Batched and Packed (Publicly) Verifiable Secret Sharing: A Unified Framework and Applications

Shahla Atapoor[1], Karim Baghery[1], Georgio Nicolas[1], Robi Pedersen[2], and Jannik Spiessens[1]

[1] COSIC, KU Leuven, Leuven, Belgium
firstname.lastname@kuleuven.be
[2] Technical University of Denmark, Denmark
robi.pedersen@protonmail.com

Abstract. Verifiable Secret Sharing (VSS) allows a dealer to distribute a secret among n parties so that each can verify their share's validity, and any qualified subset can reconstruct the secret. Publicly Verifiable Secret Sharing (PVSS) extends VSS by enabling anyone to verify the correctness of distributed shares. Both VSS and PVSS schemes are core building blocks in many cryptographic applications. We introduce a *k-batched* and *l-packed* extension of Π, a unified framework from PKC 2025 for Shamir-based computational VSS in the synchronous setting with optimal resilience. Our framework enables the sharing and verification of $l \times k$ secrets in a single protocol execution, offering a tunable trade-off between efficiency and robustness: the k-batched, non-packed variant ($l = 1$) improves performance while maintaining optimal resilience, whereas the k-batched, l-packed variant achieves even greater efficiency at the cost of slightly reduced fault tolerance. Using this framework, we construct several Batched and Packed (BP) VSS and PVSS schemes that significantly reduce both computational and communication costs for the dealer and parties. When sharing many secrets, two of our VSS schemes and our PVSS scheme perform almost as efficiently as plain Shamir sharing. For example, when sharing more than 100 secrets, the overhead of our hash-based BP-VSS is below 3%, for our BP-VSS with information-theoretic privacy it remains around 8%, and for our BP-PVSS it is under 1%. These results show that verifiability in Shamir secret sharing can be achieved in post-quantum and large-scale settings with negligible overhead for the dealer. Our proposed BP-PVSS scheme is the first that can achieve these properties and outperforms existing state-of-the-art protocols. We discuss several applications and, as a concrete case study, use our BP-PVSS scheme to revisit the ALBATROSS randomness generation protocol from ASIACRYPT 2020, yielding a more efficient variant.

Keywords: Verifiable Secret Sharing · Batched and Packed Verifiable Secret Sharing · Batched PVSS · Shamir Secret Sharing · ALBATROSS

1 Introduction

Secret sharing schemes are fundamental building blocks in modern cryptographic systems that aim to distribute trust among multiple parties. They serve as

core components in secure multi-party computation and threshold cryptography, enabling the construction of distributed protocols without relying on a single trusted entity. These protocols allows a dealer to distribute a secret among a set of participants with the guarantee that only authorized subsets of participants can reconstruct the secret, while unauthorized subsets learn nothing about it. Two well-known traditional secret sharing schemes have been proposed in 1979 by Shamir [31] and Blakley [9]. Shamir's secret sharing scheme [31] relies on polynomial interpolation over finite fields to divide a secret into parts, with any subset of participants larger than a predetermined threshold able to reconstruct the secret. Blakley's [9] geometric approach achieves a similar goal using the intersection properties of hyperplanes. Shamir and Blakley focus on securely distributing a secret, but they do not provide mechanisms to verify the integrity and honesty of the shares distributed by the dealer. In other words, while these schemes are robust against passive adversaries, they do not address active adversaries who may distribute incorrect shares during the sharing phase or provide invalid inputs during secret reconstruction.

(Publicly) Verifiable Secret Sharing. Verifiable Secret Sharing (VSS) [2,4–6, 12,19,22,25,29] and Publicly Verifiable Secret Sharing (PVSS) [5,13–15,26,27,30] protocols were developed to overcome the limitations of classical secret sharing schemes, such as Shamir's [31], which are secure only against passive adversaries. VSS protocols address this by allowing participants to verify the consistency of their received shares, ensuring that a malicious dealer cannot distribute invalid or inconsistent shares without detection. PVSS schemes further extend this functionality by enabling anyone, including external observers, to verify the correctness of the shared values, making them especially useful in publicly auditable settings. VSS protocols can be classified by their security guarantees and communication assumptions. In terms of adversarial models, VSS schemes can be either Information-Theoretically (IT) secure, offering unconditional guarantees, or computationally secure, relying on the hardness of cryptographic assumptions. The communication setting also significantly affects protocol design: in synchronous networks, where message delays are bounded, computationally secure VSS schemes can tolerate up to $t < n/2$ corrupted parties. In contrast, asynchronous networks, where message delays are unbounded, typically require a greater fraction of honest participants to ensure correctness and security. The recent demand for highly efficient and Post-Quantum (PQ) secure VSS schemes has spurred the development of hash-based (lightweight) Shamir-based protocols [2, 5, 12, 32]. These modern VSS schemes aim to minimize computational and communication overhead while maintaining (PQ) security against malicious parties. Building upon the ideas from [2], Baghery recently proposed a unified framework called Π to build Shamir-based computationally secure VSS schemes in the synchronous communication model that can achieve optimal resilience.

The Need for Batched and Packed (P)VSS Protocols. VSS and PVSS are fundamental building blocks in distributed cryptographic protocols including threshold cryptosystems, Distributed Key Generation (DKG), threshold signatures, secure Multi-Party Computation (MPC), distributed randomness genera-

tion, and privacy-preserving federated learning [2,14,20,21,24,33]. Despite their foundational role, these protocols often encounter efficiency bottlenecks in practice, particularly in cases that require sharing a large number of secrets.

These challenges motivate the development of Batched VSS (BVSS) protocols. These protocols aim to amortize the computational and communication costs across multiple VSS instances while preserving optimal resilience. A k-BVSS protocol enables a dealer to share k secrets with the same set of participants in a single protocol execution without compromising the protocol's resilience. Functionally, this is equivalent to running Shamir's protocol k times, once per secret, with the same (n, t) parameters, but generating a single proof to prove the correctness of all $n \times k$ distributed shares. This approach can significantly improve both computational and communication efficiency compared to executing k separate VSS protocols, once per secret. BVSS is particularly well-suited for applications that involve repeated sharing among a fixed group of parties, offering performance gains without sacrificing robustness.

An alternative approach is offered by *packed secret sharing schemes* [10,23], which improve the efficiency of Shamir secret sharing by encoding ℓ secrets into a single polynomial, without increasing the share size. Standard Shamir sharing is a special case where $\ell = 1$. However, since at most $t+1$ independent secrets can be securely embedded in a degree-t polynomial, packing ℓ secrets requires increasing the degree to $t + \ell$ and ensuring that the number of parties satisfies $n \geq 2t + \ell$ in the synchronous setting. As a result, increasing the packing factor ℓ typically reduces the tolerable corruption threshold t, weakening the scheme's robustness. In essence, packed secret sharing trades resilience for efficiency: it provides t-privacy and $(t + \ell)$-reconstruction, but does not achieve optimal resilience.

Given the complementary strengths and limitations of the batched and packed approaches, a compelling research question emerges:

Can we design practical, tunable VSS or PVSS schemes that support both batching and packing, thereby unifying their efficiency and robustness benefits while preserving verifiability?

Our Contributions. Our contributions can be summarized as follows:

A Unified Tunable Framework for Batched and Packed VSS. Our primary contribution is proposing a tunable variant of the Π framework, originally introduced by Baghery [5], designed to support both *batched* and *packed* secret sharing. The original Π framework facilitates the construction of Shamir-based, computationally secure VSS schemes in the synchronous communication model with optimal resilience. Our new variant, denoted (k, ℓ)-BΠ (or simply BΠ), retains the core requirements of the original, namely, a random oracle and a secure (not necessarily homomorphic) vector commitment scheme. However, (k, ℓ)-BΠ distinguishes itself by enabling the simultaneous sharing and verification of $k \times \ell$ secrets in a single protocol execution. Figure 1 visualizes how $k \times \ell$ secrets and $n \times k$ shares are encoded in (k, ℓ)-BΠ via k independent random polynomials of degree $t + \ell - 1$. Each of the ℓ secrets is encoded at points $j = 0, -1, -2, \ldots, -(\ell - 1)$ of a degree-$(t + \ell - 1)$ polynomial. The dealer then

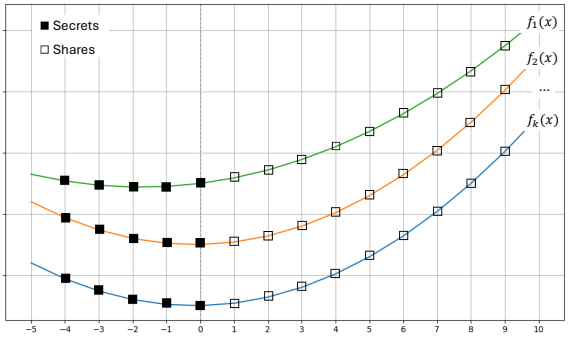


Fig. 1. A visualization of $k \times \ell$ secrets (■) and $k \times n$ shares (□) encoded in (k, ℓ)-BΠ via k random degree $t + \ell - 1$ polynomials. The $k \times \ell$ secrets are encoded at points $j = 0, -1, \ldots, -(\ell - 1)$, while the $k \times n$ shares are encoded at points $i = 1, 2, \ldots, n$. A dealer produces a single threshold proof to prove the correctness of all $k \times n$ shares.

generates a single threshold proof to prove the validity of all $k \times n$ shares, which are encoded at points $i = 1, 2, \ldots, n$ of the k random polynomials. The BΠ framework can yield substantial efficiency gains over executing the original Π protocol separately for each secret. Specifically, when sharing k secrets among n participants, the $(k, 1)$-BΠ variant reduces the total proof size from $O(nk\lambda)$ to $O((n+k)\lambda)$, without sacrificing optimal resilience, where λ denotes the security parameter. In addition to reduced communication, the computational load for both the dealer and participants is significantly decreased. Further performance improvements can be achieved by setting $\ell > 1$, which allows ℓ secrets to be packed into a single polynomial. However, this comes at the cost of lowering the protocol's resilience, as the number of tolerable corruptions must be reduced. The tunable nature of BΠ makes it a versatile framework for building a wide range of computationally secure VSS schemes in the synchronous setting, supporting batching, packing, or both.

Practical Batched and Packed VSS (BP-VSS) Schemes. Building on the BΠ framework and employing various vector commitment schemes, we construct a suite of practical, computationally secure Batched and Packed VSS (BP-VSS) protocols: namely, BΠ_{LA}, BΠ_{P}, $\text{BΠ}_{\text{P}}+$, and BΠ_{F}. Each instantiation offers distinct features, security guarantees, and efficiency trade-offs, tailored to different application needs. More concretely, we instantiate BΠ using both discrete logarithm- and hash-based vector commitments to derive new BP-VSS protocols. The new BP-VSS schemes serve as batched and packed counterparts to the original VSS schemes Π_{LA}, Π_{P}, and Π_{F} from [5].

BΠ_{F} is the batched and packed variant of Π_{F}, an alternative to Feldman's VSS [22]. Like Feldman's scheme, BΠ_{F} requires high-entropy secrets and relies on the hardness of the Discrete Logarithm (DL) problem. In contrast, BΠ_{P} is the batched and packed analogue of Π_{P} [5]. Much like Pedersen's VSS [29], it supports both high- and low-entropy secrets while achieving perfect simulatabil-

ity, ensuring that even a computationally unbounded adversary with access to t shares learns nothing about the secrets.

To further improve $\mathrm{B}\Pi_{\mathrm{P}}$ and construct $\mathrm{B}\Pi_{\mathrm{P}}+$, we introduce a novel perfectly hiding (basic) vector commitment scheme that achieves substantially better performance than the commonly used additively homomorphic Pedersen vector commitment. Our scheme requires only two exponentiations and one hash evaluation, in contrast to the $n+1$ exponentiations needed in the vector variant of Pedersen commitment. We believe this commitment scheme may be of independent interest in applications requiring non-homomorphic and perfectly hiding commitments. In particular, it can also enhance the efficiency of the original Π_{P} VSS protocol [5] in both the sharing and verification phases.

Lastly, $\mathrm{B}\Pi_{\mathrm{LA}}$ represents a batched and packed variant of the hash-based VSS scheme Π_{LA} [5], and offers an optimized and batched alternative to the ABCP VSS protocol [2]. $\mathrm{B}\Pi_{\mathrm{LA}}$ also serves as a synchronous and packed alternative to the asynchronous Shoup-Smart VSS protocol [32], operating under the milder assumption of an honest majority rather than the stricter 2/3 bound.

Batched and Packed Variants of Pedersen and Feldman Schemes. As an additional contribution, in the full version of this paper [3], we introduce batched and packed version of the original Feldman and Pedersen VSS schemes [22, 29]. These protocols achieve significant efficiency gains over k sequential executions of their base protocols: broadcast communication is reduced by the factor k, and verification cost is lowered from $\mathcal{O}(kn)$ to $\mathcal{O}(k+n)$ exponentiations, where n denotes the number of parties and k the batching parameter. While their verification is less efficient compared to our BΠ-based schemes, they offer the advantage of operating in the plain model.

Efficiency of New BP-VSS Schemes. Table 1 summarizes the asymptomatic efficiency of our new schemes, reporting both communication and computation costs, and highlighting natural groupings of constructions that achieve comparable security guarantees. For benchmarking, we compare these results against k concurrent executions of the packed variants of the original protocols from [5, 22, 29]. To ensure fairness, we first computed the packed costs of the original schemes (even though these were not explicitly presented in [5, 22, 29]).

An Efficient Batched and Packed PVSS Scheme. While our proposed BP-VSS schemes, $\mathrm{B}\Pi_{\mathrm{F}}$, $\mathrm{B}\Pi_{\mathrm{P}}$, $\mathrm{B}\Pi_{\mathrm{P}}+$, and $\mathrm{B}\Pi_{\mathrm{LA}}$, support only designated verification (i.e., correctness of distributed shares can only be verified by a subset of the respective recipients), our next contribution addresses the need for public verifiability. We introduce a Batched and Packed PVSS scheme, denoted $\mathrm{B}\Pi_{\mathrm{S}}$, built upon the $\mathrm{B}\Pi_{\mathrm{F}}$ BP-VSS scheme. This new construction supports both batched and packed secret sharing with public verifiability. To the best of our knowledge, $\mathrm{B}\Pi_{\mathrm{S}}$ is the first PVSS scheme that enables both batching and packing of secrets.

To construct $\mathrm{B}\Pi_{\mathrm{S}}$, we first develop a batched version of the generalized Schnorr protocol introduced by Baghery [5], which allows a prover to demonstrate knowledge of multiple witness polynomials in the Polynomial Discrete Logarithm (PDL) relation defined therein. We refer to this new batched sigma

Table 1. Performance of the proposed Batched and Packed VSS (BP-VSS) schemes for sharing $k \times \ell$ secrets, compared with k concurrent executions of the packed variants of the original protocols from [5, 22, 29]. Abbreviations: Comm.: Communication, (q)RO: (quantum) Random Oracle, Plain: Plain Model, BC: Broadcast, n: Number of parties, t: number of malicious parties ($t \approx \frac{n-\ell}{2}$), k: batching parameter, ℓ: number of secrets encoded in each function, $E_{\mathbb{G}}$: Exponentiation in group $\mathbb{G}$, $M_{\mathbb{G}}$: Multiplication in group $\mathbb{G}$, $M_{\mathbb{F}}$: Multiplication in finite field $\mathbb{F}$, $\mathcal{PE}$: degree-d Polynomial Evaluation with $d = t + \ell - 1$, $\mathcal{H}$: Hashing, $|X|$: X element size, $|\mathcal{H}|$: Output size of $\mathcal{H}$.

BP-VSS & Model	Sharing	Dealer's Comm.	Verification	Reconstruct
k runs of packed Fel [22], plain	$dk\ E_{\mathbb{G}}$ $nk\ \mathcal{PE}$	Private: $1nk\lvert\mathbb{Z}_q\rvert$ BC: $dk\lvert\mathbb{G}\rvert$	$dk + k\ E_{\mathbb{G}} +$ $dk\ M_{\mathbb{G}}$	$d^2k + dk\ E_{\mathbb{G}}$ $d^2k\ M_{\mathbb{G}}$
BP-Feldman full version, plain	$dk\ E_{\mathbb{G}}$ $nk\ \mathcal{PE}$	Private: $1nk\lvert\mathbb{Z}_q\rvert$ BC: $d\lvert\mathbb{G}\rvert$	$d + k\ E_{\mathbb{G}} +$ $d + k\ M_{\mathbb{G}}$	$d^2 + dk\ E_{\mathbb{G}} +$ $d^2 + dk\ M_{\mathbb{G}}$
k runs of packed Π_{F} [5], RO	$2nk\ E_{\mathbb{G}}$ $2nk\ \ \mathcal{PE}$	Private: $1nk\lvert\mathbb{Z}_q\rvert$ BC: $nk\lvert\mathbb{G}\rvert + dk\lvert\mathbb{Z}_q\rvert$	$2k\ E_{\mathbb{G}} +$ $k\ \mathcal{PE} + k\ \mathcal{H}$	$2dk\ E_{\mathbb{G}} +$ $dk\ \mathcal{PE} + dk\ \mathcal{H}$
$\mathrm{B}\Pi_{\mathrm{F}}$, Sec. 4.4 RO	$nk + n\ E_{\mathbb{G}}$ $nk + n\ \mathcal{PE}$	Private: $1nk\lvert\mathbb{Z}_q\rvert$ BC: $n\lvert\mathbb{G}\rvert + d\lvert\mathbb{Z}_q\rvert$	$k\ E_{\mathbb{G}} +$ $1\ \mathcal{PE} + 1\ \mathcal{H}$	$dk\ E_{\mathbb{G}} +$ $\mathrm{d}\ \mathcal{PE} + \mathrm{d}\ \mathcal{H}$
k runs of packed Ped [29], plain	$dk + d\ E_{\mathbb{G}}$ $nk + n\ \mathcal{PE}$	Private: $2nk\lvert\mathbb{Z}_q\rvert$ BC: $dk\ \lvert\mathbb{G}\rvert$	$dk + k\ E_{\mathbb{G}} +$ $kd\ M_{\mathbb{G}}$	$d^2k + dk\ E_{\mathbb{G}} +$ $kd^2\ M_{\mathbb{G}}$
BP-Pedersen full version, plain	$dk + d\ E_{\mathbb{G}}$ $nk + n\ \mathcal{PE}$	Private: $2nk\lvert\mathbb{Z}_q\rvert$ BC: $d\lvert\mathbb{G}\rvert$	$d + k\ E_{\mathbb{G}} +$ $d + k\ M_{\mathbb{G}}$	$d^2 + dk\ E_{\mathbb{G}} +$ $d^2 + dk\ M_{\mathbb{G}}$
k runs of packed Π_{P} [5], RO	$3nk\ E_{\mathbb{G}}$ $2nk\ \ \mathcal{PE}$	Private: $2nk\lvert\mathbb{Z}_q\rvert$ BC: $nk\lvert\mathbb{G}\rvert + dk\lvert\mathbb{Z}_q\rvert$	$3k\ E_{\mathbb{G}} +$ $k\ \mathcal{PE} + k\ \mathcal{H}$	$3dk\ E_{\mathbb{G}} +$ $dk\ \mathcal{PE} + dk\ \mathcal{H}$
$\mathrm{B}\Pi_{\mathrm{P}}$, Sec. 4.2 RO	$nk + 2n\ E_{\mathbb{G}}$ $nk + n\ \ \mathcal{PE}$	Private: $2nk\lvert\mathbb{Z}_q\rvert$ BC: $n\lvert\mathbb{G}\rvert + d\lvert\mathbb{Z}_q\rvert$	$k\ E_{\mathbb{G}} + 1\ \mathcal{PE} +$ $1\ \mathcal{H} + 2K\ M_{\mathbb{G}}$	$dk\ E_{\mathbb{G}} + d\ \mathcal{PE} +$ $d\ \mathcal{H} + 2dK\ M_{\mathbb{G}}$
$\mathrm{B}\Pi^{+}_{\mathrm{P}}$, Sec. 4.2 RO	$3n\ E_{\mathbb{G}} + n\ \mathcal{H}$ $nk + n\ \ \mathcal{PE}$	Private: $2kn\lvert\mathbb{Z}_q\rvert$ BC: $n\lvert\mathbb{G}\rvert + d\lvert\mathbb{Z}_q\rvert$	$2\ E_{\mathbb{G}} + 1\ \mathcal{PE} +$ $2\ \mathcal{H} + k\ M_{\mathbb{G}}$	$2d\ E_{\mathbb{G}} + d\ \mathcal{PE} +$ $2d\ \mathcal{H} + dk\ M_{\mathbb{G}}$
k runs of packed Π_{LA} [5] (q)RO	$1nk\ \mathcal{H}$ $2nk\ \ \mathcal{PE}$	Private: $1nk\lvert\mathbb{Z}_N\rvert$ BC: $nk\lvert\mathcal{H}\rvert + dk\lvert\mathbb{Z}_N\rvert$	$k\ \mathcal{PE} +$ $+ 2k\ \mathcal{H}$	$dk\ \mathcal{PE} +$ $+ 2dk\ \mathcal{H}$
$\mathrm{B}\Pi_{\mathrm{LA}}$, Sec. 4.1 (q)RO	$1n\ \mathcal{H}$ $nk + n\ \ \mathcal{PE}$	Private: $1nk\lvert\mathbb{Z}_N\rvert$ BC: $n\lvert\mathcal{H}\rvert + d\lvert\mathbb{Z}_N\rvert$	$1\ \mathcal{PE} +$ $2\ \mathcal{H} + k\ M_{\mathbb{F}}$	$d\ \mathcal{PE} +$ $2d\ \mathcal{H} + dk\ M_{\mathbb{F}}$

protocol as π_{BPDL}. In π_{BPDL}, a prover can prove knowledge of k secret polynomials at nearly the same computational and communication cost as proving a single one in the original protocol [5]. We believe the π_{BPDL} protocol may be of independent interest. Table 2 summarizes the performance of the new BP-PVSS scheme $\mathrm{B}\Pi_{\mathrm{S}}$ and compares it to the case of running k concurrent executions of the packed PVSS protocol Π_{S} [5, 14] and the packed YOLO-YOSO [16] PVSS protocol to share $k \times \ell$ secrets. As can be seen from Table 2, our proposed scheme $\mathrm{B}\Pi_{\mathrm{S}}$ outperforms the state-of-the-art schemes across all evaluated metrics. Notably, compared to Π_{S}, it achieves approximately a 50% improvement in the computational cost of both the sharing and verification phases. Furthermore, thanks to the batched proof π_{BPDL}, the overall proof size is reduced by a factor of k, significantly decreasing both the communication overhead and the verifier's workload. In addition, $\mathrm{B}\Pi_{\mathrm{S}}$ outperforms YOLO-YOSO in sharing, verification, and reconstruction, while also providing a smaller amortized proof size whenever

Table 2. Comparison of the proposed BP-PVSS scheme $\mathrm{B}\Pi_{\mathrm{S}}$ for sharing $k \times \ell$ secrets with k concurrent executions of the packed PVSS scheme $\Pi_{\mathrm{S}}$and YOLO-YOSO [5, 14, 16]. Abbreviations: PDL: Polynomial Discrete Logarithm, DDH: Decisional Diffie-Hellman, RO: Random Oracle, n: Number of parties, t: number of malicious parties ($t \approx \frac{n-\ell}{2}$), k: batching parameter, ℓ: number of secrets encoded in each polynomial, $E_{\mathbb{G}}$: Exponentiation in group $\mathbb{G}$, $M_{\mathbb{G}}$: Multiplication in group $\mathbb{G}$, $\mathcal{PE}$: degree-d Polynomial Evaluation with $d = t + \ell - 1$, $\mathcal{H}$: Hashing, $|X|$: X element size, $|\mathcal{H}|$: Output size of $\mathcal{H}$.

BPVSS & Security	Sharing	Dealer's Comm.	Verification	Reconstruct
k times packed YOLO [16] (DDH, RO)	$4nk\ E_{\mathbb{G}}$ $2nk\ \mathcal{PE}$	Cipher: $nk\ \lvert\mathbb{G}\rvert$ Proof: $2k\ \lvert\mathbb{Z}_q\rvert$	$2nk\ E_{\mathbb{G}} +$ $nk\ \mathcal{PE} + 2nk\ M_{\mathbb{G}}$	$5dk\ E_{\mathbb{G}} +$ $2dk\ M_{\mathbb{G}}$
k times packed Π_{S} [5, 14] (PDL, DDH, RO)	$2nk\ E_{\mathbb{G}}$ $2nk\ \mathcal{PE}$	Cipher: $nk\ \lvert\mathbb{G}\rvert$ Proof: $dk\ \lvert\mathbb{Z}_q\rvert$	$2nk\ E_{\mathbb{G}} +$ $nk\ \mathcal{PE} + nk\ M_{\mathbb{G}}$	$5dk\ E_{\mathbb{G}} +$ $dk\ M_{\mathbb{G}}$
$\mathrm{B}\Pi_{\mathrm{S}}$, Sec. 5.2 (PDL, DDH, RO)	$nk + n\ E_{\mathbb{G}}$ $nk + n\ \mathcal{PE}$	Cipher: $nk\ \lvert\mathbb{G}\rvert$ Proof: $d\ \lvert\mathbb{Z}_q\rvert$	$nk + n\ E_{\mathbb{G}} +$ $n\ \mathcal{PE} + nk\ M_{\mathbb{G}}$	$3dk\ E_{\mathbb{G}} +$ $dk\ M_{\mathbb{G}}$

$d < 2k$, where d is the degree of underlying secret polynomial. We also highlight that in a DL-based PVSS scheme even with a trusted dealer, the dealer must run the Shamir scheme k times, requiring nk polynomial evaluations, and perform at least nk encryptions to deliver k ciphertexts to each of the n parties. Thus, even in the trusted setting, the baseline cost is nk exponentiations plus nk polynomial evaluations, which serves as the reference point for efficiency comparisons. Relative to this baseline, $\mathrm{B}\Pi_{\mathrm{S}}$ achieves nearly identical efficiency.

Implementation Results. To evaluate the practical efficiency of our schemes, we developed a prototype implementation in Rust (see Sec. 6). The results show the new BP-VSS and BP-PVSS protocols are state-of-the-art, offering high computational and communication efficiency and suitability for large-scale cryptographic applications. In particular, when sharing many secrets, the BP-VSS schemes $\mathrm{B}\Pi_{\mathrm{LA}}$ and $\mathrm{B}\Pi_{\mathrm{P}}+$ and the BP-PVSS scheme $\mathrm{B}\Pi_{\mathrm{S}}$ achieve performance nearly identical to plain Shamir sharing in the distribution phase. For over 100 secrets, compared to plain Shamir, the overhead is below 3% for our hash BP-VSS, around 8% for BP-VSS with information-theoretic privacy, and under 1% for BP-PVSS. This confirms verifiability in Shamir secret sharing can be achieved in post-quantum, large-scale settings with negligible dealer overhead.

ALBATROSS+: Improving Performance of ALBATROSS with $\mathrm{B}\Pi_{\mathrm{S}}$. As our final contribution, and as a practical application of $\mathrm{B}\Pi_{\mathrm{S}}$, we introduce ALBATROSS+, an improved version of the ALBATROSS distributed randomness generation protocol [14], ranking among the most efficient PVSS-based constructions [28]. The core innovation lies in replacing the original protocol's ℓ-packed Π_{S} PVSS with our proposed $\mathrm{B}\Pi_{\mathrm{S}}$, a k-batched and ℓ-packed version of Π_{S}. In terms of efficiency, ALBATROSS+ outperforms the original in both computational and communication costs. Notably, it reduces the amortized proof size in protocol from $O(k)$ to $O(1)$, where $k = \frac{L}{(n-2t)^2}$ represents the repetition count in the original protocol, and L denotes the total number of random values that n parties (where up to t of them can be malicious) collectively generate. These improvements preserve the security guarantees of the original protocol,

while significantly boosting it efficiency, enabling practical high-throughput randomness generation with even few participants.

2 Preliminaries

Due to space constraints, we mainly treat verifiable secret sharing in this section and provide a concise overview of the Schwartz-Zippel Lemma, vector commitment schemes and Sigma-protocols along with some relevant constructions in supplementary materials of the full version of this paper.

Notation. We let λ denote a security parameter. A function 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 $x > x_0$. A function that is negligible in the security parameter λ is simply called negligible. We use the assignment operator $\leftarrow$ to denote uniform sampling from a set S, e.g. $y \leftarrow S$. We write $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ with known prime q, and $\mathbb{Z}_q[x]_t$ for polynomials of degree t in the variable x and with coefficients in $\mathbb{Z}_q$. For $n \in \mathbb{N}$, we write $[n] = \{1, \ldots, n\}$. Finally, all logarithms are in base 2.

2.1 Shamir, Packed, Batched, and Verifiable Secret Sharing

Shamir Secret Sharing. A $(n, t+1)$-Shamir secret sharing scheme [31] allows a secret f_0 to be distributed among n parties such that any subset of at most t parties learn no information about the secret, while any subset of at least $t+1$ parties can efficiently reconstruct it. This is achieved via polynomial interpolation over the the finite field $\mathbb{Z}_q$. Specifically, a random polynomial $f(x) \in \mathbb{Z}_q[x]_t$ of degree at most t is sampled, with its free term set as the secret, i.e., $f(0) = f_0$. 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 size at least $t+1$ parties can reconstruct f_0 using Lagrange interpolation: $f_0 = f(0) = \sum_{i \in Q} f_i \cdot L_{0,i}^Q$, where

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

are the Lagrange basis coefficients evaluated at 0. Note that in this setting $n \geq 2t+1$ ensures both t-privacy, i.e. any set of up to t parties cannot learn anything about the secret, and reconstruction by at least $t+1$ parties. Further, we can achieve optimal resilience.

Packed Shamir Secret Sharing. Packed Shamir secret sharing [10, 14, 23] generalizes the classical Shamir scheme by allowing multiple secrets to be shared simultaneously using a single polynomial, without increasing the size of each share. This is achieved by encoding ℓ secrets into a single polynomial over $\mathbb{Z}_q$ evaluated at ℓ publicly known positions. Let $f_0, f_{-1}, \ldots, f_{-(\ell-1)} \in \mathbb{Z}_q$ denote the ℓ secrets to be shared. A random polynomial $f(x) \in \mathbb{Z}_q[x]$ of degree $d = t+\ell-1$ is sampled such that: $f(j) = f_j \quad \text{for} \quad j = 0, -1, \ldots, -(\ell-1)$, where t denotes the

number of malicious parties the scheme tolerates. Each party P_j for $j = 1, \ldots, n$ receives $f_j = f(j)$ as its share. Given any $t+\ell$ valid shares, the entire polynomial $f(x)$, and hence all ℓ secrets, can be uniquely reconstructed via interpolation.

Although the packed variant offers substantial space efficiency by embedding multiple secrets into one polynomial (keeping share sizes constant), it comes with a trade-off in fault tolerance. Specifically, it requires $n \geq 2t + \ell$, and the degree of the polynomial increases with ℓ, thereby reducing the number of tolerated corruptions for a fixed n. Thus, the packed Shamir achieves t-privacy and $(t + \ell)$-reconstruction, sacrificing optimal resilience for improved efficiency. We also note that the evaluation points used for secrets $(0, -1, \ldots, -(\ell - 1))$ and shares $(1, \ldots, n)$ may be replaced by any set of $n+\ell$ pairwise distinct points. Alternative choices may lead to more efficient algorithms for both share computation and secret reconstruction [34].

Batched Verifiable Secret Sharing. Standard secret sharing schemes are typically secure only against passive adversaries. However, many applications require security even in the presence of malicious behaviour by the dealer or parties, which is addressed by Verifiable Secret Sharing (VSS) schemes [19]. VSS schemes enable a dealer to distribute a secret among a group of parties in a way that allows each party to verify the correctness of their share, ensuring that only a qualified subset can later reconstruct the secret.

In numerous real-world scenarios, it is either necessary or beneficial to run multiple VSS protocols simultaneously in a batched fashion. That is, a dealer holding k independent secrets can share them in parallel using k separate degree-t polynomials, where t denotes the number of malicious parties the system can tolerate. Importantly, unlike packed secret sharing, this approach preserves optimal resilience: the sharing of each secret is independent, so fault tolerance does not degrade with the number of secrets shared. Below, we summarize the formal notion of Batched VSS (BVSS) schemes, a natural generalization of classical VSS protocols, as proposed in works such as [2, 5, 29].

Definition 1. *An (n, t)-BVSS with security parameter λ consists of four PPT algorithms of (Initial, Share, Verify, Reconstruct) as follows:*

1. Initial*: In this phase, public parameters are sampled and shared with the participants.*
2. $\mathsf{Share}(n, t, (f_0^{\langle 1\rangle}, \ldots, f_0^{\langle k\rangle})) \rightarrow (\{f_i^{\langle 1\rangle}, \ldots, f_i^{\langle k\rangle}\}_{i=1}^n, \pi_{share})$*: Given (n, t), k secrets, it outputs the shares $\{f_i^{\langle 1\rangle}, \ldots, f_i^{\langle k\rangle}\}_{i=1}^n$, and a (non-interactive) threshold proof π_{share} to prove the validity of the shares.*
3. $\mathsf{Verify}(n, t, \{f_i^{\langle 1\rangle}, \ldots, f_i^{\langle k\rangle}\}_{i=1}^n, \pi_{share}) \rightarrow$ `true/false`*: Given (n, t), the shares $\{f_i^{\langle 1\rangle}, \ldots, f_i^{\langle k\rangle}\}_{i=1}^n$, and a threshold proof π_{share}, generated by* Share*, it outputs either* `true` *or* `false`*.*
4. $\mathsf{Reconstruct}(\{f_i^{\langle 1\rangle}, \ldots, f_i^{\langle k\rangle}\}_{i \in Q, |Q| \geq t+1}) \rightarrow \{f_0^{\langle 1\rangle}, \ldots, f_0^{\langle k\rangle}, \mathsf{false}\}$*: Given any at least $t + 1$ set of k shares, e.g., $\{f_i^{\langle 1\rangle}, \ldots, f_i^{\langle k\rangle}\}_{i=1}^{t+1}$, it either reconstructs and returns the secrets $(f_0^{\langle 1\rangle}, \ldots, f_0^{\langle k\rangle})$, or it returns* `false`*.*

A BVSS further requires satisfying the following properties.

- **Completeness:** If the dealer honestly executes the Share algorithm, then the Verify algorithm will always return $\mathtt{true}$. Moreover, any subset of at least $t+1$ honest parties will be able to reconstruct the shared secrets $(f_0^{\langle 1\rangle}, \ldots, f_0^{\langle k\rangle})$.
- **Verifiability constraint**: A shareholder must be able to verify the validity of the received shares. If the verification returns $\mathtt{true}$, then $\mathsf{Reconstruction}$ should produce *unique secrets* $(f_0^{\langle 1\rangle}, \ldots, f_0^{\langle k\rangle})$ when run on any $t+1$ distinct valid shares. More formally, for any integers $n \geq 2t+1$ and $t \geq 0$, a BVSS is called verifiable if for any PPT adversaries $\mathcal{A}$, and for all λ, we have

$$\Pr\left[\begin{array}{l}(\{f_i^{\langle 1\rangle}, \ldots, f_i^{\langle k\rangle}\}_{i=1}^n, \pi_{share}) \leftarrow \mathcal{A}(n,t),\\ \mathsf{Verify}(n,t,\{f_i^{\langle 1\rangle}, \ldots, f_i^{\langle k\rangle}\}_{i=1}^n, \pi_{share}) = \mathtt{true}:\\ \exists (f_0^{\langle 1\rangle}, \ldots, f_0^{\langle k\rangle})\ \forall\ V \subseteq [n], |V| \geq t+1,\\ \mathsf{Reconstruct}(\{f_i^{\langle 1\rangle}, \ldots, f_i^{\langle k\rangle}\}_{i\in V}, \pi_{share}) \rightarrow (f_0^{\langle 1\rangle}, \ldots, f_0^{\langle k\rangle})\end{array}\right] \geq 1 - \mathsf{negl}(\lambda)\ ,$$

  where V is the set of honest parties.
- **Unpredictability**: The protocol must be unpredictable, meaning that there is no strategy for selecting t sets of shares of the secrets that would enable someone to predict the secrets $(f_0^{\langle 1\rangle}, \ldots, f_0^{\langle k\rangle})$ with a significant advantage. More formally, for any integers $n > 1$ and $t < n$, a BVSS is called unpredictable if for all PPT adversaries $\mathcal{A}$ who have non-adaptively corrupted up to t parties, for all λ, and random secrets $(f_0^{\langle 1\rangle}, \ldots, f_0^{\langle k\rangle})$, we have:

$$\Pr\left[\begin{array}{l}(\{f_i^{\langle 1\rangle}, \ldots, f_i^{\langle k\rangle}\}_{i=1}^n, \pi_{share}) \leftarrow \mathsf{Share}(n,t,(f_0^{\langle 1\rangle}, \ldots, f_0^{\langle k\rangle}));\\ Q \subset [n]; |Q| \leq t; (f_0'^{\langle 1\rangle}, \ldots, f_0'^{\langle k\rangle}) \leftarrow \mathcal{A}(n,t,\{f_i^{\langle 1\rangle}, \ldots,\\ f_i^{\langle k\rangle}\}_{i\in Q}, \pi_{share}) : f_0'^{\langle 1\rangle} = f_0^{\langle 1\rangle} \vee \cdots \vee f_0'^{\langle k\rangle} = f_0^{\langle k\rangle}\end{array}\right] \leq \mathsf{negl}(\lambda)\ .$$

- **Simulatability:** In some cases, the definition of unpredictability can be strengthened by requiring that the view of the adversary who can corrupt up to t parties can be simulated.

We note that by setting $k = 1$, one can obtain the definitions for a typical VSS scheme [2, 5, 29, 30], where the dealer shares a single secret $f_0^{\langle 1\rangle}$.

3 A Unified Framework for Batched and Packed VSS

In this section, we introduce a tunable extension of the Π framework [5], referred to as (k,ℓ)-BΠ (or simply BΠ), which is designed to support batched and/or packed secret sharing in the synchronous setting. This generalization offers greater flexibility in terms of resilience and efficiently sharing multiple secrets, either by batching several independent sharings, packing multiple secrets into a single polynomial, or combining both approaches. Before presenting the details of the (k,ℓ)-BΠ protocol, we refer the reader to the full version of this paper for a brief overview of the original Π scheme, which corresponds to the special case $(1,1)$-BΠ.

3.1 (k, ℓ)-BΠ: k-Batched and ℓ-Packed Versions of Π

Intuitively, the Π framework can be seen as performing Shamir secret sharing twice, once to share the actual secret and once to prove its correctness, effectively sacrificing one sharing to verify the other. More generally, by embedding auxiliary randomness γ_i into the commitments, Π also supports the sharing of low-entropy secrets. Our investigation reveals that when a dealer wishes to share k independent secrets with the same group of participants, the proof π_{share} in Π can be efficiently batched. Moreover, the protocol can be adapted to support ℓ-packed secret sharing with minimal modifications. However, in the case of packing, we sacrifice the robustness of the protocol. On the positive side, packing can effectively double the number of shared secrets by only requiring one extra honest participant. In this k-batched and ℓ-packed setting, the dealer requires only a single additional polynomial to prove the correctness of k simultaneous ℓ-packed Shamir sharings. As a result, the cost of proving k parallel sharings becomes nearly equivalent to that of a single (packed) sharing, significantly reducing the overall overhead. In contrast to the original Π protocol, where each sharing is handled separately, BΠ allows the dealer to commit to k shares per participant simultaneously. Moreover, instead of using an individual masking polynomial per secret, the dealer uses a single randomizer polynomial to mask all k secret-sharing polynomials collectively when computing the polynomial $z(x)$. The general construction of (k, ℓ)-BΠ is illustrated in Fig. 2. It relies on a perfectly or computationally hiding vector commitment scheme $\mathcal{C}$, and the random oracle $\mathcal{RO} : \{0,1\}^\star \to \mathbb{Z}_q$ as a black-box component.

3.2 Security and Efficiency

Security. We prove the security of (k, ℓ)-BΠ in the following theorem.

Theorem 1 (k-Batched and ℓ-Packed Computational VSS). *If the commitment scheme $\mathcal{C}$ is computationally (resp. perfectly) hiding and perfectly (resp. computationally) binding, and $\mathcal{H}$ is a random oracle, then the generic construction given in Fig. 2 is a secure batched (and packed) VSS scheme in the synchronous setting for $n \geq 2t + \ell$. Specifically: (i) An honest dealer will convince honest shareholders, and any subset of $t + \ell$ honest shareholders will be able to reconstruct the secrets. (ii) The reconstruction protocol outputs a unique set of secrets distributed by the dealer for any qualified set of shareholders (i.e., of size at least $t + \ell$); (iii) Any set of up to t shareholders (i.e., a non-qualified set) cannot obtain any information about the secrets.*

Proof. Simulatability of the scheme follows from the hiding property of the commitment scheme $\mathcal{C}$ and the fact that any t secret shares from some polynomial $f^{\langle j \rangle}$ reveal nothing about the packed secrets $f^{\langle j \rangle}_{-(\ell-1)}, \ldots, f^{\langle j \rangle}_0$. In other words, one could simulate the view of the adversary with access to only the secret shares of t different parties $\{\tilde{f}_i^{\langle j \rangle}\}_{i=1,j=1}^{t,k}$. The simulator samples random degree $t + l - 1$ polynomials $\tilde{r}(X)$ and $\tilde{f}^{\langle j \rangle}(X)$ such that $\tilde{f}^{\langle j \rangle}(i) = \tilde{f}_i^{\langle j \rangle}$ for $j = 1, \ldots, k$

Initial: Parties $P_1, \cdots, P_n$ register a PK to facilitate secure communications and agree on public parameters, like random oracle $\mathcal{RO}$ and commitment scheme $\mathcal{C}$. Let there exist a dealer D and $P_1, \cdots, P_n$ parties who will receive the shares.

Share: Given (n, t), the public parameters of vector commitment, random oracle $\mathcal{H}$, to share $k \times \ell$ secrets $\{f_{-(\ell-1)}^{\langle j \rangle}, \cdots, f_0^{\langle j \rangle}\}_{j=1}^{k}$, the dealer D acts as below:

1. Sample $k+1$ uniformly random polynomials $f^{\langle 1 \rangle}(x), \ldots, f^{\langle k \rangle}(x)$ and $r(x)$ of degree $t+\ell-1$ with coefficients in a field $\mathbb{Z}_q$ (or ring $\mathbb{Z}_N$), subject to
$$f^{\langle j \rangle}(-(\ell-1)) = f_{-(\ell-1)}^{\langle j \rangle}, \ \ldots, \ f^{\langle j \rangle}(0) = f_0^{\langle j \rangle}, \quad \text{for} \quad j = 1, \ldots, k.$$
2. For $i = 1, \cdots, n$ and $j = 1, \ldots, k$: compute $f_i^{\langle j \rangle} := f^{\langle j \rangle}(i)$, and $r_i := r(i)$.
3. For $i = 1, \ldots, n$: sample $\gamma_i \leftarrow\!\!\$\ \mathbb{Z}_q$ and set $c_i = \mathcal{C}((f_i^{\langle 1 \rangle}, \cdots, f_i^{\langle k \rangle}, r_i), \gamma_i)$.
4. Compute $d_1, \ldots, d_k \leftarrow \mathcal{RO}(c_1, \ldots, c_n)$, where $\mathcal{RO}$ is a random oracle.
5. Set $z(x) = r(x) + \sum_{j=1}^{k} d_j f^{\langle j \rangle}(x)$ and $\pi_{share} := (c_1, \ldots, c_n, z(x))$.
6. Send shares $(\{f_i^{\langle j \rangle}\}_{j=1}^{k}, \gamma_i)$ securely to party P_i and publish π_{share}.

Verify: Given $\pi_{share} := (c_1, \ldots, c_n, z(x))$, and the individual shares:

1. Each P_i checks $\deg(z(x)) \leq t + \ell - 1$; if so, it proceeds as follows:
   (a) Computes $(d_1, \ldots, d_k) \leftarrow \mathcal{RO}(c_1, \ldots, c_n)$.
   (b) Uses her/his shares $(\{f_i^{\langle j \rangle}\}_{j=1}^{k}, \gamma_i)$, and checks whether
   $$c_i = \mathcal{C}((f_i^{\langle 1 \rangle}, \cdots, f_i^{\langle k \rangle}, z(i) - \textstyle\sum_{j=1}^{k} d_j f_i^{\langle j \rangle}), \gamma_i).$$
   If the verification of P_i fails, then P_i broadcasts a complaint against D.
2. If the number of shareholders complaining against the dealer exceeds t, the dealer will be disqualified, and the Verify will return `false`.
3. In case a shareholder P_i raises a complaint about the verification of their part, the dealer will broadcast $\{f_i^{\langle j \rangle} = f^{\langle j \rangle}(i)\}_{j=1}^{k}$ and the randomizer γ_i (if applicable) to enable everyone to verify it using the verification equation. If the verification passes, the protocol continues as usual. However, if it fails, the dealer will be disqualified, leading to a `false` verification outcome. Since the disqualification decision is solely based on the information broadcasted, all honest shareholders will ultimately reach a consensus either on a set of qualified parties $Q \subseteq \{P_1, P_2, \ldots, P_n\}$ or on rejecting the final verification.

Reconstruct: Given $\pi_{share} := (c_1, \ldots, c_n, z(x))$, each party P_i broadcasts the share values $\{f_i^{\langle j \rangle}\}_{j=1}^{k}$ and γ_i. A party P_i is said to be confirmed if
$$c_i = \mathcal{C}((f_i^{\langle 1 \rangle}, \cdots, f_i^{\langle k \rangle}, z(i) - \textstyle\sum_{j=1}^{k} d_j f_i^{\langle j \rangle}), \gamma_i),$$
where $(d_1, \ldots, d_k) \leftarrow \mathcal{RO}(c_1, \ldots, c_n)$. Consider $\{f_i^{\langle j \rangle}\}_{j=1}^{k}$ values of any $t + \ell$ confirmed parties and interpolate $f^{\langle 1 \rangle}(x), \cdots, f^{\langle k \rangle}(x)$ of degree $t + \ell - 1$ that passes through those points. Finally, the output is `false` (if $t + \ell$ valid sets of shares were not obtained) or
$$f^{\langle j \rangle}(-(\ell-1)) = f_{-(\ell-1)}^{\langle j \rangle}, \ \ldots, \ f^{\langle j \rangle}(0) = f_0^{\langle j \rangle}, \quad \text{for} \quad j = 1, \ldots, k.$$

Fig. 2. (k, ℓ)-BII: A unified framework for k-batched and ℓ-packed computational verifiable secret sharing in the synchronous setting for $n \geq 2t + \ell$.

and $i = 1, \ldots, t$. Then, it uses the hiding commitment scheme to compute $\tilde{c}_i = \mathcal{C}((\tilde{f}^{\langle 1 \rangle}(i), \cdots, \tilde{f}^{\langle k \rangle}(i), \tilde{r}(i)), \tilde{\gamma}_i)$ for $i = 1, \ldots, n$. Using the challenge values $\tilde{d}_1, \ldots, \tilde{d}_k$, the simulator computes a simulated $\tilde{z}(X) := \tilde{r}(X) + \sum_{j=1}^{k} \tilde{d}_j \tilde{f}^{\langle j \rangle}(X)$. The simulated transcript is constructed as $\tilde{\pi}_{Share} := (\tilde{c}_1, \ldots, \tilde{c}_n, \tilde{z}(X))$. Since the adversary could produce a perfect simulation of the public transcript, it reveals nothing about the secret shares $\{(f^{\langle j \rangle}_{-(\ell-1)}, \ldots, f^{\langle j \rangle}_0\}_{j=1}^{k}$ for up to t malicious parties. Next, we prove the scheme satisfies verifiability with unique reconstruction.

Since we assume a packing size ℓ and we have passed the verification stage, we can assume WLOG that at least the parties $P_1, \ldots, P_{t+l}$ are in possession of k secrets $\{f_i^{\langle j \rangle}\}_{i=1,j=1}^{t+\ell,k}$ such that $c_i = \mathcal{C}((\{f_i^{\langle j \rangle}\}_{j=1}^{k}, z(i) - \sum_{j=1}^{k} d_j f_i^{\langle j \rangle}), \gamma_i)$ for $i = 1, \ldots, t+\ell$, where $d_1, \ldots, d_k$ are computed using the RO and the public commitments c_i for $i = 1, \ldots, n$. These secrets can be used to construct the polynomials $f^{\langle 1 \rangle}(X), \ldots, f^{\langle k \rangle}(X)$, which define $f_0^{\langle 1 \rangle} = f^{\langle 1 \rangle}(0), \ldots, f_0^{\langle k \rangle} = f^{\langle k \rangle}(0)$. An honest party can use these polynomials to reconstruct the secrets of the other parties and verify whether $c_i = \mathcal{C}((\{f_i^{\langle j \rangle}\}_{j=1}^{k}, r_i), \gamma_i)$ for all $r_i = z(i) - \sum_{j=1}^{k} d_j f_i^{\langle j \rangle}$.

We show that at the end of the Reconstruction phase, the output is a unique set of values $F_0 = \{(f^{\langle j \rangle}_{-(\ell-1)}, \ldots, f_0^{\langle j \rangle})\}_{j=1}^{k}$ with probability greater than $1 - 1/q$. We denote the set of qualified parties remaining after verification phase as $\hat{P} = \{P_{\hat{i}_i}\}_{i=1}^{\hat{n}}$ for $\{\hat{i}_i\}_{i=1}^{\hat{n}} \subseteq [n]$ and $t + \ell \leq \hat{n} \leq n$. WLOG we can state that $\{P_i\}_{i=1}^{t+\ell} \in \hat{P}$. Assume that some set of parties P^* of size $t + \ell$ reconstructs $\hat{F}_0 = \{(\hat{f}^{\langle j \rangle}_{-(\ell-1)}, \ldots, \hat{f}_0^{\langle j \rangle})\}_{j=1}^{k} \neq F_0$ based on the values $\{\hat{f}^{\langle j \rangle}_{\hat{i}_i}\}_{j=1,i=1}^{k,t+\ell}$ for $\{\hat{i}_i\}_{i=1}^{t+1}$. Define an $(\hat{n}, t+\ell)$-Reed-Solomon code for the evaluation points $\hat{i}_1, \hat{i}_2, \ldots, \hat{i}_{\hat{n}}$ and a corresponding Vandermonde matrix

$$V = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ \hat{i}_1 & \hat{i}_2 & \ldots & \hat{i}_{\hat{n}} \\ \hat{i}_1^2 & \hat{i}_2^2 & \ldots & \hat{i}_{\hat{n}}^2 \\ \vdots & & \ddots & \vdots \\ \hat{i}_1^{\hat{n}-1} & \hat{i}_2^{\hat{n}-1} & \ldots & \hat{i}_{\hat{n}}^{\hat{n}-1} \end{bmatrix}.$$

Now define a check matrix C as the rightmost $\hat{n} - t - \ell$ columns of V^{-1}. Thereby a vector is an element of the code if and only if it lies in the nullspace of C.

By our assumption, $\hat{F}_0$ will differ from F_0 in at least one element, i.e.

$$(f^{\langle j \rangle}_{-(\ell-1)}, \ldots, f_0^{\langle j \rangle}) \neq (\hat{f}^{\langle j \rangle}_{-(\ell-1)}, \ldots, \hat{f}_0^{\langle j \rangle})$$

for some $j \in [k]$. Therefore, the sets $\{\hat{f}^{\langle j \rangle}_{\hat{i}_i}\}_{i=1}^{t+\ell}$ and $\{f_i^{\langle j \rangle}\}_{i=1}^{t+\ell}$ must differ in at least one value. This implies that the set $f^{\langle j \rangle} = \{f_1^{\langle j \rangle}, \ldots, f_{\hat{n}}^{\langle j \rangle}\}$ (of which both are a subset) is not an $(\hat{n}, t+\ell)$-Reed-Solomon codeword. In other words, $f^{\langle j \rangle} \cdot C \neq 0$. Therefore, we can state that for matrix $F = \{f_i^{\langle j \rangle}\}_{j=1,i=1}^{k,\hat{n}}$, the matrix $F \cdot C$ is a non-zero matrix. On the other hand, since the reconstruction has not resulted in false and we assume the commitment scheme is binding, we can also state that for $\boldsymbol{z} = \{z(\hat{i}_i)\}_{i=1}^{\hat{n}}$ and $\boldsymbol{d} = \{d_1, \ldots, d_k\}$, the vector $\boldsymbol{z} - \boldsymbol{d} \cdot \hat{F}$ (whose values all

lie on $\hat{r}(x)$) must be an $(\hat{n}, t+\ell)$-Reed-Solomon codeword. This implies that

$$(\boldsymbol{z} - \boldsymbol{d} \cdot F) \cdot C = \boldsymbol{z} \cdot C - \boldsymbol{d} \cdot F \cdot C = 0.$$

For a random $\boldsymbol{d} = \{d_1, \ldots, d_k\}$ we bound the probability that the above holds. Since $\mathbb{F} \cdot C$ is a non-zero matrix, it contains a non-zero column $\boldsymbol{a}$ of length k. So there must be an element of $\boldsymbol{z} \cdot C$, which we note as b, such that

$$b - \langle \boldsymbol{d}, \boldsymbol{a} \rangle = 0,$$

The Schwartz-Zippel lemma states that this holds with probability at most $1/q$. Note that for $k = 1$ and $l = 1$, the protocol in Figure 2 reduces to the protocol introduced by Baghery [5]. In this case, the proof reduces to the contradiction that $F = \{f_i^{\langle 1 \rangle}\}_{i=1}^{\hat{n}}$ is not a codeword while $\boldsymbol{z} - d \cdot \boldsymbol{F}$ is a codeword in the $(\hat{n}, t+1)$-Reed-Solomon code. This clearly contradicts the linearity of Reed-Solomon codes. □

Efficiency of (k, ℓ)-BΠ. The (k, ℓ)-BΠ protocol significantly improves efficiency over the original Π (equivalent to $(1, 1)$-BΠ) when proving the validity of $k \times \ell$ secrets. During the sharing phase, the dealer computes evaluations for k independent degree-$(t+\ell)$ polynomials $f^{\langle 1 \rangle}(x), \ldots, f^{\langle k \rangle}(x)$, each requiring n evaluations, and a single masking polynomial $r(x)$ of degree $t + \ell$ with n evaluations. This totals $n(k+1)$ polynomial evaluations, compared to the $2n(k+\ell)$ evaluations required for $k + \ell$ independent executions of Π. The process further involves generating only n commitments (one per participant, each encoding k shares) instead of $n(k+\ell)$ commitments. To construct the proof π_{share}, the dealer combines all k secret-sharing polynomials with $r(x)$ via the equation

$$z(x) = r(x) + \textstyle\sum_{j=1}^{k} d_j f^{\langle j \rangle}(x),$$

where $d_1, \ldots, d_k$ are derived from a single query to the random oracle $\mathcal{RO}$. This requires fewer arithmetic operations and fewer queries to $\mathcal{RO}$ compared to $k+l$ separate executions of Π. Note that, in practice we can set the random oracle to return just a single random element d, and then set $d_j = d^j$ for $j = 1, \ldots, k$.

During verification, each party evaluates the degree-$(t+\ell)$ polynomial $z(x)$ and verifies its consistency with the batched commitments. This step retains the computational profile of the original protocol: each party performs one polynomial evaluation (over a degree-$(t+\ell)$ polynomial), one random oracle query, and one commitment check, regardless of k. Crucially, the communication overhead is reduced: the dealer broadcasts n commitments (instead of $n(k+\ell)$) and the $t + \ell + 1$ coefficients of $z(x)$ (instead of $(k+\ell)(t+1)$), while securely sending each participant $k + \ell$ shares, as in $k + \ell$ executions of the original protocol.

By batching proofs and leveraging ℓ-packed secret sharing, BΠ achieves nearly optimal amortized complexity; the cost per secret shrinks to $\mathcal{O}(1/k)$ for large k, while preserving the security guarantees of the base Π protocol. It is important to note that the increased polynomial degree $t + \ell$ (compared to t in Π) reflects the ℓ-packed secret-sharing paradigm but does not asymptotically dominate the efficiency gains from batching. □

4 Practical Batched and Packed VSS Protocols

In this section, we instantiate the general (k, ℓ)-BΠ framework of Sec. 3 with several vector commitment schemes, obtaining the BP-VSS schemes $\mathrm{B}\Pi_{\mathrm{LA}}$, $\mathrm{B}\Pi_{\mathrm{P}}$, $\mathrm{B}\Pi_{\mathrm{P}}+$, and $\mathrm{B}\Pi_{\mathrm{F}}$. These schemes offer different security–efficiency trade-offs.

All proposed BP-VSS schemes operate in the synchronous setting for $n \geq 2t + \ell$, where n is the number of parties, t the number of corruptions, and ℓ the packing parameter. For $\ell = 1$, i.e., the batched non-packed case, they achieve optimal resilience. Larger values of ℓ improve efficiency but reduce the tolerated corruption threshold by about $\ell/2$ on average and sacrifice robustness. Thus, the parameters (k, ℓ) can be tuned according to the number of parties, the number of secrets, and the desired robustness level; combining batching and packing gives the best overall efficiency compared to either technique alone.

4.1 $\mathrm{B}\Pi_{\mathrm{LA}}$: Batched and Packed Version of Π_{LA}

Our first instantiation, $\mathrm{B}\Pi_{\mathrm{LA}}$, uses a hash-based vector commitment with computational hiding. It yields a batched, packed, lightweight, and plausibly post-quantum-secure VSS scheme in the synchronous setting. Commitments are computed as $c_i := \mathsf{H}(\boldsymbol{m}, \gamma_i)$, where $\boldsymbol{m}$ is the message vector and γ_i is the randomizer. Below, we specify only the deviations from the general BΠ scheme in Fig. 2.

Initial: Parties agree on a random oracle $\mathcal{RO}$ and a commitment scheme $\mathcal{H}$.

Share: Given (n, t), the public parameters, and secrets $\{f^{\langle j \rangle}_{-(\ell-1)}, \ldots, f^{\langle j \rangle}_0\}_{j=1}^k$, the dealer D follows Fig. 2, except that in Step 3, for each $i = 1, \ldots, n$, it samples $\gamma_i \leftarrow\!\!\$\, \mathbb{Z}_q$ and sets $c_i = \mathcal{H}\big((f_i^{\langle 1 \rangle}, \ldots, f_i^{\langle k \rangle}, r_i), \gamma_i\big)$.

Verify: Given $\pi_{share} := (c_1, \ldots, c_n, z(x))$ and its individual shares, each party first checks that $\deg(z(x)) \leq t + \ell - 1$. If so, it computes $(d_1, \ldots, d_k) \leftarrow \mathcal{RO}(c_1, \ldots, c_n)$ and verifies

$$c_i = \mathcal{H}\Big((f_i^{\langle 1 \rangle}, \ldots, f_i^{\langle k \rangle}, z(i) - \textstyle\sum_{j=1}^k d_j f_i^{\langle j \rangle}), \gamma_i\Big).$$

If this check fails, P_i broadcasts a complaint against D.

Reconstruct: The reconstruction phase is identical to Fig. 2, except that parties confirm opened shares using the hash-based commitment check defined in the Verify phase. Interpolation and output computation are unchanged.

Security. By Theorem 1, $\mathrm{B}\Pi_{\mathrm{LA}}$ satisfies Completeness, Verifiability, and Simulatability, as defined in Def. 1, against computationally bounded adversaries.

4.2 $\mathrm{B}\Pi_{\mathrm{P}}$: Batched and Packed Version of Π_{P}

Our second instantiation, $\mathrm{B}\Pi_{\mathrm{P}}$, uses the generalized vector Pedersen commitment [11, 29], which is perfectly hiding and additively homomorphic. It yields a DL-based batched and packed VSS scheme with information-theoretic privacy.

Given random generators $(g_1, \ldots, g_k, g_{k+1}, g_{k+2})$, commitments are computed as $c_i := g_1^{f_i^{\langle 1 \rangle}} \cdots g_k^{f_i^{\langle k \rangle}} g_{k+1}^{r_i} g_{k+2}^{\gamma_i}$, where $\gamma_i \leftarrow\!\!\$\ \mathbb{Z}_q$. Below, we specify only the deviations from the general framework in Fig. 2.

Initial: Parties agree on $\mathcal{RO}$ and group generators $(g_1, \ldots, g_{k+2})$.

Share: Same as Fig. 2, except that in Step 3, for each $i = 1, \ldots, n$, the dealer samples $\gamma_i \leftarrow\!\!\$\ \mathbb{Z}_q$ and sets $c_i = g_1^{f_i^{\langle 1 \rangle}} \cdots g_k^{f_i^{\langle k \rangle}} g_{k+1}^{r_i} g_{k+2}^{\gamma_i}$.

Verify: Given $\pi_{share} := (c_1, \ldots, c_n, z(x))$, each party first checks that $\deg(z(x)) \leq t + \ell - 1$. If so, it computes $(d_1, \ldots, d_k) \leftarrow \mathcal{RO}(c_1, \ldots, c_n)$ and checks

$$c_i = g_1^{f_i^{\langle 1 \rangle}} \cdots g_k^{f_i^{\langle k \rangle}} g_{k+1}^{z(i) - \sum_{j=1}^{k} d_j f_i^{\langle j \rangle}} g_{k+2}^{\gamma_i}.$$

If the check fails, P_i broadcasts a complaint against D.

Reconstruct: The reconstruction phase follows Fig. 2. The only deviation is that an opened share is confirmed iff it satisfies the Pedersen-based commitment check of the Verify phase; interpolation and output are unchanged.

Security. By Theorem 1, $\text{BI}\Pi_\text{P}$ satisfies Completeness, Verifiability, and perfect Simulatability, as defined in Definition 1 (generalized for $\ell > 1$). In particular, transcript simulation is perfect, so even an unbounded adversary corrupting up to t parties obtains no information about the secrets.

4.3 An Optimized Vector Commitment for $\text{BI}\Pi_\text{P}$ in $\text{BI}\Pi_\text{P}+$

The scheme $\text{BI}\Pi_\text{P}$ inherits perfect simulatability from Pedersen commitments, which is useful in applications such as unbiased public-key generation [24] and distributed randomness generation. However, its generalized vector Pedersen commitment requires $k(n+2)$ exponentiations in sharing and $k+2$ exponentiations in verification. We therefore introduce a more efficient perfectly hiding vector commitment that combines hashing with Pedersen commitments. The scheme loses additive homomorphism but supports re-randomization and reduces the commitment cost to two exponentiations and one hash evaluation.

An Efficient Perfectly Hiding Non-Homomorphic Commitment. Let $\mathsf{VC} = (\mathsf{Setup}, \mathcal{C}, \mathsf{Verify})$ be a vector commitment over $\mathbb{Z}_q$.

1. $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$. Samples a hash function $\mathcal{H} : \{0,1\}^\star \to \mathbb{Z}_q$ and two random generators (g_1, g_2), and outputs $\mathsf{pp} = (\mathcal{H}, g_1, g_2)$.
2. $\mathcal{C}(m_1, \ldots, m_k; r) \to \mathsf{c}$. Outputs a commitment $\mathsf{c} = g_1^{\mathcal{H}(m_1, \ldots, m_k)} g_2^r$.
3. $\mathsf{Verify}(\mathsf{c}, m_1, \ldots, m_k, r) \to \{0,1\}$. Accepts iff $\mathsf{c} = g_1^{\mathcal{H}(m_1, \ldots, m_k)} g_2^r$.

Theorem 2 (Efficient Perfectly Hiding Vector Commitment). *Let $\mathcal{H} : \{0,1\}^\star \to \mathbb{Z}_q$ be a random oracle. The vector commitment* VC *defined above satisfies completeness, perfect hiding, and computational binding under the DL assumption and the collision resistance of $\mathcal{H}$.*

Proof. The proof is given in the full version.

Efficiency. Compared with generalized vector Pedersen commitments, the above commitment requires only two exponentiations and one hash evaluation in both sharing and verification.

$\mathrm{B}\Pi_{\mathrm{P}}+$: An Optimized Batched and Packed Version of Π_{P}. Instantiating (k, ℓ)-BΠ with the above commitment gives $\mathrm{B}\Pi_{\mathrm{P}}+$, an optimized variant of $\mathrm{B}\Pi_{\mathrm{P}}$. Compared with $\mathrm{B}\Pi_{\mathrm{P}}$, it reduces the number of exponentiations by a factor of k in both sharing and verification, at the cost of using a non-homomorphic commitment. It nevertheless remains suitable for applications such as DKG [24].

The algorithms are the same as in $\mathrm{B}\Pi_{\mathrm{P}}$, with the following changes.

Initial: Parties agree on $\mathcal{RO}$, $\mathcal{H} : \{0,1\}^\star \to \mathbb{Z}_q$, and generators (g_1, g_2).

Share: For each i, the commitment is $c_i = g_1^{\mathcal{H}(f_i^{\langle 1\rangle}, \ldots, f_i^{\langle k\rangle}, r_i)} g_2^{\gamma_i}$.

Verify: Same as $\mathrm{B}\Pi_{\mathrm{P}}$, except that parties verify using the above hash-and-DL commitment; namely, r_i is replaced by $z(i) - \sum_{j=1}^{k} d_j f_i^{\langle j\rangle}$ inside the hash.

Reconstruct: Same as $\mathrm{B}\Pi_{\mathrm{P}}$, except that opened shares are confirmed using the hash-and-DL commitment check of the Verify phase.

Security. By Theorems 1 and 2, $\mathrm{B}\Pi_{\mathrm{P}}+$ satisfies Completeness, Verifiability, and perfect Simulatability.

4.4 $\mathrm{B}\Pi_{\mathrm{F}}$: Batched and Packed VSS for High-Entropy Secrets

Finally, $\mathrm{B}\Pi_{\mathrm{F}}$ is a batched and packed extension of Π_{F} [5]. It is obtained from $\mathrm{B}\Pi_{\mathrm{P}}$ by omitting the randomizer γ_i and using commitments

$$c_i := g_1^{f_i^{\langle 1\rangle}} \cdots g_k^{f_i^{\langle k\rangle}} g_{k+1}^{r_i}.$$

As in Feldman VSS [22] and Π_{F}, this scheme requires high-entropy secrets and provides unpredictability rather than the simulatability guaranteed by $\mathrm{B}\Pi_{\mathrm{LA}}$, $\mathrm{B}\Pi_{\mathrm{P}}$, and $\mathrm{B}\Pi_{\mathrm{P}}+$. Although this weakens privacy, $\mathrm{B}\Pi_{\mathrm{F}}$ is efficient and remains useful in distributed protocols where the shared values are high entropy, such as DKG [24]. Equivalently, the algorithms of $\mathrm{B}\Pi_{\mathrm{F}}$ are obtained from those of $\mathrm{B}\Pi_{\mathrm{P}}$ in Sec. 4.2 by removing all terms involving γ_i and g_{k+2}.

Security. We formalize the security of $\mathrm{B}\Pi_{\mathrm{F}}$ below.

Theorem 3 (Batched and Packed VSS from DL). *Let $(g_1, \ldots, g_{k+1})$ be $k+1$ distinct random generators of a group $\mathbb{G}$, and let $\mathcal{H}$ be a random oracle. Under the DL assumption, $\mathrm{B}\Pi_{\mathrm{F}}$ is a secure batched and packed VSS protocol in the synchronous setting for $n \geq 2t + \ell$. Specifically, it satisfies Completeness, Verifiability, and Unpredictability, as defined in Definition 1.*

Proof. The proof is given in the full version.

5 An Efficient Batched and Packed PVSS

In all proposed batched-and-packed VSS schemes in Section 4, share correctness can only be verified by shareholders. In this section, we propose a Batched-and-Packed Publicly Verifiable Secret Sharing (BP-PVSS) scheme based on the BII_{F} BP-VSS construction. The new BP-PVSS protocol BII_{S} can be viewed as a batched variant of the PVSS scheme Π_{S} proposed in [5,14].

To achieve this, we first develop a batched version of the sigma protocol π_{PDL} for the Polynomial Discrete Logarithm (PDL) problem (Definition in full version), recently introduced by Baghery [5] and employed in Π_{S} PVSS.

5.1 Batched Version of Baghery's π_{PDL} Protocol for PDL

This section presents a batched variant of the π_{PDL} protocol from [5, Sec. 5]. The original protocol extends Schnorr's identification scheme to polynomial domains. While π_{PDL} enables knowledge proofs for k degree-t polynomials, this requires k protocol executions - an inefficient approach. To overcome this limitation, we propose a batched π_{PDL} protocol allowing a prover to demonstrate witness knowledge for the relation $R_{\text{PDL}}^{\text{Mul}}$ defined as:

$$R_{\text{PDL}}^{\text{Mul}} = \left\{ \left(g, \{\alpha_i, \{F_i^{\langle j \rangle}\}_{j=1}^{k}\}_{i=1}^{n}\right), \{f^{\langle j \rangle}(x)\}_{j=1}^{k} \;\middle|\; \begin{array}{l} F_1^{\langle j \rangle} = g^{f^{\langle j \rangle}(\alpha_1)}, \ldots, F_n^{\langle j \rangle} = \\ g^{f^{\langle j \rangle}(\alpha_n)}, \quad \text{for } j = 1, \ldots, k \end{array} \right\},$$

where $\{f^{\langle 1 \rangle}(x), \ldots, f^{\langle k \rangle}(x)\} \in \mathbb{Z}_q[x]_t$ are witness polynomials of degree at most $t \leq n-1$ with coefficients in $\mathbb{Z}_q$, and $\alpha_1, \ldots, \alpha_n$ are n distinct elements from $\mathbb{Z}_q$.

Fig. 3 describes our proposed batched version of π_{PDL} protocol, which enables a prover to generate a NIZK proof of knowledge for the relation $R_{\text{PDL}}^{\text{Mul}}$.

Theorem 4 (A NIZK Proof of Knowledge for $R_{\text{PDL}}^{\text{Mul}}$). *Let g be the generator of $\mathbb{G}$, $\{F_1^{\langle j \rangle}, \ldots, F_n^{\langle j \rangle}\}_{j=1}^{k} \in \mathbb{G}^{n \times k}$, $\{\alpha_i\}_{i=1}^{n}$ be n distinct elements from $\mathbb{Z}_q$, and t be the (maximum) degree of witness polynomials $f^{\langle 1 \rangle}(x), \ldots, f^{\langle k \rangle}(x)$. Assuming PDL is hard, for $0 \leq t < n$, the protocol π_{BPDL} (given in Fig. 3) is a NIZK PoK for R_{PDL}^{Mul} in the RO model.*

Proof. The proof is given in the full version.

5.2 BII_{S}: An Efficient Batched and Packed PVSS

Given the NIZK argument π_{BPDL}, described in Fig. 3, next we construct BII_{S}, that can be considered as a batched version of the PVSS scheme Π_{S}. In a typical PVSS scheme, the dealer encrypts each share using the public key of the receiving party and also attaches a non-interactive publicly verifiable ZK proof to prove the correctness of the encrypted shares. In our case, we use our proposed batched NIZK argument to give a batched proof for k parallel and packed executions of the protocol. Intuitively, the proposed BP-PVSS scheme uses the homomorphic property of the encryption to perform BII-like verification in the exponent.

Prover: Given the statement $(g, \alpha_1, \ldots, \alpha_n, F_1^{\langle j \rangle}, \ldots, F_n^{\langle j \rangle})_{j=1}^k$ and the witness polynomials $\{f^{\langle j \rangle}(x)\}_{j=1}^k$, proceed as follows and output a NIZK proof π.

1. Sample a degree-t polynomial $r(x) \in \mathbb{Z}_q[x]_t$; Set $\{\Gamma_i = g^{r(\alpha_i)}\}_{i=1}^n$.
2. Set $d \leftarrow \mathcal{H}(\{F_1^{\langle j \rangle}, \ldots, F_n^{\langle j \rangle}\}_{j=1}^k, \Gamma_1, \ldots, \Gamma_n)$, where $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q$ a RO.
3. Set $z(x) = r(x) + \sum_{j=1}^k d^j f^{\langle j \rangle}(x) \pmod q$;
4. Return $\pi := (\Gamma_1, \ldots, \Gamma_n, z(x))$ or $\pi := (d, z(x))$

Verifier: Given the statement $(g, \alpha_1, \ldots, \alpha_n, F_1^{\langle j \rangle}, \ldots, F_n^{\langle j \rangle})_{j=1}^k$ and $\pi := (\Gamma_1, \cdots, \Gamma_n, z(x))$, the verifier first checks if $z(x)$ is a degree-t polynomial. If so, then sets $d \leftarrow \mathcal{H}(\{F_1^{\langle j \rangle}, \ldots, F_n^{\langle j \rangle}\}_{j=1}^k, \Gamma_1, \ldots, \Gamma_n)$ and checks if: $g^{z(\alpha_i)} = \Gamma_i \prod_{j=1}^k (F_i^{\langle j \rangle})^{d^j}$ for $i = 1, \ldots, n$, and outputs `true` or `false`. Note that to make the communication shorter, alternatively, the prover could publish $\pi := (d, z(x))$, and then the verifier would need to check if $z(x)$ is a degree-t polynomial and $d = \mathcal{H}(\{F_1^{\langle j \rangle}, \ldots, F_n^{\langle j \rangle}\}_{j=1}^k, \frac{g^{z(\alpha_1)}}{\prod_{j=1}^k (F_1^{\langle j \rangle})^{d^j}}, \ldots, \frac{g^{z(\alpha_n)}}{\prod_{j=1}^k (F_n^{\langle j \rangle})^{d^j}})$.

Fig. 3. π_{BPDL}: Our proposed batched version of π_{PDL}.

Construction. Given a group generator g of group $\mathbb{G}$, party P_i samples $s_i \leftarrow \mathbb{Z}_q$ and registers the public key $\mathsf{pk}_i = g^{s_i}$. We also assume that each party has given a valid proof of knowledge of their secret key. These public keys are known to all parties and can be reused. Note that similar to k executions of ℓ-packed Π_{S}, at the end of the reconstruction phase, each party P_i will obtain k distinct shares $(g^{f^{\langle 1 \rangle}(i)}, \ldots, g^{f^{\langle k \rangle}(i)})$ and later in the reconstruction phase, any qualified set of parties will be able to reconstruct all the $k \times \ell$ secrets $\{g^{f^{\langle j \rangle}_{-(\ell-1)}}, \ldots, g^{f_0^{\langle j \rangle}}\}_{j=1}^k$. More precisely, in the reconstruction phase, each party broadcasts the decryption of their k shares. To maintain security, this should be accompanied with a proof of valid decryption. This we achieve by using the NIZK proof of knowledge proposed by Chaum, Evertse, and Graaf [18] (summarized in the full version).

We summarize the algorithms of $\mathrm{B}\Pi_{\mathrm{S}}$ below. In the sharing phase, we focus only on the steps deviating from the general BΠ scheme.

Initial: Parties agree on a random oracle $\mathcal{RO}$, group generator g, register public keys $\mathsf{pk}_i = g^{s_i} i = 1^n$, and store secret keys $s_i i = 1^n$.

Share: Given (n, t), public keys $\{\mathsf{pk}_i\}_{i=1}^n$, and secrets $\{f^{\langle j \rangle}_{-(\ell-1)}, \ldots, f_0^{\langle j \rangle}\}_{j=1}^k$, the dealer D proceeds:

1. Sample k random polynomials $f^{\langle 1 \rangle}(x), \ldots, f^{\langle k \rangle}(x)$ of degree $t+\ell-1$ with coefficients in $\mathbb{Z}_q$, satisfying $f^{\langle j \rangle}(-(\ell-1)) = f^{\langle j \rangle}_{-(\ell-1)}, \ldots, f^{\langle j \rangle}(0) = f_0^{\langle j \rangle}$.
2. For $i = 1, \ldots, n$ and $j = 1, \ldots, k$: compute encrypted shares $y_i^{\langle j \rangle} = \mathsf{pk}_i^{f_i^{\langle j \rangle}} := \mathsf{pk}_i^{f^{\langle j \rangle}(i)}$.
3. Run a modified π_{BPDL} (Fig. 3) to generate a batch proof π_{share}:
   - Sample random polynomial $r(x)$ of degree $t+\ell-1$ and set $\Gamma_i = \mathsf{pk}_i^{r(i)}$.
   - Compute $d \leftarrow \mathcal{RO}(\{y_1^{\langle j \rangle}, \ldots, y_n^{\langle j \rangle}\}_{j=1}^k, \Gamma_1, \ldots, \Gamma_n)$ and set $z(x) = r(x) + \sum_{j=1}^k d^j f^{\langle j \rangle}(x)$.

- Set $\pi_{share} := (d, z(x))$.

4. Publish encrypted shares $\{y_1^{\langle j \rangle}, \ldots, y_n^{\langle j \rangle}\}_{j=1}^{k}$ and π_{share}.

Verify: Given $\{\mathsf{pk}_i\}_{i=1}^{n}$, ciphertexts $\{y_1^{\langle j \rangle}, \ldots, y_n^{\langle j \rangle}\}_{j=1}^{k}$, and $\pi_{share} = (d, z(x))$, a verifier checks that $z(x)$ is degree $\leq t + \ell - 1$ and

$$d = \mathcal{H}(\{y_1^{\langle j \rangle}, \ldots, y_n^{\langle j \rangle}\}_{j=1}^{k}, \frac{\mathsf{pk}_1^{z(1)}}{\prod_{j=1}^{k}(y_1^{\langle j \rangle})^{d^j}}, \ldots, \frac{\mathsf{pk}_n^{z(n)}}{\prod_{j=1}^{k}(y_n^{\langle j \rangle})^{d^j}}),$$

returning `true` or `false`.

Reconstruction: Parties proceed as follows:

1. Each party decrypts shares $\{F_i^{\langle j \rangle} := g^{f_i^{\langle j \rangle}}\}_{j=1}^{k}$ via $F_i^{\langle j \rangle} = (y_i^{\langle j \rangle})^{1/s_i}$ and publishes them with a NIZK proof that $(\mathsf{pk}_i = g^{s_i}) \wedge_{j=1}^{k} (y_i^{\langle j \rangle} = (F_i^{\langle j \rangle})^{s_i})$.
2. For each $j \in [k]$, given $t+\ell$ valid $F_i^{\langle j \rangle}$, the secrets $g^{f^{\langle j \rangle}(-(\ell-1))}, \ldots, g^{f^{\langle j \rangle}(0)}$ are recovered via Lagrange interpolation: $\prod_{i=1}^{t+\ell}(F_i^{\langle j \rangle})^{\lambda_i} = g^{f^{\langle j \rangle}(0)} = g^{f_0^{\langle j \rangle}}$, with $\lambda_i^{\langle j \rangle} = \prod_{m=1, m \neq i}^{t+\ell} \frac{m}{m-i}$.

An analysis of the efficiency of $\mathrm{BII_S}$ and its non-batched variant is provided in Table 2. We prove the security of new scheme according to the definition used in [5, 30]. The security of the proposed scheme is inherited from the original constructions in [5, 14], combined with the batched and packed NIZK argument introduced in Fig. 3, whose security guarantees are established in this work. For completeness, we rigorously restate the security arguments in the theorem below.

Theorem 5 (Security of BP-PVSS Scheme $\mathrm{BII_S}$). *Under the PDL and Decisional Diffie-Hellman (DDH) assumptions, the VSS scheme $\mathrm{BII_S}$ described above, is a secure PVSS scheme against a static adversary in the random oracle model. That is, (i) the Reconstruct protocol results in the secret distributed by the dealer for any qualified set of shareholders, (ii) any non-qualified set of shareholders is unable to recover any (partial) information on the secret.*

Proof. The proof is given in the full version.

6 Applications and Implementation Results

6.1 Applications of BP-VSS and BP-PVSS Schemes

The proposed BP-VSS and BP-PVSS schemes can be deployed in a broad range of distributed protocols. The tunable parameters (k, ℓ) allow practitioners to balance the number of secrets shared, fault tolerance, and computational cost: setting $\ell = 1$ preserves optimal resilience while amortizing verification cost across k secrets, whereas $\ell > 1$ trades a modest reduction in fault tolerance (by $\lfloor \ell/2 \rfloor$ corruptions on average) for improved efficiency. Table 3 summarizes the main properties of our proposed schemes and highlights some of representative applications, which we discuss below.

Table 3. Properties and potential applications of the proposed BP-VSS and BP-PVSS schemes. IT: Information-Theoretic privacy; Comp.: Computational privacy; PQ: Post-Quantum security; Entropy: Secret's Entropy; Hom.: Additive Homomorphism of the underlying commitment; ✓: supported; ×: not supported.

Scheme	Privacy	PQ	Entropy	Hom.	Example Applications
$\mathsf{B\Pi_{LA}}$	Comp.	✓	Any	×	PQ-Secure Protocols, MPC, Federated Learning, ...
$\mathsf{B\Pi_P}$	IT	×	Any	✓	Unbiased DKG [24], Sharing low-entropy secrets, ...
$\mathsf{B\Pi_P^+}$	IT	×	Any	×	Unbiased DKG, non-homomorphic variant of $\mathrm{B\Pi_P}$, ...
$\mathsf{B\Pi_F}$	Comp.	×	High	✓	DKG, threshold signatures, alternative to [22], ...
$\mathsf{B\Pi_S}$	Comp.	×	High	✓	Distributed randomness generation, e-voting, ...

Notably, $\mathrm{B\Pi_{LA}}$ is the only scheme in our suite achieving post-quantum security, relying solely on hash functions in the RO model. This makes it particularly attractive for threshold protocols built on post-quantum assumptions. Particularly, in CSIDH-based protocols [17], such as DKGs and threshold signatures [1,2,7], parties must verifiably share large numbers of secrets (e.g., $k > 2^{10}$ in CSI-FiSh [8]). Repeated invocations of standard VSS for k secrets impose prohibitive overheads, making our batched approach essential for practical PQ deployments. By enabling simultaneous distribution and verification of multiple secrets, our framework addresses a critical performance bottleneck in real-world post-quantum secure threshold protocols. Its lightweight design also makes it attractive for large-scale MPC and Byzantine-resilient federated learning [33], where computational overhead per secret must be minimal.

Both VSS schemes $\mathrm{B\Pi_P}$ and $\mathrm{B\Pi_P}+$ are more efficient batched and packed alternatives to Pedersen VSS [29], preserving its IT privacy guarantee. They can be used in many of traditional Pedersen VSS applications: DKG protocols requiring uniformly distributed public keys [24], threshold decryption and signatures protecting long-term secrets. The distinction between the two lies in the underlying commitment: $\mathrm{B\Pi_P}$ retains additive homomorphism, while $\mathrm{B\Pi_P}+$ replaces it with the more efficient commitment of Theorem 2, achieving a factor-k reduction in exponentiations at the cost of losing homomorphism. Thus $\mathrm{B\Pi_P}+$ is the preferred choice whenever homomorphism is not required by the application.

$\mathrm{B\Pi_F}$ is an efficient batched and packed alternative to Feldman VSS [22], offering the same computational privacy under the DL assumption but reducing broadcast communication by a factor of k and verification from $O(kn)$ to $O(k+n)$ exponentiations over k sequential Feldman executions. Like Feldman VSS, it requires high-entropy secrets, which is naturally satisfied in DKG protocols with randomly sampled contributions [24] and threshold ECDSA or Schnorr signatures where shared values are random nonces. Beyond its standalone use, $\mathrm{B\Pi_F}$ serves as the algebraic foundation for our BP-PVSS scheme $\mathrm{B\Pi_S}$.

$\mathrm{B\Pi_S}$ is the first batched and packed PVSS scheme, extending public verifiability to the batched-and-packed setting. This makes it particularly useful for applications that require many publicly auditable sharing instances. One of its key application is in distributed randomness generation: in Sec. 7, we revisit ALBATROSS [14] and show that replacing its packed Π_S layer with $\mathrm{B\Pi_S}$ yields

ALBATROSS+, which reduces the amortized proof size asymptotically and improves the computational cost by approximately 50%. Beyond randomness generation, $\mathrm{B}\Pi_{\mathrm{S}}$ is applicable wherever public auditability of share distribution is required, including e-voting and threshold public key infrastructure [30].

6.2 Implementation Result

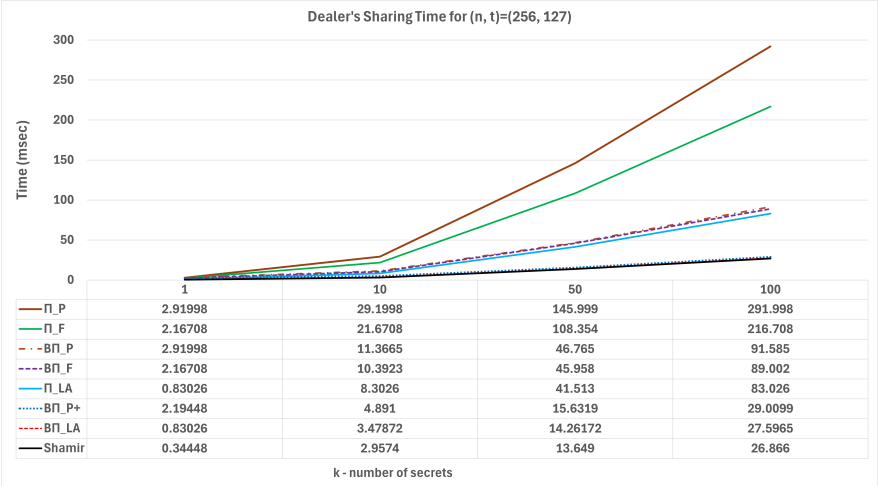
We analysed the asymptotic costs of our BP-VSS schemes - $\mathrm{B}\Pi_{\mathrm{LA}}$, $\mathrm{B}\Pi_{\mathrm{P}}$, $\mathrm{B}\Pi_{\mathrm{P}}+$, $\mathrm{B}\Pi_{\mathrm{F}}$, BP-Feldman, BP-Pedersen - and the BP-PVSS scheme $\mathrm{B}\Pi_{\mathrm{S}}$, comparing them with multiple executions of their non-batched versions from [5, 22, 29]. The descriptions of BP-Feldman, BP-Pedersen are presented in the full version. Detailed communication and computation costs are in Tables 1 and 2.

We report implementation results under various parameters based on a prototype Rust implementation. Experiments ran on a MacBook Pro with M4 Pro CPU and 24GB RAM, applying identical optimizations across all protocols. Non-batched variants were also benchmarked for reference. Each data point is averaged over 100 runs. We focus on $\ell = 1$ (non-packed); for $\ell > 1$, performance should scale proportionally, making the packed-and-batched schemes even faster.

Performance of New VSS Schemes. Our implementation results of Shamir [31] and VSS schemes (Π_{F}, Π_{P}, Π_{LA}) [5], as well as the new BP-VSS schemes ($\mathrm{B}\Pi_{\mathrm{F}}$, $\mathrm{B}\Pi_{\mathrm{P}}$, $\mathrm{B}\Pi_{\mathrm{P}}+$, $\mathrm{B}\Pi_{\mathrm{LA}}$), are summarized in the plots presented in Fig. 4, 5, and 6. Our evaluation covers key metrics like sharing time, verification time, and communication cost. All results are measured in the optimistic ("happy path") setting, where no one complains. This allows us to isolate the efficiency improvements offered by batching while preserving comparability across schemes.

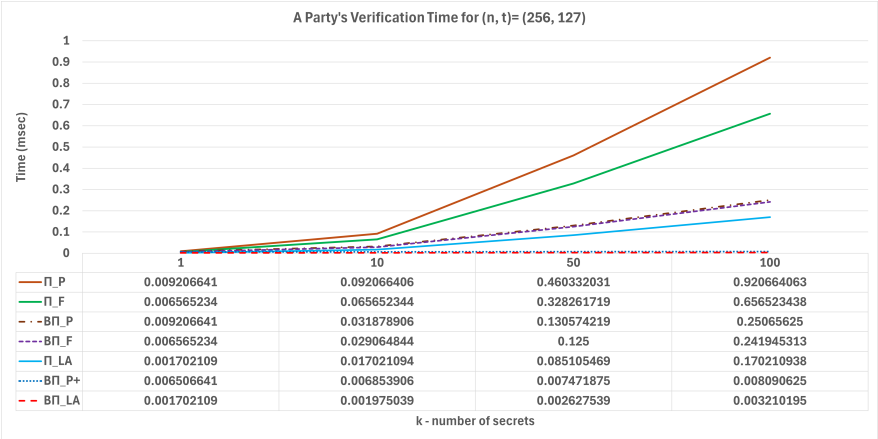
Sharing Time. The plots in Fig. 4 reports the sharing time of the proposed BP-VSS schemes. We compare these results against their basic counterparts from [5] as well as the original (non-verifiable) Shamir secret sharing [31]. This comparison allows us to quantify the additional cost introduced by verifiability. As can be seen, both $\mathrm{B}\Pi_{\mathrm{LA}}$ and $\mathrm{B}\Pi_{\mathrm{P}}+$ exhibit performance very close to Shamir scheme, particularly when sharing a large number of secrets. For instance, when sharing more than 100 secrets, the overhead of $\mathrm{B}\Pi_{\mathrm{LA}}$ is below 3%, while for $\mathrm{B}\Pi_{\mathrm{P}}+$ it remains around 8%. These results demonstrate that $\mathrm{B}\Pi_{\mathrm{LA}}$ enables verifiability in Shamir secret sharing in post-quantum and large-scale settings at almost no additional cost. This efficiency suggests that many existing protocols relying on Shamir secret sharing can be revisited and improved by substituting with $\mathrm{B}\Pi_{\mathrm{LA}}$ to obtain verifiability essentially *for free*.

Verification Time. The plots in Fig. 5 evaluate the verification time of each scheme. We observe that $\mathrm{B}\Pi_{\mathrm{LA}}$ again achieves the best performance: each party can verify 100 received shares in roughly 3 microseconds. Similarly, the $\mathrm{B}\Pi_{\mathrm{P}}+$ scheme, which additionally provides information-theoretic privacy, requires around 8 microseconds. Despite this increase, the verification remains extremely fast in practice and well within the range required for large-scale deployments.



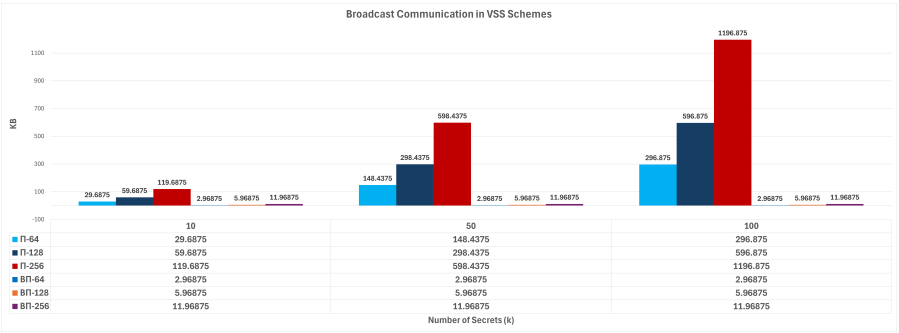
	1	10	50	100
Π_P	2.91998	29.1998	145.999	291.998
Π_F	2.16708	21.6708	108.354	216.708
ВΠ_P	2.91998	11.3665	46.765	91.585
ВΠ_F	2.16708	10.3923	45.958	89.002
Π_LA	0.83026	8.3026	41.513	83.026
ВΠ_P+	2.19448	4.891	15.6319	29.0099
ВΠ_LA	0.83026	3.47872	14.26172	27.5965
Shamir	0.34448	2.9574	13.649	26.866

Fig. 4. Sharing time in Π_{F}, Π_{P}, Π_{LA}, $\mathrm{B}\Pi_{\mathrm{F}}$, $\mathrm{B}\Pi_{\mathrm{P}}$, $\mathrm{B}\Pi_{\mathrm{P}}+$, $\mathrm{B}\Pi_{\mathrm{LA}}$ and (baseline) Shamir.



	1	10	50	100
Π_P	0.009206641	0.092066406	0.460332031	0.920664063
Π_F	0.006565234	0.065652344	0.328261719	0.656523438
ВΠ_P	0.009206641	0.031878906	0.130574219	0.25065625
ВΠ_F	0.006565234	0.029064844	0.125	0.241945313
Π_LA	0.001702109	0.017021094	0.085105469	0.170210938
ВΠ_P+	0.006506641	0.006853906	0.007471875	0.008090625
ВΠ_LA	0.001702109	0.001975039	0.002627539	0.003210195

Fig. 5. Verification time in VSS schemes Π_{F}, Π_{P}, Π_{LA}, $\mathrm{B}\Pi_{\mathrm{F}}$, $\mathrm{B}\Pi_{\mathrm{P}}$, $\mathrm{B}\Pi_{\mathrm{P}}+$, $\mathrm{B}\Pi_{\mathrm{LA}}$.



	10	50	100
Π-64	29.6875	148.4375	296.875
Π-128	59.6875	298.4375	596.875
Π-256	119.6875	598.4375	1196.875
ВΠ-64	2.96875	2.96875	2.96875
ВΠ-128	5.96875	5.96875	5.96875
ВΠ-256	11.96875	11.96875	11.96875

Fig. 6. Broadcast communication in schemes built by Π and ВΠ for $n = 64, 128, 256$.

Broadcast Communication. The plots in Fig. 5 focuses on broadcast communication overhead for different VSS schemes, measured for $n \in \{64, 128, 256\}$ and various values of k ranging from 1 to 100. Here, the benefits of batching become clear: broadcast communication in batched protocols decreases significantly, by a factor of k. This reduction translates directly into efficiency gains for higher-level protocols that build on top of VSS, as communication often represents a major performance bottleneck in distributed systems.

Performance of the Batched Pedersen VSS Scheme. We implemented our proposed batched variants of Feldman and Pedersen VSS [22,29]. For $(n, t) = (256, 127)$ and $k = 10$: in the sharing phase, Pedersen incurs 395% overhead over Shamir (14.61 ms vs. 2.95 ms), while batched Pedersen reduces this to 226% (9.62 ms) and $\text{BII}_\text{P}+$ to just 65% (4.89 ms). The verification gains are even more pronounced: Pedersen requires 5.33 ms, batched Pedersen 0.61 ms, and $\text{BII}_\text{P}+$ only 0.0068 ms, an improvement of three orders of magnitude. For larger k, these overheads decrease further, confirming that $\text{BII}_\text{P}+$ achieves near-Shamir efficiency in sharing and orders-of-magnitude improvements in verification.

Performance of the BP-PVSS Scheme BII_S. We further benchmark BII_S against the original II_S [5] and a passive-secure baseline consisting of plain Shamir secret sharing with encrypted shares and no proof. This baseline captures the minimum dealer-side cost, even when the dealer is trusted. Table 4 reports sharing times for $(n, t) = (256, 127)$ and varying batch sizes k. The results show that the overhead of BII_S over the passive baseline quickly becomes negligible as k grows: for $k \geq 100$, it is below 1%. Thus, in some large-scale batched settings, public verifiability can be added to Shamir secret sharing at essentially no extra dealer-side cost. Compared to II_S, BII_S also reduces the computational cost of verification by approximately 50%.

Table 4. Sharing time (ms) for $(n, t) = (256, 127)$ across varying k. Overhead is computed relative to the passive baseline.

PVSS Scheme	$k = 1$	$k = 10$	$k = 50$	$k = 100$	$k = 250$
Passive baseline	1.47	12.72	61.97	122.99	305.1
II_S [5]	2.77	27.76	138.80	277.60	694.1
BII_S(ours)	2.77	14.34	63.80	125.30	308.9
Overhead of BII_S over baseline	88%	13%	3%	1.9%	1.2%

7 ALBATROSS+: Boosting ALBATROSS with Batching

7.1 ALBATROSS Randomness Generation Protocol

ALBATROSS [14] is a multiparty randomness generation protocol that combines PVSS with extraction techniques to efficiently produce uniformly random values. At its core, it uses a packed variant of the II_S PVSS scheme [5,14], allowing each

party to secret-share ℓ secrets via a single polynomial. This reduces cost by encoding multiple secrets into one sharing instance. The protocol then applies linear t-resilient functions (via FFT) to extract ℓ^2 random values, even with up to t corruptions. Thus, ALBATROSS achieves $O(1)$ amortized exponentiations per party per value under the DDH assumption, outperforming prior works such as SCRAPE [13]. For a comparative study of randomness protocols, see [28].

7.2 Limitations of Packed Secret Sharing and Protocol Repetition

Packed Shamir secret sharing requires $n \geq 2t + \ell$ for privacy and correctness, a constraint inherited by ALBATROSS. For fixed n, this bounds $\ell \leq n - 2t$, so increasing ℓ demands larger n or smaller t. Reducing t weakens robustness and fails for large ℓ. When n is small, ℓ is severely limited, forcing repeated executions to scale randomness. For example, with $n = 7$, $t = 1$, one can share at most $\ell = 5$ secrets, yielding $\ell^2 = 25$ random values. To obtain $L = 1000$ values requires $k = \frac{1000}{25} = 40$ runs, greatly increasing overhead. More precisely, the amortized cost rises from $O(1)$ to $O(k)$, where $k = \frac{L}{(n-2t)^2}$.

7.3 ALBATROSS with Batching: Eliminating Repetition Overheads

To enhance efficiency, we propose ALBATROSS+, which integrates our batched and packed PVSS (Section 5). By batching multiple PVSS instances into one execution, ALBATROSS+ shares secret vectors across different polynomials (each satisfying $n \geq 2t + \ell$) while reusing common algebraic structures. This enables aggregated proofs and verifications, eliminating repetition without losing public verifiability or security. With n parties, ALBATROSS+ generates $(\ell k)^2$ random values in a single run, matching repeated executions but with sublinear cost growth. Thus, the amortized per-value cost remains $O(1)$, even for small n, making ALBATROSS+ highly scalable for real-world randomness generation.

The ALBATROSS+ protocol replaces the packed Π_S with $B\Pi_S$ and, like the original, proceeds in four phases among n parties with at most $t < \frac{n-\ell}{2}$ corruptions. Setting $\ell = n - 2t$, parties use a public ledger to post information, assuming BP-PVSS initialization and registered keys $\mathsf{pk}_i{}_{i=1}^{n}$. They also agree on a Vandermonde $(n - 2t) \times (n - t)$ matrix $M = M(\omega, n - 2t, n - t)$ with $\omega \in \mathbb{Z}_q^*$.

Commit Phase: For $1 \leq m \leq n$:

- Party P_m executes Share phase of $B\Pi_S$ as Dealer for $k \times \ell$ secrets, where $\ell = n - 2t$. Publishes encrypted shares $y_{m,1}^{\langle j \rangle} = \mathsf{pk}_1^{f_m^{\langle j \rangle}(1)}, \ldots, y_{m,n}^{\langle j \rangle} = \mathsf{pk}_n^{f_m^{\langle j \rangle}(n)}$ for $j = 1, \ldots, k$ along with the NIZK batched proof $\pi_{share} := (d, z(x))$. The party learns the secrets $\{g^{f_{-(\ell-1),m}^{\langle j \rangle}}, \cdots, g^{f_{0,m}^{\langle j \rangle}}\}_{j=1}^{k}$ and the exponents $\{f_{-(\ell-1),m}^{\langle j \rangle}, \cdots, f_{0,m}^{\langle j \rangle}\}_{j=1}^{k}$.

Reveal Phase: 1. All parties first verify the encrypted shares via Verify algorithm of new BP-PVSS scheme.

2. Once $n - t = t + \ell$ valid set of sharings are posted (set $\mathbb{C}$), each party $P_m \in \mathbb{C}$ reveals their k sharing polynomials $f_m^{\langle 1 \rangle}(x), \ldots, f_m^{\langle k \rangle}(x)$.
3. Every party now verifies that indeed $f_m^{\langle 1 \rangle}(x), \ldots, f_m^{\langle k \rangle}(x)$ match committed shares by recomputing $y_{m,1}^{\langle j \rangle} = \mathsf{pk}_1^{f_m^{\langle j \rangle}(1)}, \ldots, y_{m,n}^{\langle j \rangle} = \mathsf{pk}_n^{f_m^{\langle j \rangle}(n)}$ for $j = 1, \ldots, k$.
4. If all parties in $\mathbb{C}$ open correctly, go to *Alternative output* phase, otherwise proceed to the *Recovery* phase.

Recovery: 1. Let $\mathbb{C}_A \subseteq \mathbb{C}$ be parties failing to open secret polynomials.
2. All parties execute amortized share decryption for BP-PVSSs from $\mathbb{C}_A$.
3. Verify decrypted shares via k-DLEQ proofs (see the full version).
4. Once $n - t = t + \ell$ valid set of shares are published (set $\mathcal{Q}$), reconstruct secrets using the reconstruction algorithm of $\mathrm{B}\Pi_{\mathrm{S}}$ scheme.

Output: 1. Let T be $(n-t) \times (\ell k)$ matrix with rows indexed by the parties in $\mathbb{C}$, and where the row corresponding to P_m is $(g^{f_{-(\ell-1),m}^{\langle 1 \rangle}}, \cdots, g^{f_{0,m}^{\langle 1 \rangle}}), \ldots, (g^{f_{-(\ell-1),m}^{\langle k \rangle}}, \cdots, g^{f_{0,m}^{\langle k \rangle}})$.
2. Each computes the $\ell k \times \ell k$-matrix $R = M \circ T$ by applying FFTE on each column $T^{(m)}$ of T, resulting in column $R^{(m)}$ of R (since $R^{(m)} = M \circ T^{(m)}$ and M is Vandermonde) for $m \in [0, \ldots, \ell k - 1]$. Here $\circ$ denotes the matrix multiplication.
3. Parties output the $(\ell k)^2$ elements of R as final randomness.

Alternative output: if every party in $\mathbb{C}$ has opened her secrets correctly in step *Reveal*, then:
1. Parties compute $R = M \circ T$ in the following way: Let S be $(n-t) \times (\ell k)$ matrix with rows indexed by the parties in $\mathbb{C}$ and where the row corresponding to P_m is $(g^{f_{-(\ell-1),m}^{\langle 1 \rangle}}, \cdots, g^{f_{0,m}^{\langle 1 \rangle}}), \ldots, (g^{f_{-(\ell-1),m}^{\langle k \rangle}}, \cdots, g^{f_{0,m}^{\langle k \rangle}})$. Then, each party computes $U = M \cdot S \in \mathbb{Z}_q^{\ell k \times k \ell}$ (using the standard FFT in $\mathbb{Z}_q$ to compute each column) and $R = g^U$, meaning the (i,j)-th element in R is g^y, where y is the (i,j)-th element in U.
2. Parties output the $(\ell k)^2$ elements of R as final randomness.

Security & Efficiency. The security of ALBATROSS+ inherits the guarantees of the original ALBATROSS protocol, as we only replace the packed Π_{S} with $\mathrm{B}\Pi_{\mathrm{S}}$, a batched and packed variant. This preserves the same security properties while enabling batching k executions of the protocol.

Our hybrid approach asymptotically reduces protocol executions, achieving $O(1)$ amortized communication versus the $O(k)$ cost of naive repetition. Table 2 shows ALBATROSS+ improves commit and reveal phases by $\sim 50\%$. These gains preserve security while enabling efficient high-throughput distributed randomness generation with few parties.

8 Conclusion

We introduced a tunable batched-and-packed extension of the unified Π framework [5], enabling highly efficient (publicly) verifiable secret sharing protocols

based on Shamir sharing in the synchronous model. As in the original framework, our construction relies only on a random oracle and a secure basic vector commitment scheme, without imposing homomorphism requirements. The tunable parameters allow practitioners to flexibly trade robustness for efficiency, making the framework adaptable to a wide range of deployment scenarios.

Using this generalized protocol, we derived batched and packed variants of all VSS schemes from [5], achieving substantial efficiency gains for large-scale secret sharing. In particular, our hash-based instantiation yields lightweight and plausibly post-quantum-secure batched VSS schemes, making it especially attractive for lattice- and isogeny-based threshold cryptography. We also proposed batched variants of classical Feldman [22] and Pedersen [29] VSS schemes, thereby unifying both classical and modern constructions under a single tunable framework.

To obtain an efficient perfectly simulatable batched VSS scheme that provides information-theoretic privacy against computationally unbounded adversaries, we introduced a novel and simple perfectly hiding vector commitment scheme in the random oracle model. This commitment significantly outperforms the generalized vector Pedersen commitment while retaining re-randomizability, and we expect it to find applications beyond the VSS setting.

Building on the same framework, we further presented the first tunable batched and packed publicly verifiable secret sharing (PVSS) scheme, generalizing earlier packed PVSS constructions [5, 14]. A key technical ingredient is a batched generalization of Baghery's Schnorr-style protocol [5], which enables proving knowledge of multiple secret polynomials at nearly the cost of a single non-batched instance. As a concrete application, we revisited the ALBATROSS randomness generation protocol [14] and demonstrated that instantiating it with our batched and packed PVSS significantly improves its practical performance. More broadly, our batched PVSS construction is applicable to other publicly auditable protocols, such as those envisioned in [30].

Our implementation results confirm that both the BP-VSS and BP-PVSS protocols achieve excellent practical performance in terms of computation and communication. In particular, when sharing many secrets, the BP-VSS schemes $\mathrm{B}\Pi_{\mathrm{LA}}$ and $\mathrm{B}\Pi_{\mathrm{P}}+$ and the BP-PVSS scheme $\mathrm{B}\Pi_{\mathrm{S}}$ closely match the distribution cost of plain Shamir secret sharing, demonstrating that strong verifiability guarantees can be obtained at almost no additional cost in large-scale settings. In summary, the tunable batched-and-packed nature of our framework provides a versatile foundation for constructing efficient VSS and PVSS schemes and for re-engineering threshold protocols. While a full exploration of these applications is beyond the scope of this work, promising directions for future research include integration with advanced distributed protocols and further optimizations tailored to emerging post-quantum cryptographic primitives.

Acknowledgments. This work was supported by the Flemish Government through the Cybersecurity Research Program with grant number VOEWICS02 and Research grant VIL53029 from VILLUM FONDEN. Jannik Spiessens is funded by an FWO fellowship with reference number 1139125N.

References

1. Atapoor, S., Baghery, K., Cozzo, D., Pedersen, R.: Practical robust DKG protocols for CSIDH. In: Tibouchi, M., Wang, X. (eds.) ACNS 2023: 21st International Conference on Applied Cryptography and Network Security, Part II. Lecture Notes in Computer Science, vol. 13906, pp. 219–247. Springer, Cham, Switzerland, Kyoto, Japan (Jun 19–22, 2023). https://doi.org/10.1007/978-3-031-33491-7_9
2. Atapoor, S., Baghery, K., Cozzo, D., Pedersen, R.: VSS from distributed ZK proofs and applications. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology – ASIACRYPT 2023, Part I. Lecture Notes in Computer Science, vol. 14438, pp. 405–440. Springer, Singapore, Singapore, Guangzhou, China (Dec 4–8, 2023). https://doi.org/10.1007/978-981-99-8721-4_13
3. Atapoor, S., Baghery, K., Nicolas, G., Pedersen, R., Spiessens, J.: Batched and packed (publicly) verifiable secret sharing: A unified framework and applications. Cryptology ePrint Archive, Report 2025/2018 (2025), https://eprint.iacr.org/2025/2018
4. Backes, M., Kate, A., Patra, A.: Computational verifiable secret sharing revisited. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology – ASIACRYPT 2011. Lecture Notes in Computer Science, vol. 7073, pp. 590–609. Springer Berlin Heidelberg, Germany, Seoul, South Korea (Dec 4–8, 2011). https://doi.org/10.1007/978-3-642-25385-0_32
5. Baghery, K.: Π: A unified framework for computational verifiable secret sharing. In: Jager, T., Pan, J. (eds.) PKC 2025: 28th International Conference on Theory and Practice of Public Key Cryptography, Part IV. Lecture Notes in Computer Science, vol. 15677, pp. 110–142. Springer, Cham, Switzerland, Røros, Norway (May 12–15, 2025). https://doi.org/10.1007/978-3-031-91829-2_4
6. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: 20th Annual ACM Symposium on Theory of Computing. pp. 1–10. ACM Press, Chicago, IL, USA (May 2–4, 1988). https://doi.org/10.1145/62212.62213
7. Beullens, W., Disson, L., Pedersen, R., Vercauteren, F.: CSI-RAShi: Distributed key generation for CSIDH. In: Cheon, J.H., Tillich, J.P. (eds.) Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021. pp. 257–276. Springer, Cham, Switzerland, Daejeon, South Korea (Jul 20–22, 2021). https://doi.org/10.1007/978-3-030-81293-5_14
8. Beullens, W., Kleinjung, T., Vercauteren, F.: CSI-FiSh: Efficient isogeny based signatures through class group computations. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology – ASIACRYPT 2019, Part I. Lecture Notes in Computer Science, vol. 11921, pp. 227–247. Springer, Cham, Switzerland, Kobe, Japan (Dec 8–12, 2019). https://doi.org/10.1007/978-3-030-34578-5_9
9. Blakley, G.R.: Safeguarding cryptographic keys. In: 1979 International Workshop on Managing Requirements Knowledge, MARK 1979, New York, NY, USA, June 4-7, 1979. pp. 313–318. IEEE (1979). https://doi.org/10.1109/MARK.1979.8817296, https://doi.org/10.1109/MARK.1979.8817296
10. Blakley, G.R., Meadows, C.: Security of ramp schemes. In: Blakley, G.R., Chaum, D. (eds.) Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings. Lecture Notes in Computer Science, vol. 196, pp. 242–268. Springer (1984). https://doi.org/10.1007/3-540-39568-7_20, https://doi.org/10.1007/3-540-39568-7_20

11. Bootle, J., Groth, J.: Efficient batch zero-knowledge arguments for low degree polynomials. In: Abdalla, M., Dahab, R. (eds.) PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part II. Lecture Notes in Computer Science, vol. 10770, pp. 561–588. Springer, Cham, Switzerland, Rio de Janeiro, Brazil (Mar 25–29, 2018). https://doi.org/10.1007/978-3-319-76581-5_19
12. Cascudo, I., Cozzo, D., Giunta, E.: Verifiable secret sharing from symmetric key cryptography with improved optimistic complexity. In: Chung, K.M., Sasaki, Y. (eds.) Advances in Cryptology – ASIACRYPT 2024, Part VII. Lecture Notes in Computer Science, vol. 15490, pp. 100–128. Springer, Singapore, Singapore, Kolkata, India (Dec 9–13, 2024). https://doi.org/10.1007/978-981-96-0941-3_4
13. Cascudo, I., David, B.: SCRAPE: Scalable randomness attested by public entities. In: Gollmann, D., Miyaji, A., Kikuchi, H. (eds.) ACNS 2017: 15th International Conference on Applied Cryptography and Network Security. Lecture Notes in Computer Science, vol. 10355, pp. 537–556. Springer, Cham, Switzerland, Kanazawa, Japan (Jul 10–12, 2017). https://doi.org/10.1007/978-3-319-61204-1_27
14. Cascudo, I., David, B.: ALBATROSS: Publicly AttestabLe BATched Randomness based On Secret Sharing. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2020, Part III. Lecture Notes in Computer Science, vol. 12493, pp. 311–341. Springer, Cham, Switzerland, Daejeon, South Korea (Dec 7–11, 2020). https://doi.org/10.1007/978-3-030-64840-4_11
15. Cascudo, I., David, B.: Publicly verifiable secret sharing over class groups and applications to DKG and YOSO. In: Joye, M., Leander, G. (eds.) Advances in Cryptology – EUROCRYPT 2024, Part V. Lecture Notes in Computer Science, vol. 14655, pp. 216–248. Springer, Cham, Switzerland, Zurich, Switzerland (May 26–30, 2024). https://doi.org/10.1007/978-3-031-58740-5_8
16. Cascudo, I., David, B., Garms, L., Konring, A.: YOLO YOSO: Fast and simple encryption and secret sharing in the YOSO model. In: Agrawal, S., Lin, D. (eds.) Advances in Cryptology – ASIACRYPT 2022, Part I. Lecture Notes in Computer Science, vol. 13791, pp. 651–680. Springer, Cham, Switzerland, Taipei, Taiwan (Dec 5–9, 2022). https://doi.org/10.1007/978-3-031-22963-3_22
17. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) Advances in Cryptology – ASIACRYPT 2018, Part III. Lecture Notes in Computer Science, vol. 11274, pp. 395–427. Springer, Cham, Switzerland, Brisbane, Queensland, Australia (Dec 2–6, 2018). https://doi.org/10.1007/978-3-030-03332-3_15
18. Chaum, D., Evertse, J.H., van de Graaf, J.: An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In: Chaum, D., Price, W.L. (eds.) Advances in Cryptology – EUROCRYPT'87. Lecture Notes in Computer Science, vol. 304, pp. 127–141. Springer Berlin Heidelberg, Germany, Amsterdam, The Netherlands (Apr 13–15, 1988). https://doi.org/10.1007/3-540-39118-5_13
19. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In: 26th Annual Symposium on Foundations of Computer Science. pp. 383–395. IEEE Computer Society Press, Portland, Oregon (Oct 21–23, 1985). https://doi.org/10.1109/SFCS.1985.64

20. Cramer, R., Damgård, I., Maurer, U.M.: General secure multi-party computation from any linear secret-sharing scheme. In: Preneel, B. (ed.) Advances in Cryptology – EUROCRYPT 2000. Lecture Notes in Computer Science, vol. 1807, pp. 316–334. Springer Berlin Heidelberg, Germany, Bruges, Belgium (May 14–18, 2000). https://doi.org/10.1007/3-540-45539-6_22
21. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings. Lecture Notes in Computer Science, vol. 435, pp. 307–315. Springer (1989). https://doi.org/10.1007/0-387-34805-0_28, https://doi.org/10.1007/0-387-34805-0_28
22. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: 28th Annual Symposium on Foundations of Computer Science. pp. 427–437. IEEE Computer Society Press, Los Angeles, CA, USA (Oct 12–14, 1987). https://doi.org/10.1109/SFCS.1987.4
23. Franklin, M.K., Yung, M.: Communication complexity of secure computation (extended abstract). In: Kosaraju, S.R., Fellows, M., Wigderson, A., Ellis, J.A. (eds.) Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada. pp. 699–710. ACM (1992). https://doi.org/10.1145/129712.129780, https://doi.org/10.1145/129712.129780
24. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. Journal of Cryptology **20**(1), 51–83 (Jan 2007). https://doi.org/10.1007/s00145-006-0347-3
25. Gennaro, R., Rabin, M.O., Rabin, T.: Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In: Coan, B.A., Afek, Y. (eds.) 17th ACM Symposium Annual on Principles of Distributed Computing. pp. 101–111. Association for Computing Machinery, Puerto Vallarta, Mexico (Jun 28 – Jul 2, 1998). https://doi.org/10.1145/277697.277716
26. Gentry, C., Halevi, S., Lyubashevsky, V.: Practical non-interactive publicly verifiable secret sharing with thousands of parties. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology – EUROCRYPT 2022, Part I. Lecture Notes in Computer Science, vol. 13275, pp. 458–487. Springer, Cham, Switzerland, Trondheim, Norway (May 30 – Jun 3, 2022). https://doi.org/10.1007/978-3-031-06944-4_16
27. Groth, J.: Non-interactive distributed key generation and key resharing. Cryptology ePrint Archive, Report 2021/339 (2021), https://eprint.iacr.org/2021/339
28. Kavousi, A., Wang, Z., Jovanovic, P.: Sok: Public randomness. In: 9th IEEE European Symposium on Security and Privacy, EuroS&P 2024, Vienna, Austria, July 8-12, 2024. pp. 216–234. IEEE (2024). https://doi.org/10.1109/EUROSP60621.2024.00020, https://doi.org/10.1109/EuroSP60621.2024.00020
29. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) Advances in Cryptology – CRYPTO'91. Lecture Notes in Computer Science, vol. 576, pp. 129–140. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 11–15, 1992). https://doi.org/10.1007/3-540-46766-1_9
30. Schoenmakers, B.: A simple publicly verifiable secret sharing scheme and its application to electronic. In: Wiener, M.J. (ed.) Advances in Cryptology – CRYPTO'99. Lecture Notes in Computer Science, vol. 1666, pp. 148–164. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 1999). https://doi.org/10.1007/3-540-48405-1_10
31. Shamir, A.: How to share a secret. Communications of the ACM **22**(11), 612–613 (1979)

32. Shoup, V., Smart, N.P.: Lightweight asynchronous verifiable secret sharing with optimal resilience. Journal of Cryptology **37**(3), 27 (Jul 2024). https://doi.org/10.1007/s00145-024-09505-6
33. So, J., Güler, B., Avestimehr, A.S.: Byzantine-resilient secure federated learning. IEEE J. Sel. Areas Commun. **39**(7), 2168–2181 (2021). https://doi.org/10.1109/JSAC.2020.3041404, https://doi.org/10.1109/JSAC.2020.3041404
34. Tomescu, A., Chen, R., Zheng, Y., Abraham, I., Pinkas, B., Golan-Gueta, G., Devadas, S.: Towards scalable threshold cryptosystems. In: 2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020. pp. 877–893. IEEE (2020). https://doi.org/10.1109/SP40000.2020.00059, https://doi.org/10.1109/SP40000.2020.00059