Batched and Packed (Publicly) Verifiable Secret Sharing

Boosting practical performance of VSS schemes

S. Atapoor, K. Baghery, G. Nicolas, R. Pedersen, **J. Spiessens**
COSIC, KU Leuven

Outline

Introduction

The Basics of Secret Sharing

- A **dealer** distributes a secret f_0 among n parties
- Any **qualified subset** (size $> t$) can reconstruct the secret
- Any **unqualified subset** (size $\leq t$) learns nothing
- **Shamir's secret sharing** (1979): encode f_0 as the free coefficient of a random degree-t polynomial $f(x)$, give party P_i the share $f_i = f(i)$
- Reconstruction via Lagrange interpolation from any $t + 1$ shares

$\rightarrow$ Secure only against *passive* adversaries: a malicious dealer or malicious parties can break correctness

Why Verifiable Secret Sharing (VSS)?

- Prevents **shareholders** from dishonestly opening secret shares
- Prevents **dealer** from dishonestly generating the secret shares
- **Publicly** Verifiable Secret Sharing (PVSS): *anyone* can perform verification

Applications:

- Maliciously-secure Multi-Party Computation
- Distributed Key Generation (DKG)
- Threshold cryptography (signatures, decryption)
- Distributed randomness generation (e.g. ALBATROSS)

Security Notions for VSS

t out of n parties are malicious
$\Rightarrow n - t \geq t + 1$ for reconstruction $\Rightarrow n \geq 2t + 1$ $\rightarrow$ **honest majority** setting

Security properties:

- **Completeness:** an honest dealer always convinces honest parties and any $t + 1$ honest parties can reconstruct
- **Verifiability:** if verification succeeds, *any* qualified subset Q of size $|Q| \geq t + 1$ reconstructs the *same* unique secret
- **Unpredictability:** no subset of parties Q of size $|Q| \leq t$ learns anything about the secret

Feldman/Pedersen's VSS Scheme

- Classical approach: publish homomorphic commitments to the coefficients of $f(x)$
- Exploit the homomorphism to let each party check its share *against the public commitments*
- Two classical instantiations, both using a group G of prime order q with hard Discrete Logarithm (DL):
  - **Feldman VSS** (1987) — efficient, but requires high-entropy secrets (no hiding)
  - **Pedersen VSS** (1991) — perfectly hiding, supports low-entropy secrets

Feldman/Pedersen's VSS Scheme

Summary of the construction

Share: f_0 using generators g_1, g_2

1. Sample random $f(X) = f_0 + a_1 X + \cdots + a_t X^t$ and $r(X) = r_0 + b_1 X + \cdots + b_t X^t$
2. For $i = 1, \ldots, n$: set $f_i := f(i)$ and $r_i := r(i)$
3. Compute commitments $c_0 = g_1^{f_0} g_2^{r_0}$, $c_j = g_1^{a_j} g_2^{b_j}$ for $j = 1, \ldots, t$
4. Broadcast $\pi_{\text{share}} := (c_0, \ldots, c_t)$; send f_i, r_i privately to P_i

Verify: P_i accepts iff $g_1^{f_i} g_2^{r_i} = \prod_{j=0}^{t} c_j^{i^j}$.

- Verification: $O(t)$ exponentiations

  Reconstruction: $O(t^2)$ exponentiations
- Pedersen is twice as expensive but has information-theoretic privacy

Batching and Packing

Increase throughput with Batching

- Many applications need the *same* dealer to share **many** secrets with the **same** group of parties (e.g. randomness beacons, sharing Beaver triples, e-voting, ...)
- Share k secrets using k independent degree-t polynomials
- Only a *single* proof for all $n \times k$ shares

Goal: amortize computational and communication cost across the k instances

Increase throughput with Packing

- Batching k secrets always requires distributing k shares
- **Packed secret sharing:** encode ℓ secrets into a *single* polynomial of degree $d = t + \ell - 1$
- Privacy/reconstruction trade-off changes:
  - t-privacy and $(t + \ell)$-reconstruction thus requiring $n \geq 2t + \ell$

$\Rightarrow$ Packing **trades resilience for efficiency**: for fixed n, increasing ℓ forces a smaller tolerated t

- Batching and packing are **complementary**

Batching and Packing combined

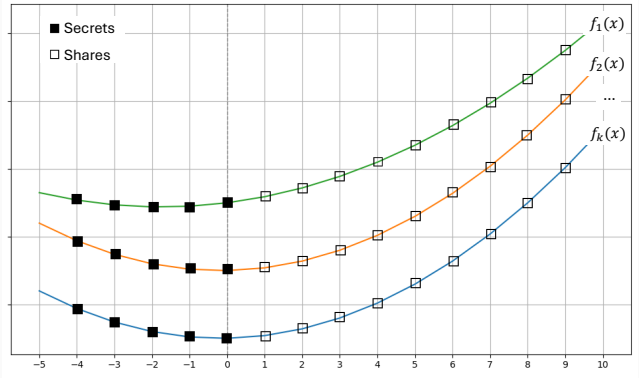


Figure 1: A visualization of $k \times \ell$ secrets (■) and $k \times n$ shares (□) encoded via k random degree $t + \ell - 1$ polynomials. The $k \times \ell$ secrets are encoded at points $j = 0, -1, \ldots, -(\ell - 1)$, while the $k \times n$ shares are encoded at points $i = 1, 2, \ldots, n$. A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

Batched and Packed Feldman / Pedersen

- Feldman and Pedersen schemes can be be batched and packed
- Idea: use k extra generators $g_1, \ldots, g_k$, one per secret
- Efficiency versus k sequential executions:
  - Broadcast communication reduced by a factor k
  - Verification cost: $O(kn) \rightarrow O(k + n)$ exponentiations
- and operates in the plain model!

Batched and Packed Feldman / Pedersen

Summary of the construction

Share: $\{f_{-(\ell-1)}^{\langle j\rangle}, \ldots, f_0^{\langle j\rangle}\}_{j=1}^{k}$ using generators $g_1, \ldots, g_k, g_{k+1}$

1. Sample k random $f^{\langle j\rangle}(X) = f_0^{\langle j\rangle} + a_1^{\langle j\rangle} X + \cdots + a_{t+\ell-1}^{\langle j\rangle} X^{t+\ell-1}$ such that $f^{\langle j\rangle}(i) = f_i^{\langle j\rangle}$ for $i = -(\ell-1), \ldots, 0$
2. Sample random $r(X) = r_0 + b_1 X + \cdots + b_{t+\ell-1} X^{t+\ell-1}$
3. For $i = 1, \ldots, n$ and $j = 1, \ldots, k$: set $f_i^{\langle j\rangle} := f^{\langle j\rangle}(i)$ and $r_i := r(i)$
4. Compute commitments $c_0 = g_{k+1}^{r_0} \prod_{j=1}^{k} g_j^{f_0^{\langle j\rangle}}$ and $c_v = g_{k+1}^{b_v} \prod_{j=1}^{k} g_j^{a_v^{\langle j\rangle}}$ for $v = 1, \ldots, t+\ell-1$
5. Broadcast $\pi_{\text{share}} := (c_0, \ldots, c_{t+\ell-1})$; send $r_i, \{f_i^{\langle j\rangle}\}_{j=1}^{k}$ privately to P_i

Verify: P_i accepts iff $g_{k+1}^{r_i} \prod_{j=1}^{k} g_j^{f_i^{\langle j\rangle}} = \prod_{v}^{t+\ell-1} c_v^{i^v}$.

- Verification still requires $k + t + \ell$ exponentiations in the group
  $\rightarrow$ can be reduced to k exponentiations $\rightarrow$ can be reduced to 0 exponentiations

The Π Framework

Theoretical Basis: Strong ZK on Distributed Data

- Baghery's Π framework (PKC 2025) builds Boneh et al.'s **Strong (Threshold) Zero-Knowledge** proofs on secret-shared data
- Core object: the **Shamir relation**, an n-distributed relation

$$R_i = \{(f_i, f(X)) \mid f(i) = f_i\}, \quad i = 1, \ldots, n$$

  where each verifier P_i only holds its own share f_i of the full statement
- Constructs a Strong ZK proof that shows a *single* degree-t polynomial underlies all shares while up to t colluding verifiers learn *nothing besides their own share*
- Π turns this into a **unified framework** for building VSS schemes
  $\rightarrow$ in the Random Oracle (RO) model
  $\rightarrow$ without requiring homomorphic commitments!

The Generic Π Construction

Summary of the construction

Share: f_0 using generators commitment C and RO H

1. Sample random degree-t polynomials $f(X)$ and $r(X)$
2. For $i = 1, \ldots, n$: $f_i := f(i)$, $r_i := r(i)$ and $c_i := C(f_i, r_i)$
3. $d \leftarrow H(c_1, \ldots, c_n)$; set $z(X) = r(X) + d \cdot f(X)$
4. Broadcast $\pi_{\text{share}} := (c_1, \ldots, c_n, z(X))$; send f_i privately to P_i

Verify: P_i accepts iff $\deg z \leq t$ and $c_i = C(f_i, z(i) - d \cdot f_i)$

- Intuition: adjust Schnorr identification scheme to enforce Shamir-type sharing
- Only needs commitment C to be hiding + binding

Specializing Π: DL-Based Instances

- Instantiating C recovers efficient alternatives to Feldman and Pedersen VSS:
  - Π_{F}: $c_i = g_1^{f_i} g_2^{r_i}$ (alternative to Feldman, needs high-entropy secrets)
  - Π_{P}: $c_i = g_1^{f_i} g_2^{r_i} g_3^{\gamma_i}$ (alternative to Pedersen, perfectly simulatable)
- Both push cost from reconstruction/verification into the sharing phase:
  - Verification: $O(1)$ exponentiations (vs. $O(t)$ for Feldman/Pedersen)
  - Reconstruction: $O(t)$ exponentiations (vs. $O(t^2)$ for Feldman/Pedersen)
  - Trade-off: a constant factor **slower sharing** and **larger communication**

Specializing Π: Π_{LA}

- Instantiate C with a plain **hash function**: $c_i = H(f_i, r_i)$ or $c_i = H(f_i, r_i, \gamma_i)$
  Verify: P_i accepts iff $\deg z \leq t$ and $c_i = H(f_i, z(i) - d \cdot f_i)$
- No group operations required $\rightarrow$ only lightweight cryptography
  - extremely fast
  - **plausibly post-quantum**
- Outperforms the prior PQ-secure VSS scheme of ABCP'23 in every metric

Batched and Packed Π: $\mathrm{B}\Pi$

(k, ℓ)-BΠ: the Idea

- Extend Π to share $k \times \ell$ secrets using k random polynomials

$$f^{\langle 1 \rangle}(x), \ \ldots, \ f^{\langle k \rangle}(x)$$

  of degree $t + \ell - 1$

- ℓ secrets are packed at points $-(\ell - 1), \ldots, -1, 0$
  n shares per polynomial at points $1, \ldots, n$
  $\rightarrow$ other evaluation points give faster algorithms

- A **single** threshold proof certifies correctness of *all* $k \times n$ shares at once

The (k, ℓ)-BΠ Construction

Summary of construction

Share: $\{f_{-(\ell-1)}^{\langle j \rangle}, \ldots, f_0^{\langle j \rangle}\}_{j=1}^{k}$ using commitment C and RO H

1. Sample k random $f^{\langle j \rangle}(X) = f_0^{\langle j \rangle} + a_1^{\langle j \rangle} X + \cdots + a_{t+\ell-1}^{\langle j \rangle} X^{t+\ell-1}$ such that $f^{\langle j \rangle}(i) = f_i^{\langle j \rangle}$ for $i = -(\ell-1), \ldots, 0$
2. Sample random degree-$(t+\ell-1)$ polynomial $r(X)$
3. For $i = 1, \ldots, n$: $f_i^{\langle j \rangle} := f^{\langle j \rangle}(i)$, $r_i := r(i)$ and $c_i := C\big((f_i^{\langle 1 \rangle}, \ldots, f_i^{\langle k \rangle}, r_i), \gamma_i\big)$
4. $(d_1, \ldots, d_k) \leftarrow H(c_1, \ldots, c_n)$; set $z(X) = r(X) + \sum_{j=1}^{k} d_j f^{\langle j \rangle}(X)$
5. Broadcast $\pi_{\text{share}} := (c_1, \ldots, c_n, z(X))$; send $\gamma_i, \{f_i^{\langle j \rangle}\}_{j=1}^{k}$ privately to P_i

Verify: P_i accepts iff $\deg z \leq t + \ell - 1$ and $c_i = C\big((f_i^{\langle 1 \rangle}, \ldots, f_i^{\langle k \rangle}, z(i) - \sum_{j=1}^{k} d_j f_i^{\langle j \rangle}), \gamma_i\big)$

- Only **one** masking polynomial $r(x)$ and **one** RO query for all k secrets
- Proof size drops from $O(nk)$ (running Π k times) to $O(n)$

Batching versus Packing?

- **Packing** ($\ell > 1$) no additional cost but *lowers resilience*
  $\rightarrow$ number of tolerated corruptions decreases (on average by $\ell/2$)
- **Batching** ($k > 1$) reduces verification cost and proof size but *increases prover costs*
  $\rightarrow$ sharing and private communication cost have factor k

$\Rightarrow$ Combine both!

- Setting $(k, \ell) = (1, 1)$ recovers the original Π framework exactly

Four New BP-VSS Instantiations

Instantiating $\mathrm{B}\Pi$ with different commitment schemes gives four schemes, each a batched and packed counterpart of a Π scheme:

- $\mathrm{B}\Pi_{\mathrm{F}}$: DL-based, only high-entropy secrets
- $\mathrm{B}\Pi_{\mathrm{P}}$: DL-based, perfectly simulatable, vector Pedersen commitment
- $\mathrm{B}\Pi_{\mathrm{LA}}$: plausibly post-quantum, lightweight
- $\mathrm{B}\Pi_{\mathrm{P}}{}^{+}$: uses more efficient commitment
  $\rightarrow$ only 2 exponentiations + 2 hash evaluations for verification!

Commitment Scheme for $\mathrm{B\Pi_P}^+$

- **Setup**: hash $H : \{0,1\}^\star \to \mathbb{Z}_q$, generators g_1, g_2.
- **Commit**: $C(m_1, \ldots, m_k; r) := g_1^{H(m_1,\ldots,m_k)} g_2^r$.

- *Not* additively homomorphic $\to$ *not* needed in $\mathrm{B\Pi}$
- **Perfectly hiding**: inherited directly from the Pedersen commitment on $H(\vec{m})$
- **Computationally binding**: reduces to collision-resistance of H and the DL assumption
- Cost: only **2 exponentiations + 1 hash**
  versus $k+1$ exponentiations for the naive vector Pedersen commitment

Publicly Verifiable Secret Sharing

Schoenmakers' PVSS Scheme (Crypto'99)

Summary of the construction

Share: $g_1^{f_0}$ using generators g_1, g_2 and public keys $h_i = g_1^{s_i}$

1. Sample random $f(X) = f_0 + a_1 X + \cdots + a_t X^t$
2. For $i = 1, \ldots, n$: set $f_i := f(i)$ and $h_i' := h_i^{f_i}$
3. Compute commitments $c_0 = g_2^{f_0}$ and $c_j = g_2^{a_j}$ for $j = 1, \ldots, t$
4. Prove knowledge of f_i such that $x_i = g_2^{f_i} \wedge h_i' = h_i^{f_i}$ using an extended Chaum–Pedersen DLEQ proof for $x_i := \prod_{j=0}^{t} c_j^{i^j}$ for $i = 1, \ldots, n$
5. Broadcast all commitments c_j, encrypted shares h_i' and DLEQ proofs

Verify: accept iff all DLEQ proofs hold using $x_i = \prod_{j=0}^{t} c_j^{i^j}$

Reconstruct: P_i decrypts $F_i := h_i'^{1/s_i} = g_1^{f_i}$ and proves decryption using DLEQ $h_i = g_1^{s_i} \wedge h_i' = F_i^{s_i}$

- Drawback: verification costs $O(nt)$ group exponentiations

Π_{S}: An Efficient PVSS from the PDL Problem

- Baghery generalizes DLEQ to prove relation

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

Summary of π_{PDL} construction

Prove:

1. Sample degree-t polynomial $r(X)$; set $\Gamma_i = g^{r(x_i)}$ for $i = 1, \ldots, n$
2. Set $d \leftarrow H(F_1, \ldots, F_n, \Gamma_1, \ldots, \Gamma_n)$.
3. Set $z(X) = r(X) + df(X)$
4. Output $\pi := (\Gamma_1, \ldots, \Gamma_n, z(X))$

Verify: accept iff $g^{z(x_i)} = \Gamma_i (F_i)^d$

- Security: PDL + DDH assumptions, random oracle model

Batching π_{PDL}: The π_{BPDL} Protocol

Summary of π_{BPDL} construction

Prove:

1. Sample degree-t polynomial $r(X)$; set $\Gamma_i = g^{r(x_i)}$ for $i = 1, \ldots, n$
2. Set $d \leftarrow H(\{F_1^{\langle j \rangle}, \ldots, F_n^{\langle j \rangle}\}, \Gamma_1, \ldots, \Gamma_n)$
3. Set $z(X) = r(X) + \sum_{j=1}^{k} d^j f^{\langle j \rangle}(X)$.
4. Output $\pi := (\Gamma_1, \ldots, \Gamma_n, z(X))$

Verify: accept iff $g^{z(x_i)} = \Gamma_i \prod_{j=1}^{k} (F_i^{\langle j \rangle})^{d^j}$

- Proves knowledge of k witness polynomials at (nearly) the cost of **one** instance of π_{PDL}

$\mathrm{BII_S}$: Batched and Packed PVSS

Summary of the construction

Share: $\{g^{f^{\langle j\rangle}_{-(\ell-1)}}, \ldots, g^{f^{\langle j\rangle}_0}\}_{j=1}^{k}$ using commitment C and RO H and public keys $h_i = g^{s_i}$

1. Sample k random $f^{\langle j\rangle}(X) = f^{\langle j\rangle}_0 + a^{\langle j\rangle}_1 X + \cdots + a^{\langle j\rangle}_{t+\ell-1} X^{t+\ell-1}$ such that $f^{\langle j\rangle}(i) = f^{\langle j\rangle}_i$ for $i = -(\ell-1), \ldots, 0$
2. Sample random degree-$(t+\ell-1)$ polynomial $r(X)$
3. For $i = 1, \ldots, n$, $j = 1, \ldots, k$: $f^{\langle j\rangle}_i := f^{\langle j\rangle}(i)$, $r_i := r(i)$ and $h^{\langle j\rangle'}_i := h_i^{f^{\langle j\rangle}_i}$
4. Prove knowledge of $f^{\langle j\rangle}$ that interpolates all $h^{\langle j\rangle'}_i$ in the exponent using π_{BPDL}
5. Broadcast all commitments c_j, encrypted shares $h^{\langle j\rangle'}_i$ and *one* π_{BPDL} proof

Verify: accept iff π_{BPDL} proof verifies

Reconstruct: P_i decrypts $F^{\langle j\rangle}_i := (h^{\langle j\rangle'}_i)^{1/s_i} = g_1^{f^{\langle j\rangle}_i}$ and proves all decryptions using *one* DLEQ proof

$\mathrm{B}\Pi_{\mathrm{S}}$: Batched and Packed PVSS

- Built on top of $B\Pi_F$ using π_{BPDL} for public verifiability
- First PVSS scheme supporting **both** batching and packing simultaneously
- Compared to k concurrent executions of the packed Π_{S}:
  - $\sim$50% improvement in sharing *and* verification cost
  - Proof size reduced by a factor of k (amortized $O(1)$ per secret)
- Achieves near-identical efficiency to the passively-secure baseline

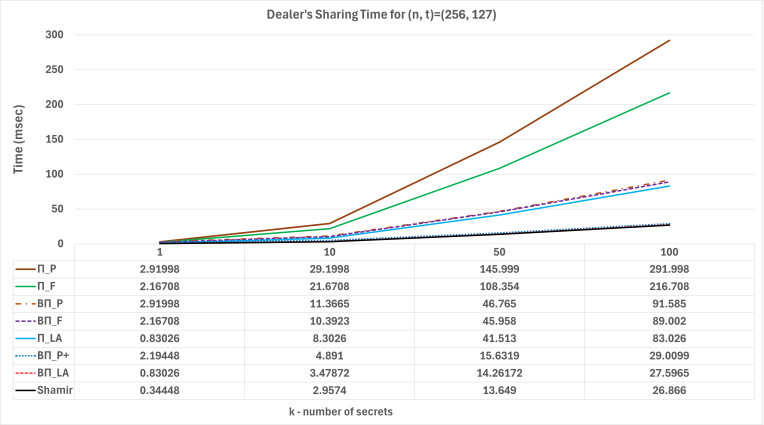
Implementation

Experimental Setup

- Prototype implementation in **Rust**, run on a MacBook Pro (M4 Pro, 24GB RAM), averaged over 100 runs
- Results reported for $\ell = 1$ (non-packed and "happy path")
- For $\ell > 1$: performance scales proportionally
- Comparing:
  - Sharing time
  - Verification time
  - Broadcast communication

  between k repetitions of Π schemes and our constructions

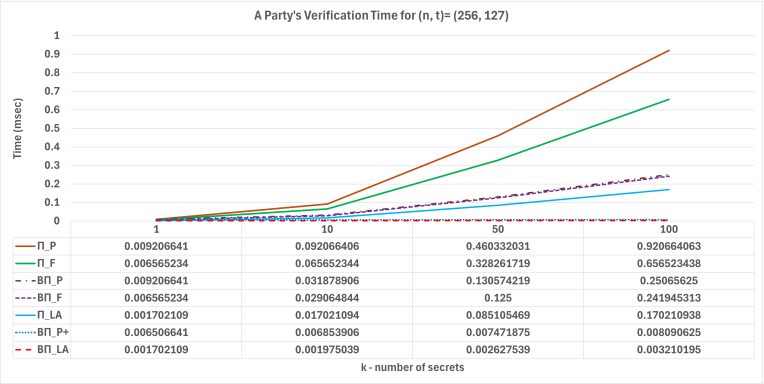
Sharing Time



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

- $B\Pi_{\mathrm{LA}}$ and $B\Pi_P^+$ track **plain Shamir** closely
- For $k > 100$ secrets: overhead below **3%** for $B\Pi_{\mathrm{LA}}$, around **8%** for $B\Pi_P^+$

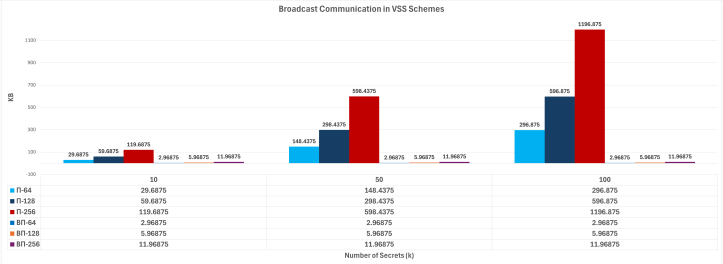
Verification Time



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

- $B\Pi_{\mathrm{LA}}$ verifies 100 shares in $\sim 3\,\mu$s; $B\Pi_P^+$ in $\sim 8\,\mu$s
- Both remain orders of magnitude faster than Π_F/Π_P run k times

Broadcast Communication



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

- Broadcast cost for Π schemes grows linearly with k; for $B\Pi$ schemes it stays (almost) flat
- Reduction factor of k in broadcast communication

Batched Pedersen vs. $\mathrm{B}\Pi_{\mathrm{P}}{}^{+}$

For $(n, t) = (256, 127)$, $k = 10$:

Scheme	Sharing	Verification
Shamir	2.95 ms	—
Pedersen (395% overhead)	14.61 ms	5.33 ms
Batched Pedersen (226% overhead)	9.62 ms	0.61 ms
$B\Pi_P^+$ (65% overhead)	4.89 ms	0.0068 ms

- $B\Pi$-based construction wins decisively in verification (near-Shamir sharing cost, orders-of-magnitude faster checks)
- Overhead of $B\Pi_P^+$ decreases further as k grows

$\mathrm{B}\Pi_S$: Batched PVSS Benchmarks

Sharing time of PVSS schemes	$k = 1$	$k = 10$	$k = 50$	$k = 100$	$k = 250$
Shamir+Enc	1.47	12.72	61.97	122.99	305.1
Π_{S}	2.77	27.76	138.80	277.60	694.1
$\mathrm{B}\Pi_{\mathrm{S}}$	2.77	14.34	63.80	125.30	308.9
Overhead of $\mathrm{B}\Pi_{\mathrm{S}}$ over baseline	88%	13%	3%	1.9%	1.2%

Single sharing: $\sim$88% overhead over the passive baseline; drops to **1.2%** at $k = 250$