{pes}(\{y_i, \mathsf{sk}_i\}_{i \in Q, |Q|=t+1}) \rightarrow \{f_0, \mathsf{false}\}$*: Given any* $t+1$ *of the encrypted shares and secret keys, e.g.,* $\{y_i, \mathsf{sk}_i\}_{i=1}^{t+1}$*, it returns either the main secret* f_0*, or* `false`*. This can be done in two phases: decryption of the shares and share pooling. In the first step, every party* $P_i \in Q$ *decrypts the share* f_i *from the ciphertext* y_i *by using its secret key* sk_i*, and publishes* f_i *together with a NIZK proof* π_i^{Dec} *that this value is indeed a correct decryption of* y_i*. In the share pooling phase, a verifier* V *(not necessarily from the participants) first checks whether the proofs* $\{\pi_i^{Dec}\}_{i=1}^{t+1}$ *are correct. If the check passes for less than* t *parties in* Q *then* V *returns* false*; otherwise* V *applies a reconstruction procedure to the set of valid shares, i.e.,* $\{f_i\}_{i=1}^{t+1}$*, and calculates and returns* f_0*.*

Similar to a PVSS scheme [7,22], a PPVSS scheme needs to satisfy the following security guarantees.

- **Correctness**: If the dealer and parties follow the protocol, then the Verify algorithm will return $\mathtt{true}$ and the $\mathsf{Reconstruct}$ by both approaches will return f_0. More formally, for any integers $n > 1$ and $t < n$ a PPVSS is called correct, if for $(\{\mathsf{pk}_i, \mathsf{sk}_i\}_{i=1}^n, h_0 \text{ or } (\mathsf{pk}_0, \mathsf{sk}_0)) \leftarrow \mathsf{Initial}(1^\lambda)$, we have:

$$\Pr\left[\begin{array}{l}(\{y_i\}_{i=0}^n, \pi_{PPVSS}) \leftarrow \mathsf{Share}(n, t, f_0, \{\mathsf{pk}_i\}_{i=1}^n, h_0 \text{ or } \mathsf{pk}_0) : \\ \mathtt{true} \leftarrow \mathsf{Verify}(n, t, \{\mathsf{pk}_i, y_i\}_{i=0}^n, \pi_{PPVSS})\end{array}\right] = 1,$$

$$\Pr\left[\begin{array}{ll}(\{y_i\}_{i=0}^n, \pi_{PPVSS}) \leftarrow \mathsf{Share}(n, t, f_0, \{\mathsf{pk}_i\}_{i=1}^n, h_0 \text{ or } \mathsf{pk}_0), & \\ f_0' \leftarrow \mathsf{Reconstruct}^{opt}((h_0, y_0, f_0, r_0) \text{ or } (\mathsf{pk}_0, y_0, \mathsf{sk}_0)) & \vee \\ f_0' \leftarrow \mathsf{Reconstruct}^{pes}(\{y_i, \mathsf{sk}_i\}_{i \in Q, |Q|=t+1}) : f_0' = f_0 & \end{array}\right] = 1 \ .$$

- **Verifiability**: If Verify algorithm returns true, then with high probability the values $\{y_i\}_{i=1}^n$ are encryptions of a valid sharing of some secret, and y_0 is a commitment (or a ciphertext under pk_0) to the same secret. Furthermore if the check in the (pessimistic) $\mathsf{Reconstruct}^{pes}$ phase passes then the communicated values f_i are indeed the shares of the secret distributed by the dealer. Alternatively, if the check in the (optimistic) $\mathsf{Reconstruct}^{opt}$ phase passes then the input value f_0 is the main secret, shared in the sharing phase. More formally, given λ, for any integers $n \geq 2t + 1$ and $t \geq 0$, a PVSS is called verifiable if for any PPT adversaries $\mathcal{A}$, we have:

$$\Pr\left[\begin{array}{l}(\{\mathsf{pk}_i, \mathsf{sk}_i\}_{i=1}^n, h_0 \text{ or } (\mathsf{pk}_0, \mathsf{sk}_0)) \leftarrow \mathsf{Initial}(1^\lambda), \\ (\{y_i\}_{i=0}^n, \pi_{PPVSS}) \leftarrow \mathcal{A}(n, t, \{\mathsf{pk}_i\}_{i=1}^n, h_0 \text{ or } \mathsf{pk}_0), \\ \mathtt{true} \leftarrow \mathsf{Verify}(n, t, \{\mathsf{pk}_i, y_i\}_{i=0}^n, \pi_{PPVSS}) : \\ \exists s, \forall Q \in [n], s \leftarrow \mathsf{Reconstruct}^{pes}(\{y_i, \mathsf{sk}_i\}_{i \in Q, |Q| \geq t+1}), \\ \vee \ \ s \leftarrow \mathsf{Reconstruct}^{opt}((h_0, y_0, s, r_0) \text{ or } (\mathsf{pk}_0, y_0, \mathsf{sk}_0))\end{array}\right] \geq 1 - \mathsf{negl}(\lambda) \ ,$$

  where Q is the set of honest parties.

- **IND1-Secrecy (Indistinguishability of Secrets)**: Prior to the reconstruction phase, the public information together with the secret keys sk_i of any set of at most t parties, excluding sk_0, gives no information about the secret f_0. This is formalized by the following indistinguishability game adapted from [7,19]. We say that for any integers $n > 1$ and $t < n$, the PVSS scheme satisfies *IND1-Secrecy* if for any polynomial-time adversary $\mathcal{A}$ corrupting at most t parties, excluding the owner of sk_0, $\mathcal{A}$ has negligible advantage in the following game played against a challenger.
  1. The challenger runs $\mathsf{Initial}(1^\lambda)$ of the PPVSS and obtains $(\{\mathsf{pk}_i, \mathsf{sk}_i\}_{i=1}^n, h_0 \text{ or } (\mathsf{pk}_0, \mathsf{sk}_0))$ and sends all public information along with the secret information of all corrupted parties to the $\mathcal{A}$.
  2. The challenger chooses secrets s_0 and s_1 at random in the space of secrets. Furthermore, it chooses $b \in \{0, 1\}$ uniformly at random and runs

$\mathsf{Share}(n, t, s_0, \{\mathsf{pk}_i\}_{i=1}^n, h_0 \text{ or } \mathsf{pk}_0)$ algorithm of the PPVSS scheme with s_0 as the secret and obtains $(\{y_i\}_{i=0}^n, \pi_{PPVSS})$. It then sends $\mathcal{A}$ all public information generated in that phase, together with s_b.

3. $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$.

The advantage of $\mathcal{A}$ is defined as $\left|\Pr[b = b'] - \frac{1}{2}\right|$.

3.2 Building a PPVSS from a Shamir-based PVSS

From the definition of PVSS schemes presented in Def. 7, we observe that a PPVSS extends the functionality of PVSS schemes by committing to the main secret and providing a unique and efficient reconstruction approach. These features make a PPVSS scheme more useful and versatile than standard PVSS schemes. Over the past decade, a wide range of practical PVSS schemes have been proposed, achieving significant performance [3, 7–9, 17, 18, 20].

In this section, we introduce a general strategy that enables the efficient transformation of a Shamir-based PVSS scheme into an alternative PPVSS scheme with minimal computational and communication overhead. This strategy provides a generic and efficient approach to constructing a PPVSS scheme from a Shamir-based PVSS scheme. Our key observation is that in (n, t)-Shamir-based secret sharing protocols, both the main secret f_0 and the individual shares $\{f_i\}_{i=1}^n$ are distinct evaluations of a unique degree-t polynomial $f(x)$. In a publicly verifiable secret-sharing scheme, given individual public keys $\{\mathsf{pk}_i\}_{i=1}^n$, the dealer proves, in a publicly verifiable manner, that each individual share f_i encrypted under public key pk_i is an evaluation of a unique degree-t polynomial $f(x)$ at point i. Building on this, we observe that in a PVSS scheme with public keys $\{\mathsf{pk}_i\}_{i=1}^n$, if a dealer can convince a public verifier that $y_i = \mathsf{Enc}(f(i), \mathsf{pk}_i)$ for $i = 1, \ldots, n$, then the dealer can be slightly modified to prove the same statement for $i = 0, 1, \ldots, n$ as well. Here, pk_0 can represent the public key of a specific party in the target application, or it can be a public key whose secret key is unknown to any party. It is worth mentioning that any public key encryption scheme can act as a computationally hiding and perfectly binding commitment scheme if the committer (or encryptor) does not know the secret key. Thus, pk_0 can be set either as a specific party's public key in the target application or as the commitment key in a commitment scheme.

Applying this subtle modification to the Share algorithm of the underlying PVSS scheme yields a modified Share' algorithm for a new PPVSS scheme, which returns $n+1$ ciphertexts (or n ciphertexts and 1 commitment) along with a proof π_{PPVSS} to verify the correctness of the new sharing phase. Consequently, a verifier in the resulting PPVSS scheme would run the verification algorithm of the underlying PVSS scheme using $n+1$ ciphertexts and $n+1$ public keys (or n public keys and one commitment key). We demonstrate that this subtle modification has minimal impact on the efficiency of the underlying PVSS scheme while significantly enhancing its functionality. Furthermore, it enables a new approach to secret reconstruction in the resulting PPVSS scheme, which, in practice, can be considerably more efficient than current methods.

Initial′: In addition to running the Initial algorithm of the underlying PVSS scheme, parties agree on a commitment key pk_0, or a public key pk_0 with its associated secret key sk_0 which is only known to a specific party.

Share′: Given (n, t), a secret f_0, $\{\mathsf{pk}_i\}_{i=1}^n$, and the commitment or public key pk_0, the dealer runs the Share algorithm of the underlying PVSS scheme for $i = 0, 1, \ldots, n$ and obtains π_{PPVSS}. Specifically, the dealer proceeds as follows:

1. Samples a uniformly random polynomial $f(x)$ of degree t with $f(0) = f_0$ and coefficients in $\mathbb{Z}_q$. Then, for $i = 0, 1, \ldots, n$, sets $f_i = f(i)$.
2. Using $\{\mathsf{pk}_i, i, f_i\}_{i=0}^n$, computes the encrypted shares $\{y_i\}_{i=0}^n$, where y_0 is the encryption/commitment of f_0 under pk_0.
3. Runs the proof generation procedure of the underlying PVSS scheme with inputs $(\{\mathsf{pk}_i, y_i\}_{i=0}^n, f(x))$ and obtains the proof π_{PPVSS}.
4. Finally, broadcasts $(y_0, y_1, \ldots, y_n, \pi_{PPVSS})$.

Verify′: Given (n, t), $\{\mathsf{pk}_i, y_i\}_{i=0}^n$, and π_{PPVSS}, a verifier runs the Verify algorithm of Λ for $i = 0, 1, \ldots, n$ and outputs either `true` or `false`.

Reconstruct′: This phase can proceed in one of two approaches:

1. **Approach 1 (Optimistic Case with Commitment):** The dealer publishes the opening information for y_0, such as f_0 (and any associated randomness r_0 if needed). Then, given pk_0 and y_0, parties verify the validity of the revealed value f_0 (and associated randomness r_0 if needed) and return either f_0 (if the opening was valid) or false.
2. **Approach 2 (Pessimistic Case):** Given any $t+1$ of the encrypted shares and secret keys, e.g., $\{y_i, \mathsf{sk}_i\}_{i=1}^{t+1}$, parties run the Reconstruct algorithm of the underlying PVSS scheme and return either f_0 or false.

Fig. 2. A general construction for building a Shamir-based PPVSS scheme $\Lambda' = (\mathsf{Initial}', \mathsf{Share}', \mathsf{Verify}', \mathsf{Reconstruct}')$ from a Shamir-based PVSS scheme Λ.

In Fig. 2, we present a general construction for building a PPVSS scheme $\Lambda' = (\mathsf{Initial}', \mathsf{Share}', \mathsf{Verify}', \mathsf{Reconstruct}')$ using a Shamir-based PVSS scheme $\Lambda = (\mathsf{Initial}, \mathsf{Share}, \mathsf{Verify}, \mathsf{Reconstruct})$.

Efficiency of Resulting PPVSS Scheme. The efficiency of the new protocol Λ' closely aligns with that of the original protocol Λ, with only a marginal increase. Specifically, the efficiency of the new algorithms is approximately $1/n$ times more than the original ones, implying that running the algorithms of the PPVSS scheme Λ' is comparable to executing the original algorithms in the PVSS scheme for $n + 1$ parties instead of n parties. The new PPVSS scheme Λ' achieves enhanced functionality and efficiency with minimal cost.

We will discuss the security of resulting PPVSS scheme, later in the context of particular instantiations and the proposed protocols.

4 On Practical PPVSS Schemes

The strength of the scheme proposed in Fig. 2 lies in its generality and efficiency, as it only requires a secure PVSS scheme Λ. Over the past decades, various PVSS

schemes with different trade-offs have been proposed [3, 7, 8, 10, 17, 19, 22]. We compare the efficiency of well-known PVSS schemes in App. A.1.

In this section, we instantiate the general construction from Fig. 2 using two efficient PVSS schemes [3, 7] in the RO and plain models and construct two novel and practical PPVSS schemes. The proposed PPVSS schemes exhibit various trade-offs in terms of efficiency and security and the second construction uses bilinear pairing groups. We first highlight two less efficient PVSS schemes from the literature that can act as a PPVSS scheme, which in the dealer also publishes commitments to the (pre-constructed) secret.

4.1 Schoenmakers' PVSS Scheme is a PPVSS Scheme as Well

Among current constructions for PVSS schemes, one notable protocol was introduced by Schoenmakers [22]. We refer to Appendix A.2 for an overview of his construction. His proposed PVSS scheme is constructed in the RO model and is distinguished by its versatility, being applicable in many protocols. Specifically, this PVSS scheme can be used for random beacons, universally verifiable e-voting, binding ElGamal, revocable cash, and many other applications [22]. This versatility is primarily because, in Schoenmakers' construction, as part of the proof, the dealer uses Feldman VSS and publishes commitments to the coefficients of the secret polynomial $f(x)$, which includes a commitment to the first coefficient. The commitment to the first coefficient can also be considered a commitment to the main secret. This additional property is precisely what a type of PPVSS schemes needs to guarantee, compared to standard PVSS schemes. In [19], Heidarvand and Villar proposed a pairing-based variant of Schoenmakers' PVSS scheme, which removes the need for a random oracle and similarly publishes a commitment to the main secret. Thus, Schoenmakers' scheme [22] and its pairing-based variant proposed in [19] can also be considered as (special) PPVSS schemes, where in both cases, the dealer publishes a commitment to the pre-constructed secret as well. Despite their versatility, Schoenmakers' PPVSS scheme [22] and its pairing-based variant [19] are significantly less efficient compared to more recent PVSS schemes. Notably, their verification costs are dominated by $O(n^2)$ exponentiations.

Since the introduction of Schoenmakers' (P)PVSS [22], and its pairing-based variant [19], several follow-up and more efficient PVSS schemes have been developed [3, 7, 8, 10, 17], reducing the verification complexity to $O(n)$. Although all the recent PVSS schemes are considerably more efficient compared to the PPVSS schemes from [19, 22], they all are limited in terms of applicability and are mostly used in a restricted number of applications (e.g., distributed randomness generation). Consequently, many protocols based on (P)PVSS, such as universally verifiable e-voting and threshold binding ElGamal [22], still need to rely on Schoenmakers' construction [22] or its pairing-based variant [19], which both are too slow, rendering them impractical for large-scale applications. In PPVSS schemes build with our proposed construction, with minimal computational overhead, the dealer can publish either or both the encryption and/or commitment of the main secret f_0, providing greater flexibility.

Initial′: As in the Initial phase of the original PVSS scheme [3,8], given the generator g for $\mathbb{G}$, for $i = 1, \ldots, n$, each party P_i register their public key $\mathsf{pk}_i = g^{s_i}$, and store the secret key s_i. The parties also agree on $\mathsf{pk}_0 \neq g$ as the commitment key, or a target public key. In case of former, pk_0 is sampled randomly from $\mathbb{G}$, and in case of later, it is registered by the target person P_0 as $\mathsf{pk}_0 = g^{s_0}$, where $s_0 \in \mathbb{Z}_q$ is the secret key of the target person.

Share′: Given the parameters (n, t), a secret f_0, the public keys $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$, and the commitment or public key pk_0, the dealer acts as follows:

1. Samples a uniformly random polynomial $f(x)$ of degree t with $f(0) = f_0$ and coefficients in $\mathbb{Z}_q$. Then, for $i = 0, 1, \ldots, n$ sets $f_i = f(i)$.
2. Sets the encrypted shares $y_0 = \mathsf{pk}_0^{f_0}, y_1 = \mathsf{pk}_1^{f_1}, \ldots, y_n = \mathsf{pk}_n^{f_n}$. Note that, depending on the target application, in practice, the secret key of pk_0 might be unknown, or it can be known for a specific party.
3. Based on the proof scheme π_{PDL} (presented in [3]), samples a new random degree-t polynomial $r(x)$ with coefficients in $\mathbb{Z}_q$ and sets $c_i = \mathsf{pk}_i^{r(i)}$ for $i = 0, 1, \ldots, n$.
4. Given the random oracle $\mathcal{H}$, compute $d = \mathcal{H}(y_0, y_1, \ldots, y_n, c_0, c_1, \ldots, c_n)$ and $z(x) = r(x) + df(x)$.
5. Finally, broadcasts $(y_0, y_1, \ldots, y_n, \pi_{PPVSS} := (d, z(x)))$.

Verify′: Given (n, t), $(\mathsf{pk}_0, \mathsf{pk}_1, \ldots, \mathsf{pk}_n, y_0, y_1, \ldots, y_n)$, and $\pi_{PPVSS} = (d, z(x))$, a verifier first checks if $z(x)$ is a degree-t polynomial. If so, given the random oracle $\mathcal{H}$, it checks the following equation and outputs `true/false`

$$d == \mathcal{H}(y_0, y_1, \ldots, y_n, \frac{\mathsf{pk}_0^{z(0)}}{y_0^d}, \frac{\mathsf{pk}_1^{z(1)}}{y_1^d}, \ldots, \frac{\mathsf{pk}_n^{z(n)}}{y_n^d}).$$

Reconstruct′: This phase can proceed in one of two approaches:

1. Approach 1 (Optimistic Case): The dealer publishes f_0. Then, given (g, pk_0, y_0), a verifier (not necessarily from the parties) checks if $y_0 == \mathsf{pk}_0^{f_0}$ and returns either g^{f_0} (if the check passes) or false.
2. Approach 2 (Pessimistic Case): Parties run the Reconstruct algorithm of Π_S PVSS scheme and return either g^{f_0} or false. To this end,
   (a) They first use their secret key s_i, and obtain their share $F_i := g^{f_i}$ from y_i by computing $F_i = y_i^{1/s_i}$. Then, they publish F_i plus a NIZK proof that the value F_i is a correct decryption of y_i. To this end, the party P_i needs to prove knowledge of an s_i such that, $(\mathsf{pk}_i = g^{s_i}) \wedge (y_i = F_i^{s_i})$, which can be done by the NIZK proof scheme for the DLEQ relation (given in Fig. 1).
   (b) Then, given any $t + 1$ valid values of F_i, w.l.o.g. for $i = 1, \ldots, t + 1$, the secret $g^{f_0} = g^{f(0)}$, can be obtained by Lagrange interpolation,

$$\prod_{i=1}^{t+1} F_i^{\lambda_i} = \prod_{i=1}^{t+1} (g^{f_i})^{\lambda_i} = g^{\sum_{i=1}^{t+1} f_i \lambda_i} = g^{f(0)} = g^{f_0},$$

   where $\lambda_i = \prod_{j \neq i} \frac{j}{j-i}$ is a Lagrange coefficients.

Fig. 3. Λ_{RO}: an efficient RO-based PPVSS scheme based on the PVSS scheme Π_S.

4.2 A Practical PPVSS Scheme in the Random Oracle Model

In this section, we instantiate the general construction from Fig. 2 using the RO-based PVSS scheme proposed by Baghery [3], so-called Π_S, which can also be considered as a formally proven and non-packed version of the PVSS scheme initially proposed by Cascudo and David [8]. The resulting PPVSS scheme $\Lambda_{RO} = (\mathsf{Initial}', \mathsf{Share}', \mathsf{Verify}', \mathsf{Reconstruct}')$ is summarized in Fig. 3. The key differences between the original scheme and the resulting PPVSS scheme are highlighted in gray color. The resulting RO-based PPVSS scheme can be seen as the first alternative to the RO-based (P)PVSS scheme proposed by Schoenmakers [22], offering greater flexibility and more efficient verification, requiring $O(n)$ exponentiations instead of $O(n^2)$. This improvement allows us to revisit all the applications discussed in [22], and enhance their performance.

Theorem 1 (RO-Based PPVSS). *Under the Polynomial Discrete Logarithm (PDL) and Decisional Diffie-Hellman (DDH) assumptions, the PPVSS scheme (outlined in Fig. 3), is secure against an static adversary in the random oracle model. That is, (i) the protocol satisfies Correctness, (ii) it satisfies Verifiability under PDL assumption, and both the reconstruction algorithms described in* $\mathsf{Reconstruct}'$ *phase result in the secret* g^{f_0} *distributed by the dealer (for any qualified set of shareholders in case of approach 2), (iii) it satisfies IND1-Secrecy under DDH assumption, and any non-qualified set of shareholders is unable to distinguish between the shares of two secrets.*

Proof. In summary, the properties of *Correctness*, *Verifiability*, and *IND1-Secrecy* of the new PPVSS scheme inherit those of the underlying PVSS scheme [3]. However, for the sake of completeness, we provide a detailed discussion below.

Correctness. If the dealer and parties follow the protocol honestly, the verification and reconstruction processes yield the expected results. At the end of initialization phase, parties register public keys ($\{\mathsf{pk}_i := g^{s_i}\}_{i=1}^n, \mathsf{pk}_0$ or (pk_0, s_0)) and store the associated secret keys $\{s_i\}_{i=1}^n$ (or $\{s_i\}_{i=0}^n$) for the reconstruction phase. Given (n, t), the secret f_0, public keys $\{\mathsf{pk}_i\}_{i=1}^n$, and the commitment or public key pk_0, the dealer proceeds as follows: samples a polynomial $f(x)$ of degree t such that $f(0) = f_0$ and generates the encrypted shares $\{y_i = \mathsf{pk}_i^{f(i)}\}_{i=0}^n$, where y_0 is the encryption or commitment of f_0 under pk_0. Then, it samples $r(x)$ of degree t and sets $c_i = \mathsf{pk}_i^{r(i)}$ for $i = 0, 1, \ldots, n$. Next, it computes the challenge value $d = \mathcal{H}(y_0, y_1, \ldots, y_n, c_0, c_1, \ldots, c_n)$ and $z(x) = r(x) + df(x)$. Finally, it broadcasts $\pi_{PPVSS} := (d, z(x))$ and $y_0, y_1, \ldots, y_n$. By substituting these values in the verification equation, we observe that

$$d == \mathcal{H}(y_0, y_1, \ldots, y_n, \frac{\mathsf{pk}_0^{z(0)}}{y_0^d}, \frac{\mathsf{pk}_1^{z(1)}}{y_1^d}, \ldots, \frac{\mathsf{pk}_n^{z(n)}}{y_n^d})$$

$$= \mathcal{H}(y_0, y_1, \ldots, y_n, \frac{\mathsf{pk}_0^{r(0)+df(0)}}{\mathsf{pk}_0^{df(0)}}, \frac{\mathsf{pk}_1^{r(1)+df(1)}}{\mathsf{pk}_1^{df(1)}}, \ldots, \frac{\mathsf{pk}_n^{r(n)+df(n)}}{\mathsf{pk}_n^{df(n)}})$$

$$= \mathcal{H}(y_0, y_1, \ldots, y_n, \mathsf{pk}_0^{r(0)}, \mathsf{pk}_1^{r(1)}, \ldots, \mathsf{pk}_n^{r(n)}) = \mathcal{H}(y_0, y_1, \ldots, y_n, c_0, c_1, \ldots, c_n).$$

During the optimistic reconstruction phase, given $(g, \mathsf{pk}_0, y_0, f_0)$, clearly $y_0 = \mathsf{pk}_0^{f_0}$ and the algorithm will always return f_0. For the pessimistic case, given any $t + 1$ shares and their secret keys, e.g., $\{y_i, s_i\}_{i \in Q, |Q|=t+1}$, the parties run the $\mathsf{Reconstruct}$ algorithm of the original PVSS scheme and this also always yields f_0 if the shares will be generated honestly.

Verifiability. For the verifiability, we show that if Verify algorithm returns true, then the values $\{y_i\}_{i=1}^n$ are encryptions of a valid sharing of some secret, and y_0 is a commitment (or a ciphertext under pk_0) to the same secret. Furthermore if the check in the (pessimistic) $\mathsf{Reconstruct}^{pes}$ phase passes then the communicated values f_i are indeed the shares of the secret distributed by the dealer and the reconstruction with any $t + 1$ of valid shares will reach to the same secret

g^{f_0}. Alternatively, if the check in the (optimistic) $\mathsf{Reconstruct}^{opt}$ phase passes then the output value g^{f_0} is the main secret, shared in the sharing phase. This is achieved by showing that the NIZK argument deployed in the new PPVSS scheme satisfies special soundness under PDL problem for the relation $y_i = \mathsf{pk}_i^{f(i)}$ for $i = 0, 1, 2, \ldots, n$, where $f(x)$ is a (at most) degree-t witness polynomial. In other words, akin to the case of original PVSS scheme [3], we show that given two acceptable transcripts of the (interactive) protocol, any $t+1$ honest shareholders can use theirs secret keys and extract a unique (at most) degree-t polynomial from the dealer, which is a solution for the PDL problem. This implies that both the reconstitution approaches will result in a unique secret g^{f_0}. Let, $(c_i, d, z(x))$ and $(c_i, d', z'(x))$ be two acceptable transcripts of the (interactive) protocol, for $i = 0, 1, \ldots, n$. From the verification equation, we know that

$$\mathsf{pk}_i^{z(i)} = c_i(y_i)^d \quad , \quad \mathsf{pk}_i^{z'(i)} = c_i(y_i)^{d'} \quad \text{for} \;\; i = 0, 1, , \ldots, n \; .$$

Note that, as pointed in [3], the verification equation can be written in the above form as well. This implies that,

$$\mathsf{pk}_i^{z(i)-z'(i)} = y_i^{d-d'} \;\Rightarrow\; y_i = \mathsf{pk}_i^{\frac{z(i)-z'(i)}{d-d'}} \qquad \text{for } i = 0, 1, \ldots, n \; . \tag{2}$$

Then, if all $n \geq 2t + 1$ of the checks in the verification pass successfully, given the (pessimistic) reconstruction protocol detailed in Fig. 3, any set of $t + 1$ honest parties can decrypt $\{y_i\}_{i \in Q, |Q|=t+1, i \neq 0}$, as $F_i := y_i^{1/s_i}$, and rewrite the last equation as below,

$$g^{z(i)-z'(i)} = F_i^{d-d'} \;\Rightarrow\; F_i = g^{\frac{z(i)-z'(i)}{d-d'}} \qquad \text{for } i \in Q, \; |Q| = t + 1, i \neq 0.$$

Note that at this stage parties need to give NIZK proof for DLEQ relation for their correct decryption, therefore malicious parties will be aborted. Now, since $z(x)$ is a degree-t polynomial, and since from $f_i := \frac{z(i)-z'(i)}{d-d'}$ for $i \in Q, |Q| = t+1, i \neq 0$, we obtain $t+1$ *distinct* evaluations of a (witness) degree-t polynomial $\frac{z(x)-z'(x)}{d-d'}$, therefore an extractor can use $\{f_i\}_{i \in Q, |Q|=t+1, i \neq 0}$ and reconstruct (extract) a *unique* (at most) degree-t polynomial $f(x)$, which is a solution for the PDL problem. This implies that, in the secret reconstruction with pessimistic approach, any set of $t + 1$ honest parties, can use their individual (decrypted) shares $F_i := y_i^{1/s_i}$, employ Lagrange interpolation (as in Fig. 3), and evaluate a unique degree-t polynomial $f(x)$ in the *exponent* at any point. By evaluating $g^{f(x)}$ at point 0, they can obtain a unique secret value $g^{f(0)}$.

Note that in the case of $i = 0$, from equation (2), we can write

$$\mathsf{pk}_0^{z(0)-z'(0)} = y_0^{d-d'} \;\Rightarrow\; y_0 = \mathsf{pk}_0^{\frac{z(0)-z'(0)}{d-d'}}$$

which is a discrete logarithm solution. This implies that $\frac{z(0)-z'(0)}{d-d'}$ is the constant term of the same (up to) degree-t witness polynomial $f(x)$, extracted using the pessimistic reconstruction approach. Consequently, we conclude $y_0 = \mathsf{pk}_0^{f(0)} =$

$\mathsf{pk}_0^{f_0}$. Thus, in secret reconstruction using the optimistic approach, given the opening information f_0, anyone can verify its validity and compute the unique secret g^{f_0}. In the scenario where pk_0 is the public key of a specific party, given the associated secret key s_0, one can compute the unique secret $F_0 := g^{f_0}$ by calculating y_0^{1/s_0}.

IND1-Secrecy. We show that, if there exists an adversary $\mathcal{A}$ which can break the IND1-secrecy property of the PPVSS scheme from Fig. 3, then there exists an adversary $\mathcal{B}$ which can use $\mathcal{A}$ as a subroutine to break the Decisional Diffie-Hellman (DDH) assumption with the same advantage.

Without loss of generality, we assume $\mathcal{A}$ corrupts the $t-1$ first parties (excluding the owner of s_0). Let $(h, h^\alpha, h^\beta, h^\gamma)$ be an instance of the DDH problem. We assume the values of α or β are nonzero. Now $\mathcal{B}$, using $\mathcal{A}$, can simulate an IND1 game as follows:

1. The challenger sets $g = h^\alpha$ and runs the initial phase of the protocol. For $t \leq i \leq n$ and $i = 0$, $\mathcal{B}$ selects uniformly random values $u_i \leftarrow \mathbb{Z}_q$ (implicitly defining $s_i = u_i/\alpha$) and sends the values $\mathsf{pk}_i = h^{u_i}$ to $\mathcal{A}$.
2. For $1 \leq i \leq t-1$, $\mathcal{A}$ chooses uniformly random values $s_i \leftarrow \mathbb{Z}_q$ and sets $\mathsf{pk}_i = g^{s_i}$ and sends this to the challenger.
3. For $1 \leq i \leq t-1$, the challenger chooses uniformly random values $f_i \leftarrow \mathbb{Z}_q$ and sets $v_i = h^{f_i}$ and $y_i = \mathsf{pk}_i^{f_i}$.
   For $t \leq i \leq n$, it generates the values $v_i = h^{p(i)}$, where $p(x)$ is the unique polynomial of degree at most t determined by $p(0) = \beta$ and $p(i) = f_i$ for $i = 1, \ldots, t-1$. Note that $\mathcal{B}$ does not know β, but it does know $h^\beta = h^{p(0)}$ and $h^{f_i} = h^{p(i)}$ for $1 \leq i \leq t-1$, so it can use Lagrange interpolation in the exponent to compute the adequate v_i. It also creates the values $y_i = v_i^{u_i}$ for $t \leq i \leq n$ and $y_0 = (h^\beta)^{u_0}$. From all the computed values $\{\mathsf{pk}_i, y_i\}_{i=0}^n$, and given the challenge value of the interactive proof scheme, the simulator can sample a random degree-t polynomial $z'(x)$ and set $c'_i := \mathsf{pk}_i^{z'(i)}/y_i^d$ for $i = 0, 1, \ldots, n$. Finally, it sends all this information together with the value h^γ (which plays the role of s_b in the IND1 game) to $\mathcal{A}$.
4. $\mathcal{A}$ makes a guess b'. If $b' = 0$, $\mathcal{B}$ guesses that $\gamma = \alpha \cdot \beta$. If $b' = 1$, $\mathcal{B}$ guesses that γ is a random element in $\mathbb{Z}_p$.

The information that $\mathcal{A}$ receives in step 3 is distributed exactly like a sharing of the value $g^\beta = h^{\alpha \cdot \beta}$ with the PPVSS. Consequently, $\gamma = \alpha \cdot \beta$ if and only if the value h^γ sent to $\mathcal{A}$ is the secret shared by the PPVSS. It is now easy to see that the guessing advantage of $\mathcal{B}$ is the same as the advantage of $\mathcal{A}$. □

4.3 An Efficient PPVSS Scheme in the Plain Model

The PPVSS scheme proposed in Sec. 4.2 is built in the random oracle model. In this section, we present an efficient PPVSS scheme based on pairings in the plain model. To this end, we instantiate the general construction from Fig. 2 using the pairing-based PVSS scheme proposed by Cascudo and David [7], that relies on Decisional Bilinear Square (DBS) assumption [19]. Similarly, the resulting

pairing-based PPVSS scheme can be seen as the first alternative to the paring-based PPVSS scheme proposed by Heidarvand and Villar [19], offering greater flexibility and more efficient verification, requiring $O(n)$ exponentiations and $O(n)$ pairings instead of $O(nt)$ exponentiations and $O(n)$ pairings.

As in the underlying PVSS scheme, in the new PPVSS scheme, the dealer initially uses pairings to prove the equality of the encrypted and committed shares, i.e., $\{y_i, c_i\}_{i=0}^{n}$. Subsequently, the verifier employs the dual-code trick to check that the committed shares $\{c_i\}_{i=0}^{n}$ are valid and properly formed. It is important to note that the equality proof provided by the dealer ensures that the correctness of the committed shares $\{c_i\}_{i=0}^{n}$ implies that the encrypted shares $\{y_i\}_{i=0}^{n}$ are also correct and well-formed.

Let $(q, \mathbb{G}, \mathbb{G}_T, e)$ be a description of a bilinear group and (g, h) be two independently chosen generators of $\mathbb{G}$. Let RS be the linear error correcting code equivalent to the (n, t)-threshold Shamir secret sharing scheme and RS^{T} be its dual code. The resulting PPVSS scheme $\Lambda_{Plain} = (\mathsf{Initial}', \mathsf{Share}', \mathsf{Verify}', \mathsf{Reconstruct}')$ is summarized in Fig. 4. As in Λ_{RO}, the key differences between the original scheme [19] and the new PPVSS scheme are highlighted.

As in the original and prior PVSS schemes in the plain model [7, 19], we present our PPVSS schemes over a symmetric bilinear group, also known as Type-I bilinear groups. However, as in original PVSS case, we note that our proposed PPVSS scheme can be adapted to asymmetric bilinear groups, where pairing-friendly curves enable more efficient pairing algorithms.

Security. The algorithms of new PPVSS scheme are similar to the underlying PVSS scheme [7], while the optimistic reconstruction is specific for the new PPVSS scheme. We have made adjustments in the security proof of underlying PVSS scheme to adapt for our protocol under the Decisional Bilinear Square (DBS) assumption. This proof leverages pairing-based verification, ensuring that any maliciously constructed encrypted shares $y_0, y_1, \ldots, y_n$ will pass verification with probability no higher than $1/q$, while $c_0, c_1, \ldots, c_n$ reveal no information about the secret $e(h, h)^{f_0}$ under the DBS assumption.

Theorem 2. *Under the DBS assumption, the PPVSS scheme Λ_{Plain} outlined in Fig. 4 satisfies IND1-secrecy against a static PPT adversary in the plain model.*

Proof. We show that, if an adversary $\mathcal{A}$ can break the IND1-secrecy property of PPVSS scheme Λ_{Plain}, then we can build another adversary $\mathcal{B}$ that can exploit $\mathcal{A}$ to break the Decisional Bilinear Square (DBS) assumption with equivalent advantage. Without loss of generality, we assume $\mathcal{A}$ corrupts the t first parties (excluding the owner of s_0).

Assume an instance $(g, g^{\alpha}, g^{\beta}, T)$ is given, where $\mathcal{B}$ must determine whether $T = e(g^{\alpha}, g^{\alpha})^{\beta}$ or $T = e(g^{\alpha}, g^{\alpha})^{\gamma}$ for a random $\gamma \leftarrow \mathbb{Z}_q$. Obviously if $\alpha = 0$ or $\beta = 0$ then the problem is trivial, so we assume these values are nonzero. Now $\mathcal{B}$, using $\mathcal{A}$ in an IND1 game as follows:

1. The challenger sets $h = g^{\alpha}$ and runs the Setup phase of PPVSS in plain model. For $t < i \leq n$ and $i = 0$, A_{DBS} selects uniformly random values

Initial′: Let $(q, \mathbb{G}, \mathbb{G}_T, e)$ be a description of a bilinear group and (g, h) be two independently chosen generators of $\mathbb{G}$. As in the Initial phase of the original PVSS scheme [7], given the generator h, for $i = 1, \ldots, n$, each party P_i register their public key $\mathsf{pk}_i = h^{s_i}$, and store the secret key s_i. The parties also agree on $\mathsf{pk}_0 \neq h$ as the commitment key, or a target public key. In case of former, pk_0 is sampled randomly from $\mathbb{G}$, and in case of later, it is registered by the target person P_0 as $\mathsf{pk}_0 = h^{s_0}$, where $s_0 \in \mathbb{Z}_q$ is the secret key of P_0.

Share′: Given the parameters (n, t), a secret f_0, the public keys $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$, and the commitment or public key pk_0, the dealer defines the secret $S = e(h, h)^{f_0}$ and acts as follows:

1. Samples a uniformly random polynomial $f(x)$ of degree t with $f(0) = f_0$ and coefficients in $\mathbb{Z}_q$. Then, for $i = 0, 1, \ldots, n$ sets $f_i = f(i)$.
2. Sets the encrypted shares $y_0 = \mathsf{pk}_0^{f_0}, y_1 = \mathsf{pk}_1^{f_1}, \ldots, y_n = \mathsf{pk}_n^{f_n}$. Note that, depending on the target application, in practice, the secret key of pk_0 might be unknown, or it can be known for a specific party.
3. Sets the commitments $c_i = g^{f_i}$ for $i = 0, 1, \ldots, n$.
4. Finally, broadcasts $\pi_{PPVSS} := (c_0, c_1, \ldots, c_n)$ and $(y_0, y_1, \ldots, y_n)$.

Verify′: Given (n, t), $(\mathsf{pk}_0, \mathsf{pk}_1, \ldots, \mathsf{pk}_n, y_0, y_1, \ldots, y_n)$, and $\pi_{PPVSS} = (c_0, c_1, \ldots, c_n)$, a verifier first checks if $e(y_i, g) = e(\mathsf{pk}_i, c_i)$, for $i = 0, 1, \ldots, n$. If so, it samples a random codeword $v^{\intercal} = (v_0^{\intercal}, v_1^{\intercal}, \ldots, v_n^{\intercal})$ of the dual code $\mathsf{RS}^{\intercal}$ corresponding to RS, the instance of Shamir's (n, t)-threshold secret sharing used by dealer, and checks if:

$$c_0^{v_0^{\intercal}} \prod_{i=1}^{n} c_i^{v_i^{\intercal}} == 1$$

and outputs either `true/false`.

Reconstruct′: This phase can proceed in one of two approaches:

1. Approach 1 (Optimistic Case): The dealer publishes f_0. Then, given (h, pk_0, y_0), a verifier (not necessarily from the parties) checks if $y_0 == \mathsf{pk}_0^{f_0}$ and returns either $S = e(h^{f_0}, h)$ or false.
2. Approach 2 (Pessimistic Case): Parties run the Reconstruct algorithm of the original PVSS scheme [7] and return either $S = e(h^{f_0}, h)$ or false. To this end,
   (a) Parties in a qualified set Q first use their secret key s_i, and obtain their share $F_i := h^{f_i}$ from y_i by computing $F_i = y_i^{1/s_i}$. Then, they publish F_i. Note that here the NIZK proof for the correctness of decryption is empty.
   (b) Given $\{\mathsf{pk}_i, c_i, F_i\}_{i \in Q}$, a verifier checks if $e(\mathsf{pk}_i, F_i) = e(y_i, h)$ for every $P_i \in Q$.
   (c) If the check succeeds, they reconstruct $h^{f_0} = h^{f(0)}$ by Lagrange interpolation as,
   $$\prod_{P_i \in Q} F_i^{\lambda_i} = \prod_{P_i \in Q} (h^{f_i})^{\lambda_i} = h^{\sum_{P_i \in Q} f_i \lambda_i} = h^{f(0)} = h^{f_0},$$
   where $\lambda_i = \prod_{j \neq i} \frac{j}{j-i}$ is a Lagrange coefficients.
   (d) They return $S = e(h^{f_0}, h)$ or false if fewer than $t + 1$ valid shares were collected.

Fig. 4. Λ_{Plain}: an efficient PPVSS scheme in the plain model.

$u_i \leftarrow \mathbb{Z}_q$ (these can be thought of implicitly defining s_i as $s_i = u_i/\alpha$) and sends the values $pk_i = g^{u_i}$ to $\mathcal{A}$.

2. For $1 \leq i \leq t$, $\mathcal{A}$ chooses uniformly random values $s_i \leftarrow \mathbb{Z}_q$ and sets $\mathsf{pk}_i = h^{s_i}$ and sends this to the challenger.
3. For $1 \leq i \leq t$, the challenger chooses uniformly random values $f_i \leftarrow \mathbb{Z}_q$ and sets $c_i = g^{f_i}$ and $y_i = \mathsf{pk}_i^{f_i}$.
4. For $t < i \leq n$ and $i = 0$, the challenger generates the values $c_i = g^{f(i)}$ where $f(X)$ is a unique (at most) degree-t polynomial, determined by $f(0) = e(h, h)^{\beta} = e(g^{\alpha}, g^{\alpha})^{\beta}$ and $f(i) = f_i$ for $i = 1, \ldots, t$. The challenger now sends T (which plays the role of s_b in the IND1 game) to $\mathcal{A}$.
5. At the end of sharing phase, the value T is sent as part of the game response. If $\mathcal{A}$'s guess $b' = 0$, then $\mathcal{B}$ guesses that $T = e(g^{\alpha}, g^{\alpha})^{\beta}$; if $b' = 1$, it guesses $T = e(g^{\alpha}, g^{\alpha})^{\gamma}$. This ensures that $\mathcal{B}$'s advantage mirrors that of $\mathcal{A}$.

This completes the proof. □

Theorem 3. *If the dealer does not construct values* (c_i, y_i) *of the right form in the sharing phase (i.e. either* $\log_g c_i \neq \log_{\mathsf{pk}_i} y_i$ *for some* i*, or* $\log_g c_i = \log_{\mathsf{pk}_i} y_i = f_i$ *for all* $0 \leq i \leq n$ *but the values* $\{f_i\}_{i=1}^n$ *do not constitute a valid sharing of* $f_0 \in \mathbb{Z}_q$ *with the* (n,t)*-threshold Shamir secret sharing scheme), then this is detected in the verification phase with probability at least* $1 - 1/q$*.*

Proof. For all $i = 0, \ldots, n$ the relation of $\log_g c_i = \log_{\mathsf{pk}_i} y_i$ is verified by checking that $e(y_i, g) = e(\mathsf{pk}_i, c_i)$. Note that if $a = \log_g c_i \neq \log_{\mathsf{pk}_i} y_i = b$ for some i, then $e(y_i, g) = e(\mathsf{pk}_i, g)^b \neq e(\mathsf{pk}_i, g)^a = e(\mathsf{pk}_i, c_i)$ and the check fails with probability 1. Now with probability ϵ, for every $0 \leq i \leq n$, there exists f_i with $c_i = g^{f_i}$ and $y_i = \mathsf{pk}_i^{f_i}$. Now the values f_i are a valid sharing of f_0 with the (n,t)-threshold Shamir secret sharing scheme if and only if the vector $\vec{F} = (f_0, f_1, \ldots, f_n) \in \mathsf{RS}$, where RS is the liner error correcting code equivalent to the (n,t)-threshold Shamir secret sharing. Suppose that $\vec{F} := (f_0, f_1, \ldots, f_n) \notin \mathsf{RS}$. Let, $\mathsf{RS}^\intercal$ denotes the dual code of RS. Then, since the codeword $\vec{V}^\intercal := (v_0^\intercal, v_1^\intercal, \ldots, v_n^\intercal) \in \mathsf{RS}^\intercal$ is sampled uniformly at random, then $\langle \vec{F}, \vec{V}^\intercal \rangle \neq 0$ except with probability $1/q$, where $\langle \vec{F}, \vec{V}^\intercal \rangle$ denotes the inner product of two vectors $\vec{F}$ and $\vec{V}^\intercal$ as $\langle \vec{F}, \vec{V}^\intercal \rangle = \sum_{i=0}^n f_i \cdot v_i^\intercal$. But then, from the verification equation, we will have

$$\prod_{i=0}^n c_i^{v_i^\intercal} = \prod_{i=0}^n g^{f_i \cdot v_i^\intercal} = g^{\langle \vec{F}, \vec{V}^\intercal \rangle} \neq 1.$$

Hence if the values $\{f_i\}_{i=1}^n$ are not a valid Shamir sharing of f_0, then the check fails with probability $1 - 1/q$. This completes the proof. □

Theorem 4. *In the reconstruction phase of new PPVSS scheme, if the dealer or parties communicate invalid values, this is detected by the verifier. Namely, i) if a party in* Q *communicates an erroneous decryption share* y_i *in the pessimistic reconstruction phase, this is detected by the verifier with probability* 1*. i) if a dealer publishes an invalid* f_0 *in the optimistic reconstruction phase, then this is catched with probability* 1*.*

Proof. For the pessimistic case we observe that, if $y_i = h^a$ with $a \neq f_i$ then $e(pk_i, y_i) = e(pk_i, h)^a \neq e(pk_i, h)^{s_i} = e(f_i, h)$. For the optimistic case, we note that given $(h, \mathsf{pk}_0, y_0, f_0)$, a verifier checks if $y_0 == \mathsf{pk}_0^{f_0}$, which can be seen as a perfectly binding commitment scheme and the check fails with probability 1 if $y_0 \neq \mathsf{pk}_0^{f_0}$. □

In App. B.3, we discuss the homomorphic property of new PPVSS schemes.

4.4 Empirical Performance of Λ_{RO} PPVSS Scheme and Comparison

We conducted a detailed analysis of the concrete efficiency of the proposed PPVSS schemes and compared them with relevant constructions from the literature. A summary of the results is presented in Table 1.

In addition, we evaluated the practical performance of the new RO-based PPVSS scheme through a prototype implementation in Rust and compared its

performance with the (P)PVSS scheme proposed by Schoenmakers [22] [1]. From the results in Table 1, we observe that the reconstruction phases of both new PPVSS schemes are as efficient as those of comparable schemes. Therefore, we focus primarily on the performance of the sharing and verification phases, as well as the communication size. Our experiments are done using the Ristretto Group under `Curve25519` and the hash function `Blake3` for the random oracle. The tests were run on MacBook Pro with an M4 Pro CPU with 24GB RAM. Both sharing and verification were limited to 8 threads to utilize the 8 performance cores available. All numbers are averaged over 100 iterations. The implementation results for several parameter sets are summarized in Table 2.

Table 2. Implementation results of Λ_{RO} PPVSS scheme and comparison with Schoenmakers' scheme [22]. n: Number of parties, t: Threshold value, $|\mathbb{Z}_q| = |\mathbb{G}| = 256$ bits.

(n, t)	Metrics	Schoenmakers [22]	Λ_{RO} PPVSS (Fig. 3)
(64, 31)	Sharing	1.29 ms	0.59 ms
	Verification	8.30 ms	0.51 ms
	Dealer's Broadcast	5 KB	3 KB
(128, 63)	Sharing	2.39 ms	1.12 ms
	Verification	31.60 ms	0.94 ms
	Dealer's Broadcast	10 KB	6 KB
(256, 127)	Sharing	4.87 ms	2.47 ms
	Verification	123.58 ms	2.10 ms
	Dealer's Broadcast	20 KB	12 KB
(512, 255)	Sharing	10.51 ms	6.21 ms
	Verification	482.70 ms	4.97 ms
	Dealer's Broadcast	40 KB	24 KB
(1024, 513)	Sharing	24.73 ms	17.96 ms
	Verification	1.91 s	13.68 ms
	Dealer's Broadcast	80 KB	48 KB
(2048, 1023)	Sharing	64.56 ms	59.16 ms
	Verification	7.72 s	42.97 ms
	Dealer's Broadcast	160 KB	96 KB

The implementation results also confirm that the new PPVSS protocol improves upon Schoenmakers' scheme [22] in all aspects and offers significantly faster verification. For $(n, t) = (512, 255)$, the new scheme achieves verification times that is approximately 97× faster in comparison to the Schoenmakers' scheme, and this gap increases for the large values of n.

5 A Practical Universally Verifiable E-Voting Protocol

In [22, Sections 5 and 6], Schoenmakers proposed a variety of applications based on his (P)PVSS scheme, including a universally verifiable e-voting protocol,

[1] Our source code for the implementations is publicly available at https://github.com/KULeuven-cosic/ppvss-evoting.

threshold software key escrow, and threshold binding ElGamal encryption. Our analysis shows that these applications are specifically suited for PPVSS schemes and crucially rely on the dealer's ability to commit to the main secret or encrypt it under a specific public key–features that are absent in standard, more efficient PVSS schemes [3,7,8,10]. Our proposed PPVSS schemes, presented in Section 4, directly address these limitations.

In Appendix B, we revisit the threshold software key escrow and threshold binding ElGamal encryption applications proposed by Schoenmakers [22] and present a more efficient variant of each. The threshold binding ElGamal encryption application relies on the dealer encrypting the main secret under a specific public key, a property readily achievable with our proposed PPVSS schemes. The threshold software key escrow application relies on the dealer publishing a commitment to the main secret.

As another important application of PPVSS schemes, in this section, we use our proposed RO-based PPVSS scheme (from Fig. 3) to revisit Schoenmakers' universally verifiable e-voting protocol and improve its overall performance. Notably, the new e-voting protocol requires $O(mn)$ exponentiations, rather than $O(mn^2)$, in the verification phase, where n and m denote the number of talliers and voters, respectively. This improvement makes our proposal considerably more efficient in practice, particularly for elections with large values of n. We begin by briefly outlining Schoenmakers' original e-voting protocol [22, Section 5] before presenting our revised variant.

5.1 Schoenmakers' Universally Verifiable E-Voting Protocol

Schoenmakers' proposed e-voting protocol follows the model for universally verifiable elections introduced by Benaloh et al. [6]. This model assumes the availability of a public bulletin board where all participants can post messages. The players in the scheme consist of a set of tallying authorities (talliers) $A_1, \ldots, A_n$, a set of voters $V_1, \ldots, V_m$ and a set of passive observers. These two sets need not be disjoint. As an instance, in small-scale elections like board-room elections, each player may be both a voter and a tallier. Schoenmakers' protocol leverages his proposed (P)PVSS scheme to achieve public verification while maintaining voter privacy. It consists of three main phases. In the first phase, voters use the (P)PVSS scheme to cast their encrypted ballots on the public bulletin board. Since voter anonymity is not a requirement in this scheme, double voting can be easily prevented by checking the votes and verifying the validity of ballots before accepting them. In the second phase, the bulletin board verifies the ballots and obtain the list of valid votes $\{V_i\}_{j=1}^{m}$. Note that, since all the proofs verified in this phase are publicly verifiable, therefore they can be verified by anyone. After a successful verification phase, in the third phase, the talliers use their private keys to collectively compute the final tally, by accumulating all valid ballots and decrypting the accumulated ballot. Fig. 5 summarizes the procedure of Schoenmakers' e-voting protocol.

Initialization Phase: In this phase, the initialization of the PPVSS scheme has been completed, and each tallier A_i has sampled a secret key $x_i \in \mathbb{Z}_q$ and registered public key $\mathsf{pk}_i = G^{x_i}$. They also agree on pk_0 as the commitment key.

Ballot Casting Phase: Given the public keys of tallies, to casts a vote $v \in \{0,1\}$, a voter acts as below:

1. Samples a random secret value $s \in \mathbb{Z}_q$.
2. Using the public keys of talliers $\{\mathsf{pk}_i\}_{i=0}^{n}$ and the secret value s, runs the distribution protocol for his PVSS (or our PPVSS) scheme and obtains the encrypted shares and the associated proof π_{PVSS} (or π_{PPVSS}).
3. Computes the value $U = G^{s+v}$ and using the protocol given in Sec. 2.1 constructs a well-formedness proof π_U, showing that indeed $v \in \{0,1\}$, without revealing any information on v. The proof π_U refers to the value of $C_0 = h^s$ (or $y_0 = \mathsf{pk}_0^s$) which is published as part of π_{PVSS} (or π_{PPVSS}) in the (P)PVSS (or PPVSS) distribution protocol, and shows that:

$$\log_G U = \log_h C_0 \vee \log_G U = 1 + \log_h C_0 \text{ (or } \log_G U = \log_{\mathsf{pk}_0} y_0 \vee \log_G U = 1 + \log_{\mathsf{pk}_0} y_0\text{)}.$$

   The ballot for voter V consists of the output of the (P)PVSS (or PPVSS) distribution protocol (which consists of encryption of shares and the proof π_{PVSS} (or π_{PPVSS})), the value U (which can be seen as an encryption of the vote), and proof π_U (which proves the well-formedness of U).

Ballot Verification Phase: Once voters submitted their ballots, the ballots are checked by the bulletin board. Note that both proofs π_{PVSS} (or π_{PPVSS}) and π_U are publicly verifiable and can be checked by anyone.

Tallying Phase: Suppose voters $\{V_j\}_{j=1}^{m}$, have all cast valid ballots. The tallying protocol consists of the following step:

1. Talliers first accumulate all the respective encrypted shares, under the same public keys, by computing the values $Y_i^\star$, where the index j ranges over all voters:

$$Y_i^\star = \prod_{j=1}^{m} Y_{ij} = y_i^{\sum_{j=1}^{m} p_j(i)} .$$

2. Next, each tallier A_i applies the reconstruction protocol of Schoenmakers' (P)PVSS (or our PPVSS) scheme to the value $Y_i^\star$, which will produce $S^\star = G^{\sum_{j=1}^{m} p_j(0)} = G^{\sum_{j=1}^{m} s_j}$, due to the homomorphic property of the (P)PVSS (or PPVSS) scheme.
3. Using $\{U_j\}_{j=1}^{m}$, talliers compute $U^\star = \prod_{j=1}^{m} U_j = G^{\sum_{j=1}^{m} s_j + v_j}$. Then, using $S^\star$ and $U^\star$ they compute $V^\star = \frac{U^\star}{S^\star} = G^{\sum_{j=1}^{m} v_j}$. Given $V^\star$, the tally $V = \sum_{j=1}^{m} v_j$, $0 \le V \le m$, can be computed efficiently.

Fig. 5. A universally verifiable e-voting protocol based on Schoenmakers' (P)PVSS [22] and our proposed PPVSS. The highlighted texts are particular for the new variant.

5.2 Schoenmakers' E-Voting Protocol Revisited

Leveraging the proposed RO-based PPVSS scheme from Fig. 3, next we revisit Schoenmakers' e-voting protocol and present a more efficient version. Given the new PPVSS scheme, the revision is straightforward. Essentially, in Schoenmakers' e-voting protocol (summarized in Fig. 5) we replace his (P)PVSS scheme with our proposed PPVSS scheme and adapt the rest of the protocol accordingly. With this replacement, using our PPVSS scheme, each voter will publish $y_0 = \mathsf{pk}_0^s$ as the commitment to their random secret s, which will serve the role of C_0 in Schoenmakers' voting protocol. Fig. 5 presents the resulting e-voting protocol and highlights its key difference with the original scheme.

5.3 Efficiency Comparisons and Benchmarks

Efficiency of Schoenmakers' E-voting Protocol. The efficiency of Schoenmakers' e-voting protocol can be summarized as follows. In the first phase, each voter

acts as the dealer in Schoenmakers' (P)PVSS scheme and distributes a random secret s. This requires each voter (i.e., the dealer) to compute $4.5n$ exponentiations and n evaluations of a degree-t polynomial. Using C_0 and U, each voter also runs the proof scheme from Sec. 2.1 and generates a proof π_U. This proof consists of $(4|\mathbb{G}|, 4|\mathbb{Z}_q|)$ elements and requires an additional 6 exponentiations in the group. In the second phase, the verifier needs to run the verifier of both Schoenmakers' (P)PVSS protocol and the NIZK proof scheme from Sec. 2.1 to check the validity of all ballots submitted by $\{V_j\}_{j=1}^m$. The computational cost of this step is approximately $m(nt + 4.5n)$ exponentiations, where m and n are the number of voters and talliers, respectively, and $t = \lfloor \frac{n-1}{2} \rfloor$ is the threshold value. In the third phase, each tallier A_i needs to compute $Y_i^\star$, run the reconstruction of Schoenmakers' (P)PVSS scheme, check the openings of other talliers and compute the final tally by calculating the discrete logarithm of $V^\star = G^{\sum_{j=1}^m}$. The total cost for this step is approximately $5t$ exponentiations and $n + t$ multiplications in the underplaying group.

Efficiency of New E-voting Protocol. In the first phase of new e-voting protocol, each voter compute $2n$ exponentiations and $2n$ evaluations of a degree-t polynomial. Using y_0 and U, each voter also runs the proof scheme from Sec. 2.1 and generates a proof π_U, which consists of $(4|\mathbb{G}|, 4|\mathbb{Z}_q|)$ elements and requires an additional 6 exponentiations in the group. In the second phase, the verifier needs to verify both proofs π_{PPVSS} and π_U to check the validity of all ballots submitted by $\{V_j\}_{j=1}^m$. The computational cost of this step is approximately $2nm$ exponentiations and nm degree-t polynomial evaluations, where n and m are the number of talliers and voters, respectively. The tallying phase (i.e., the third phase) is identical for both protocols, therefore have the same efficiency.

Empirical Performance of E-Voting Protocols. In addition, we assessed the practical performance of both e-voting protocols by implementing them in Rust and comparing their performance. As in case of PPVSS schemes, our experiments are done using the Ristretto Group under `Curve25519` and the hash function `Blake3` for the random oracle. The tests were run on MacBook Pro with an M4 Pro CPU with 24GB RAM. All three phases, ballot casting, ballot verification, and tallying were limited to 8 threads to utilize the 8 performance cores available. All numbers are averaged over 100 iterations. We have summarized our implementation results for various parameter sets in Table 3.

Upon comparing the implementation outcomes, it becomes apparent that the new e-voting protocol outperforms Schoenmakers' scheme in all metrics. Notably, it yields a remarkable acceleration in the verification phase, particularly for large value of n, making it practical for even large-scale board-room elections. As an instance, the new scheme achieves verification times that are approximately $52\times$ and $90\times$ faster in comparison to the Schoenmakers' scheme for each voter with the parameter pairs (n, t) equal to $(256, 127)$ and $(512, 255)$. In large-scale elections, we observe that for $(n, t, m) = (17, 8, 100K)$, ballot casting takes just 0.42 milliseconds, each voter needs to broadcast 1104 bytes of data, each ballot can be verified in 0.26 milliseconds, and vote tallying is completed in 2.62 seconds.

Table 3. Implementation results of the universally verifiable e-voting protocols proposed in [22] and Section 5 for different types and sizes of elections. n: Number of talliers, t: Threshold value (number of malicious talliers), m: Number of voters, $|\mathbb{Z}_q| = |\mathbb{G}| = 256$ bits, s: seconds, ms: milliseconds, B: Byte, K: Kilo, M: Million.

Election, (n, t, m)	Metrics	Schoenmakers [22]	Our Scheme (Fig. 5)
Election 1 (board-room) (256, 127, 256)	Ballot Casting Time	4.65 ms	2.63 ms
	Voter's Communication	20.28 KB	12.28 KB
	Verification of One Ballot	121.15 ms	2.30 ms
	Tallying Time	11.18 ms	11.18 ms
Election 2 (board-room) (512, 255, 512)	Ballot Casting Time	9.92 ms	6.43 ms
	Voter's Communication	40.28 KB	24.28 KB
	Verification of One Ballot	481.19 ms	5.33 ms
	Tallying Time	27.91 ms	27.91 ms
Election 3, 4, 5 (large-scale) $(17, 8, m)$	Ballot Casting Time	0.56 ms	0.42 ms
	Voter's Communication	1648 B	1104 B
	Verification of One Ballot	1.24 ms	0.41 ms
$m = 50$K	Tallying Time	1.31 s	1.31 s
$m = 100$K	Tallying Time	2.61 s	2.61 s
$m = 1$M	Tallying Time	25.70 s	25.70 s

6 Conclusion

In this paper, we introduced Pre-constructed PVSS (PPVSS), an extension of standard PVSS schemes that offers improved flexibility and efficiency by requiring the dealer to publish a commitment (or an encryption) to the main secret. This new design, combined with a novel secret reconstruction approach, directly addresses the limitations of existing PVSS schemes in applications that require the dealer's commitment to the secret itself, such as e-voting, threshold key escrow, and other transparency-focused protocols [22].

A major contribution of our work is a general strategy for transforming Shamir-based PVSS schemes into PPVSS schemes, applicable across a wide range of PVSS schemes. This flexibility enables PPVSS schemes to support a broader array of applications, including areas such as distributed key generation and randomness generation protocols. Our constructions include both RO-based and plain-model PPVSS schemes, both of which are built on state-of-the-art PVSS protocols [3, 7, 8]. Through theoretical analysis and practical evaluation, we showed that these PPVSS schemes not only match the security properties of PVSS but also improves upon current alternative protocols in terms of efficiency and flexibility. The RO-based scheme, in particular, enables the revision of universally verifiable e-voting protocol from [22], that improves its overall performance and significantly reduces the computational complexity of ballot verification, especially in settings with a high number of talliers.

While our focus has been on PVSS schemes where the secret is g^{f_0} and parties can only reconstruct g^{f_0} in the reconstruction phase, we believe our general strategy can readily apply to other PVSS schemes where the secret is shared directly as f_0, e.g., [9, 18]. In addition, we revisited only a select few

applications of PPVSS schemes, emphasizing their adaptability and potential for improved efficiency. Given the inherent flexibility of PPVSS, we anticipate that these schemes could find even wider applicability in cryptographic protocols. This opens new avenues for future research to explore additional applications where PPVSS can offer advantages over traditional PVSS schemes.

We primarily examined PPVSS schemes based on the Discrete Logarithm (DL) assumption. However, constructing PPVSS schemes based on alternative assumptions, such as lattice-based cryptography, presents an interesting direction for further study, potentially broadening the scope of applications and enhancing security against quantum adversaries.

Our work used one positive feature of PPVSS schemes–publishing an encryption or commitment of the main secret under a specific public key–across three revisited applications. We also introduced a new reconstruction approach that may improve efficiency in PVSS-based protocols by supporting constant communication and verification costs under an optimistic reconstruction method. This approach leverages the assumption that in some applications, each participant acts as a dealer at least once, and a majority of participants are honest. In such cases, this optimistic reconstruction approach can streamline the reconstruction phase and reduce overall computational overhead.

In summary, our work provides a flexible, efficient PPVSS framework that expands the practical capabilities of secret sharing schemes, setting the stage for future advancements in publicly verifiable threshold protocols and their applications in secure and decentralized systems.

Acknowledgments

This work was supported by the Flemish Government through the Cybersecurity Research Program with grant number VOEWICS02.

References

1. Shahla Atapoor, Karim Baghery, Daniele Cozzo, and Robi Pedersen. VSS from distributed ZK proofs and applications. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings*, Lecture Notes in Computer Science. Springer, 2023.
2. Michael Backes, Aniket Kate, and Arpita Patra. Computational verifiable secret sharing revisited. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 590–609, Seoul, South Korea, December 4–8, 2011. Springer Berlin Heidelberg, Germany.
3. Karim Baghery. A unified framework for computational verifiable secret sharing. In Tibor Jager and Jiaxin Pan, editors, *Public-Key Cryptography - PKC 2025 - 28th IACR International Conference on Practice and Theory of Public-Key Cryptography, Roros, Norway May 12-15, 2025, Proceedings*, Lecture Notes in Computer Science.

4. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 1–10, Chicago, IL, USA, May 2–4, 1988. ACM Press.
5. Josh Cohen Benaloh. Secret sharing homomorphisms: Keeping shares of a secret sharing. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 251–260, Santa Barbara, CA, USA, August 1987. Springer Berlin Heidelberg, Germany.
6. Josh Cohen Benaloh and Moti Yung. Distributing the power of a government to enhance the privacy of voters (extended abstract). In Joseph Y. Halpern, editor, *5th ACM Symposium Annual on Principles of Distributed Computing*, pages 52–62, Calgary, Alberta, Canada, August 11–13, 1986. Association for Computing Machinery.
7. Ignacio Cascudo and Bernardo David. SCRAPE: Scalable randomness attested by public entities. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *ACNS 17: 15th International Conference on Applied Cryptography and Network Security*, volume 10355 of *Lecture Notes in Computer Science*, pages 537–556, Kanazawa, Japan, July 10–12, 2017. Springer, Cham, Switzerland.
8. Ignacio Cascudo and Bernardo David. ALBATROSS: Publicly AttestabLe BATched Randomness based On Secret Sharing. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part III*, volume 12493 of *Lecture Notes in Computer Science*, pages 311–341, Daejeon, South Korea, December 7–11, 2020. Springer, Cham, Switzerland.
9. Ignacio Cascudo and Bernardo David. Publicly verifiable secret sharing over class groups and applications to DKG and YOSO. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part V*, volume 14655 of *Lecture Notes in Computer Science*, pages 216–248, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.
10. Ignacio Cascudo, Bernardo David, Lydia Garms, and Anders Konring. YOLO YOSO: Fast and simple encryption and secret sharing in the YOSO model. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part I*, volume 13791 of *Lecture Notes in Computer Science*, pages 651–680, Taipei, Taiwan, December 5–9, 2022. Springer, Cham, Switzerland.
11. David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105, Santa Barbara, CA, USA, August 16–20, 1993. Springer Berlin Heidelberg, Germany.
12. Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *26th Annual Symposium on Foundations of Computer Science*, pages 383–395, Portland, Oregon, October 21–23, 1985. IEEE Computer Society Press.
13. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187, Santa Barbara, CA, USA, August 21–25, 1994. Springer Berlin Heidelberg, Germany.
14. Frank A. Feldman. Fast spectral tests for measuring nonrandomness and the DES. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 243–254, Santa Barbara, CA, USA, August 16–20, 1988. Springer Berlin Heidelberg, Germany.

15. Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science*, pages 427–437, Los Angeles, CA, USA, October 12–14, 1987. IEEE Computer Society Press.
16. 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*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer Berlin Heidelberg, Germany.
17. Craig Gentry, Shai Halevi, and Vadim Lyubashevsky. Practical non-interactive publicly verifiable secret sharing with thousands of parties. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part I*, volume 13275 of *Lecture Notes in Computer Science*, pages 458–487, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.
18. Jens Groth. Non-interactive distributed key generation and key resharing. Cryptology ePrint Archive, Report 2021/339, 2021.
19. 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: 15th Annual International Workshop on Selected Areas in Cryptography*, volume 5381 of *Lecture Notes in Computer Science*, pages 294–308, Sackville, New Brunswick, Canada, August 14–15, 2009. Springer Berlin Heidelberg, Germany.
20. Aniket Kate, Easwar Vivek Mangipudi, Pratyay Mukherjee, Hamza Saleem, and Sri Aravinda Krishnan Thyagarajan. Non-interactive VSS using class groups and application to DKG. Cryptology ePrint Archive, Report 2023/451, 2023.
21. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140, Santa Barbara, CA, USA, August 11–15, 1992. Springer Berlin Heidelberg, Germany.
22. Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 148–164, Santa Barbara, CA, USA, August 15–19, 1999. Springer Berlin Heidelberg, Germany.
23. Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.
24. Markus Stadler. Publicly verifiable secret sharing. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 190–199, Saragossa, Spain, May 12–16, 1996. Springer Berlin Heidelberg, Germany.
25. Eric R. Verheul and Henk C. A. van Tilborg. Binding ElGamal: A fraud-detectable alternative to key-escrow proposals. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 119–133, Konstanz, Germany, May 11–15, 1997. Springer Berlin Heidelberg, Germany.
26. Adam Young and Moti Yung. Auto-recoverable auto-certifiable cryptosystems. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 17–31, Espoo, Finland, May 31 – June 4, 1998. Springer Berlin Heidelberg, Germany.

A Appendix

A.1 Efficiency Comparison of PVSS Schemes

Table 4 (adapted from [3]) summarizes the detailed performance metrics for the established PVSS schemes [3, 7, 8, 10] and two particular (P)PVSS schemes from [19, 22] which are proposed to share the secret g^{f_0}.

Table 4. A comparison of established PVSS schemes for sharing g^s from [3,7,8,10] and two particular PPVSS schemes from [19,22]. BC: Dealer's Broadcast Communication, Dow.: Download size by a verifier, DL: Discrete Logarithm, PDL: Polynomial Discrete Logarithm, DDH: Decisional Diffie-Hellman, DBS: Decisional Bilinear Square, RO: Random Oracle, Plain: Plain Model, BC: Broadcast, n: Number of parties, t: threshold parameter ($t \approx n/2$), $P_{\mathbb{G}}$: Pairing Operation, $E_{\mathbb{G}}$: Exponentiation in group $\mathbb{G}$, $M_{\mathbb{G}}$: Multiplication in group $\mathbb{G}$, $\mathcal{PE}$: degree-t Polynomial Evaluation, $|\mathbb{G}|$: $\mathbb{G}$ element size, $|\mathbb{Z}_q|$: $\mathbb{Z}_q$ element size.

Scheme	Share	Broadcast & Dow.	Verification	Recon.
[22], PPVSS (DDH, RO)	$4.5n\ E_{\mathbb{G}}$ $1n\ \mathcal{PE}$	BC: $1.5n\|\mathbb{G}\| + n\|\mathbb{Z}_q\|$ Dow: $2.5n\|\mathbb{G}\| + n\|\mathbb{Z}_q\|$	$nt + 4n\ E_{\mathbb{G}}$ $+\ nt + 2n\ M_{\mathbb{G}}$	$5t\ E_{\mathbb{G}}$ $+\ t\ M_{\mathbb{G}}$
[19], PPVSS (DDH, Plain)	$1.5n\ E_{\mathbb{G}}$ $1n\ \mathcal{PE}$	BC: $1.5n\|\mathbb{G}\|$ Dow: $2.5n\|\mathbb{G}\|$	$nt\ E_{\mathbb{G}} + 2nP_{\mathbb{G}}$ $+\ nt\ M_{\mathbb{G}}$	$5t\ E_{\mathbb{G}}$ $+\ t\ M_{\mathbb{G}}$
[7], PVSS (DBS, Plain)	$2n\ E_{\mathbb{G}}$ $1n\ \mathcal{PE}$	BC: $2n\|\mathbb{G}\|$ Dow: $3n\|\mathbb{G}\|$	$2n\ P_{\mathbb{G}} +$ $n\ E_{\mathbb{G}} + n\ M_{\mathbb{G}}$	$2t\ P_{\mathbb{G}} +$ $t\ E_{\mathbb{G}} + t\ M_{\mathbb{G}}$
[7], PVSS (DDH, RO)	$4n\ E_{\mathbb{G}}$ $1n\ \mathcal{PE}$	BC: $2n\|\mathbb{G}\| + n\|\mathbb{Z}_q\|$ Dow: $3n\|\mathbb{G}\| + n\|\mathbb{Z}_q\|$	$5n\ E_{\mathbb{G}}$ $+\ 3n\ M_{\mathbb{G}}$	$5t\ E_{\mathbb{G}}$ $+\ t\ M_{\mathbb{G}}$
[10], PVSS (DDH, RO)	$4n\ E_{\mathbb{G}}$ $2n\ \mathcal{PE} + 3n\ M_{\mathbb{G}}$	BC: $n\|\mathbb{G}\| + 2\|\mathbb{Z}_q\|$ Dow: $2n\|\mathbb{G}\| + n\|\mathbb{Z}_q\|$	$2n\ E_{\mathbb{G}} +$ $n\ \mathcal{PE} + 2n\ M_{\mathbb{G}}$	$5t\ E_{\mathbb{G}} +$ $2t\ M_{\mathbb{G}}$
[3,8], PVSS (PDL, DDH, RO)	$2n\ E_{\mathbb{G}}$ $2n\ \mathcal{PE}$	BC: $n\|\mathbb{G}\| + 0.5n\|\mathbb{Z}_q\|$ Dow: $2n\|\mathbb{G}\| + 0.5n\|\mathbb{Z}_q\|$	$2n\ E_{\mathbb{G}} +$ $n\ \mathcal{PE} + n\ M_{\mathbb{G}}$	$5t\ E_{\mathbb{G}} +$ $t\ M_{\mathbb{G}}$

A.2 Overview of Schoenmakers' (P)PVSS Scheme

In CRYPTO'99, Schoenmakers [22] proposed a PVSS scheme, based on Feldman's VSS scheme, which allows a dealer to encrypt the shares under the public key of the parties, and then generate a publicly-verifiable NIZK proof to show that the secret sharing and encryptions are done correctly. Next, we present an overview of his proposed protocol.

Let g, h be two random generators of the group $\mathbb{G}$. In the initialization step, a party P_i generates a secret key $s_i \leftarrow\$\ \mathbb{Z}_q$ and registers $\mathsf{pk}_i = g^{s_i}$, as its public key. Then, given n and t, to share a *high-entropy* secret f_0, the dealer proceeds as follows:

1. Sample a uniformly random degree-t polynomial $f(X) := f_0 + a_1 X + \cdots + a_t X^t$ with coefficients in $\mathbb{Z}_q$, subject to $f(0) = f_0$.
2. For $i = 1, 2, \cdots, n$: set $f_i := f(i)$ and $y_i = \mathsf{pk}_i^{f(i)}$.
3. Set $C_0 = h^{f_0}$ and $C_j = h^{a_j}$ for $j = 1, 2, \cdots, t$.

4. Let $x_i = \prod_{j=0}^{t} C_j^{i^j}$, for $i = 1, 2, \cdots, n$. Then, the dealer shows that the encrypted shares y_i are consistent by producing a proof of knowledge of the unique $f(X)$, $1 \leq i \leq n$, satisfying: $x_i = h^{f(i)} \ \wedge \ y_i = \mathsf{pk}_i^{f(i)}$.
5. To generate the proof for above relation, the dealer uses an extended version of Chaum-Pedersen PoK scheme for DLEQ [11] and acts as follows:
   (a) For $i = 1, 2, \cdots, n$, it samples $r_i \leftarrow\!\!\$\ \mathbb{Z}_q$, and sets $a_i = h^{r_i}$ and $b_i = \mathsf{pk}_i^{r_i}$.
   (b) Using Fiat-Shamir transform, feeds $\{a_i, b_i, x_i, y_i\}_{i=1}^n$ into the random oracle $\mathcal{H}$, an obtains a challenge value $d \in \mathbb{Z}_q$.
   (c) For $i = 1, 2, \cdots, n$: computes $z_i = r_i - d \cdot f_i \mod q$.
6. Publish $\pi_{Share} := (h, C_j, \mathsf{pk}_i, y_i, d, z_i)$ for $0 \leq j \leq t$, and $1 \leq i \leq n$.

Verification. To verify the shares, given $\pi_{Share} := (h, C_j, \mathsf{pk}_i, y_i, d, z_i)$ for $0 \leq j \leq t$, and $1 \leq i \leq n$, the verifier acts as follows:

- For $1 \leq i \leq n$: computes $x_i = \prod_{j=0}^{t} C_j^{i^j}$.
- For $1 \leq i \leq n$: using $(h, d, x_i, \mathsf{pk}_i, y_i, z_i)$, computes a_i and b_i, as follows

$$a_i := h^{z_i} x_i^d \quad , \quad b_i := \mathsf{pk}_i^{z_i} (y_i)^d$$

  and checks if the hash of $\{a_i, b_i, x_i, y_i\}_{i=1}^n$ matches the challenge value d. If so returns `true`, otherwise returns `false`.

Reconstruction. To reconstruct the secret g^{f_0}, the parties proceed as follows.

1. They first use their secret key s_i, and obtain their share $F_i := g^{f_i}$ from y_i by computing $F_i = y_i^{1/s_i}$. Then, they publish F_i plus a NIZK proof that the value F_i is a correct decryption of y_i. To this end, the party P_i needs to prove knowledge of an s_i such that, $(\mathsf{pk}_i = g^{s_i}) \wedge (y_i = F_i^{s_i})$, which is done using Chaum-Pedersen [11] proof system for DLEQ (described in Fig. 1).
2. Then, given any $t+1$ valid values of F_i, w.l.o.g. for $i = 1, \ldots, t+1$, the secret $g^{f(0)}$, can be obtained by Lagrange interpolation,

$$\prod_{i=1}^{t+1} F_i^{\lambda_i} = \prod_{i=1}^{t+1} (g^{f_i})^{\lambda_i} = g^{\sum_{i=1}^{t+1} f_i \lambda_i} = g^{f(0)} = g^{f_0},$$

   where $\lambda_i = \prod_{j \neq i} \frac{j}{j-i}$ is a Lagrange coefficient.

B More Efficient Protocols Based on PPVSS

Next, we revisit two more applications of PPVSS schemes, proposed in [22].

B.1 More Efficient Threshold Binding ElGamal

In [25], Verheul and van Tilborg introduced a scheme called binding ElGamal, designed to strengthen a public-key infrastructure that relies on software key escrow. The binding ElGamal scheme enables a session key to be shared not only with the intended recipient but also with n trustees, who can collectively reconstruct the key using a (t, n)-threshold scheme. This approach, referred to as

threshold binding ElGamal, can be implemented through distributed key generation, as outlined in [25]. However, their approach involves multiple rounds of interaction, is computationally intensive, and requires repetition each time new participants are added or removed. To address these limitations, Schoenmakers [22] proposed using his (P)PVSS scheme as an alternative. While useful, Schoenmakers' protocol is less efficient, and recent PVSS schemes lack the ability to serve threshold binding ElGamal directly, as they do not provide commitment to the main secret.

Our proposed PPVSS schemes address this limitation in standard PVSS schemes. With our RO-based PPVSS, threshold binding ElGamal can be implemented efficiently. The sender, acting as the dealer, uses the PPVSS scheme from Fig. 3 to share the secret g^{f_0} with a committee. For this purpose, the sender simply uses the receiver's public key as pk_0. Unlike in Schoenmakers' protocol, our approach does not require the dealer to provide an additional DLEQ NIZK proof to verify that $y_0 = \mathsf{pk}_0^{f_0} \wedge C_0 = g^{f_0}$, where y_0 represents the encryption of f_0 and C_0 is the commitment to f_0, published as part of the transcript of Feldman VSS.

The need for an additional DLEQ NIZK proof highlights why Schoenmakers' PPVSS scheme is considered a special type of PPVSS, with less flexibility compared to our proposed PPVSS schemes. In Schoenmakers' design, the commitment to the secret f_0 cannot be replaced with encryption of f_0, while in our protocols this is possible without additional overhead. Additionally, our protocols deliver improved overall performance, especially in the verification phase.

Using the PPVSS scheme to share the secret g^{f_0} guarantees that both the intended receiver (i.e., the party who knows the secret key for pk_0) and the committee members will reconstruct the same value g^{f_0} through either optimistic or pessimistic reconstruction methods. The advantage of this approach is that the sender can pre-select a group of participants capable of reconstructing the secret without the need for a prior key generation protocol. This structure intuitively combines a full-threshold access structure with a (n, t)-threshold structure.

B.2 Threshold Software Key Escrow

In [26], Young and Yung presented a software key escrow solution allowing users to independently sample and register their secret and public keys with a certification authority. This approach involves distributing encrypted shares of the private key to a designated group of trustees using a secret sharing scheme. The key pair is accepted only if these encrypted shares match the public key registered with the authority.

Alternatively, Schoenmakers [22] proposes leveraging his (P)PVSS scheme to achieve this functionality more efficiently. In this version, the user generates a random private key $S = G^s$ and then distributes shares of this key among trustees using the (P)PVSS scheme. The user also publishes their public key $H = f(G^s)$, where f is a suitable generator (as referenced in [24]). Finally, to confirm that the trustees indeed receive valid shares of the private key $S = G^s$,

the user uses a protocol from [24] and proves knowledge of a witness s such that

$$H = f(G^s) \wedge C_0 = g^s, \tag{3}$$

where C_0 is published during the sharing phase of Schonmakers' (P)PVSS scheme.

To decrypt an ElGamal ciphertext $(x, y) = (f^\alpha, H^\alpha m)$, x is raised to the power G^s. Assuming the first t trustees participate in decrypting the ciphertext, and given that $G^s = \prod G^{p(i)\lambda_i}$, decryption is carried out by setting $z_0 = x$ and allowing each trustee to transform z_i as follows:

$$z_{i+1} = z_i^{G^{p(i)\lambda_i}} = z_i^{S^{\lambda_i}}.$$

If needed, each trustee can produce a proof confirming the correctness of their decryption step. Stadler's proof is applied here to show:

$$z_{i+1} = z_i^{Y^{\lambda_i/x_i}} \wedge g^{\lambda_i} = y^{\lambda_i/x_i},$$

where y_i represents the i-th trustee's public key, and Y_i is their encrypted share of G^s.

As in other applications, for enhanced performance, our proposed PPVSS scheme from Fig. 3 can be employed. In this modified setup, the dealer publishes a commitment $y_0 = g^s$ and proves its validity alongside the encryption of other shares. Subsequently, in equation (3), $y_0 = g^s$ is substituted for $C_0 = g^s$, while the rest of the protocol proceeds as outlined.

B.3 Homomorphic Property of New PPVSS Schemes

The notion of homomorphic secret sharing was introduced by Benaloh [5], highlighting its relevance to applications like electronic voting. Informally, homomorphic secret sharing enables combining shares of independent secrets so that reconstructing the combined shares results in a combined secret.

In the context of PVSS [22], there is an operation $\oplus$ on the shares and an operation $\otimes$ on the encrypted shares, such that for all participants:

$$E_i(f_i) \otimes E_i(f_i') = E_i(f_i \oplus f_i').$$

This property implies that by decrypting the $\otimes$-combined encrypted shares, the reconstructed secret will equal $f_i \oplus f_i'$, assuming that the underlying secret sharing scheme is $\oplus$-homomorphic. Our proposed PPVSS scheme adheres to this principle. For encrypted (committed) shares f_0 and f_0', their encrypted combination through multiplication yields a new encrypted commitment to $f_0 + f_0'$. The corresponding shares of participant i maintain the same property, leading to a homomorphic PPVSS that enables secure combination of shares. In Section 5, we use the new RO-based homomorphic PPVSS scheme and revisit Schoenmakers' universally verifiable e-voting protocol [22].