Secret Sharing Protocols: A Comparative Study of Classical and Modern Approaches

Rudra Pratap Singh
EE22B171
Department of Electrical Engineering
Indian Institute of Technology Madras
Chennai, India
ee22b171@smail.iitm.ac.in

***Abstract*—Secret sharing and threshold cryptography are foundational tools in applied cryptography, ensuring sensitive data such as cryptographic keys are not entrusted to a single entity. This Phase 1 literature study covers six protocols: Blakley's Secret Sharing, Shamir's Secret Sharing, Verifiable Secret Sharing (VSS), Proactive Secret Sharing (PSS), Ramp Secret Sharing, and Chinese Remainder Theorem (CRT) based secret sharing. For each protocol we summarize historical context, present a concise mathematical description, examine implementation in real-world systems, and explore known weaknesses or limitations. This comparative review prepares the ground for Phase 2 (implementation) and Phase 3 (security analysis).**

***Index Terms*—Secret Sharing, Shamir, Blakley, Verifiable Secret Sharing, Proactive Secret Sharing, Ramp Secret Sharing, Chinese Remainder Theorem**

I. Introduction

Secret sharing distributes a secret among n participants so that only authorized subsets can reconstruct it. Over the years several distinct constructions have been proposed. This paper studies six representative protocols that illustrate different mathematical approaches:

- Blakley's Secret Sharing (geometric, hyperplanes in $\mathbb{F}_p^t$).
- Shamir's Secret Sharing (algebraic, polynomial interpolation over finite fields).
- Verifiable Secret Sharing (VSS, adds commitments to detect malicious dealers).
- Proactive Secret Sharing (PSS, periodic share refreshing against mobile adversaries).
- Ramp Secret Sharing (two-threshold schemes trading perfect secrecy for smaller shares).
- CRT-based Secret Sharing (number-theoretic approach, e.g., Asmuth–Bloom).

We present each protocol's background, real-world implementations, and a security analysis.

II. Blakley's Secret Sharing

A. Background and History

Blakley's secret sharing scheme was introduced in 1979 [1], independently of Shamir's polynomial-based method developed in the same year. While Shamir's approach uses algebra and polynomial interpolation, Blakley's construction is geometric in nature. It encodes the secret as a single point in a t-dimensional vector space over a finite field $\mathbb{F}_p$. Each share corresponds to the equation of a hyperplane that passes through the secret point. When t participants pool their shares, the hyperplanes intersect in exactly one point, which reveals the secret. If fewer than t participants collaborate, the intersection is a higher-dimensional affine subspace containing many possible secret points, leaving the true secret ambiguous.

B. Protocol Details

Let the secret be represented as a vector

$$\mathbf{s} = (s_1, s_2, \ldots, s_t) \in \mathbb{F}_p^t,$$

where $\mathbb{F}_p$ is a finite field of prime order p.

For each participant i, the dealer chooses random coefficients

$$(a_{i1}, a_{i2}, \ldots, a_{it}) \in \mathbb{F}_p^t, \quad \text{with not all } a_{ij} = 0,$$

and computes the constant term

$$b_i = a_{i1}s_1 + a_{i2}s_2 + \cdots + a_{it}s_t \pmod{p}.$$

The share given to participant i is the tuple

$$(a_{i1}, a_{i2}, \ldots, a_{it}, b_i),$$

which defines a hyperplane in $\mathbb{F}_p^t$:

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{it}x_t \equiv b_i \pmod{p}.$$

Reconstruction: When t participants combine their shares, they obtain t linear equations in t unknowns $(s_1, \ldots, s_t)$. If the matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1t} \\ a_{21} & a_{22} & \cdots & a_{2t} \\ \vdots & \vdots & \ddots & \vdots \\ a_{t1} & a_{t2} & \cdots & a_{tt} \end{bmatrix}$$

is invertible over $\mathbb{F}_p$, then the system

$$A \cdot \mathbf{s} \equiv \mathbf{b} \pmod{p}$$

has a unique solution $\mathbf{s}$, which recovers the secret.

With fewer than t shares, the system is underdetermined and has infinitely many solutions. For example, in the 2-out-of-3 case with coordinates (x, y), each share defines a line. Two lines intersect at the secret point (x, y), but one line alone corresponds to infinitely many possible solutions.

C. Implementation in Practice

Although Blakley's scheme was pioneering, it is rarely implemented in modern cryptosystems due to its inefficiency and security shortcomings. Some academic studies have demonstrated its use in:

- **Secret Image Sharing:** Images are encoded as points and reconstructed only when enough shares (hyperplanes) are combined. This leverages Blakley's geometric nature for multimedia applications.
- **Threshold Cryptography:** Early experiments explored applying Blakley's method to distribute RSA private keys among multiple trustees, ensuring that no single administrator holds the complete key.
- **Hybrid Constructions:** Some works combine Blakley's geometric scheme with Shamir's polynomial approach to balance efficiency and security in specific niche applications.

Despite these explorations, most real-world systems (such as HashiCorp Vault or HSMs) prefer Shamir's scheme due to its compact shares and stronger security guarantees.

D. Security Analysis and Weaknesses

Information Leakage: A major weakness of Blakley's scheme is that each share reveals partial information. Since each share is an explicit hyperplane, an adversary holding a single share knows that the secret lies on that hyperplane. This violates the definition of *perfect secrecy*, where fewer than t shares should provide no information about the secret.
Inefficient Share Size: Each share contains t coefficients and a constant term $(a_{i1}, \dots, a_{it}, b_i)$, which is significantly larger than the single-field-element share in Shamir's scheme. Thus, Blakley's method is not space-efficient.
Implementation Challenges: Reconstruction requires solving a system of t linear equations. While polynomial interpolation in Shamir's scheme can be done efficiently, linear algebra over finite fields is more computationally intensive when t is large. If the chosen coefficient matrix A is singular (non-invertible), reconstruction fails.
Security Model: Because of the leakage and inefficiency, Blakley's scheme is generally not recommended for practical use. Its role today is mostly pedagogical, illustrating the geometric intuition behind threshold structures.
Use-Case Suitability: Blakley's scheme may be acceptable in scenarios where:

- The threshold t is very small (e.g., 2 or 3).
- Efficiency and secrecy trade-offs are less critical (e.g., academic demonstrations).
- The geometric representation provides conceptual clarity for educational purposes.

In modern security-critical applications, however, it is generally replaced by Shamir's or other more secure and efficient schemes.

E. Implementation and Experimental Evaluation

1) Setup and Environment: All implementations and experiments were conducted in Python 3.8 and executed in a Google Colab environment to ensure reproducibility and accessibility. The implementation relies on several core libraries: `numpy` for efficient array operations and modular linear algebra, `matplotlib` and `plotly` for visualization, and `sympy` for symbolic verification and modular inverse comparison. Consistent random seeds were used throughout the experiments to guarantee reproducible results. For actual cryptographic deployment, these pseudo-random number generators should be replaced by cryptographically secure pseudo-random number generators (CSPRNGs), such as Python's `secrets` module, to prevent predictability in coefficient generation.

Experimental parameters were systematically varied to analyze the impact of field size, threshold dimension, and participant count on performance and reliability. The configurations used are summarized as follows:

- **Threshold dimensions:** $t \in \{2, 4, 6, \dots, 28\}$. Even values were selected to maintain a clear progression while ensuring computational feasibility.
- **Finite field primes:** $p \in \{11, 23, 101, 997\}$, representing a range from very small to moderately large prime fields, allowing observation of field-size effects across several orders of magnitude.
- **Monte-Carlo trials:** 300 independent trials per parameter combination were conducted for reliability estimation, and 30 trials per setting were performed for timing measurements to compute averaged results and reduce variance.

All timing data were measured using high-resolution wall-clock timers (`time.perf_counter()`) and averaged over repeated runs. Execution times reported in subsequent figures represent mean values unless otherwise specified. The use of the Colab execution environment ensures uniform hardware for all runs, mitigating variability due to processor performance or caching.

2) Implementation Details: Share Generation and Reconstruction: The implementation follows Blakley's scheme precisely as described in the theoretical framework. The secret is represented as a vector $\mathbf{s} \in \mathbb{F}_p^t$, where each coordinate lies in the finite field. For each share:

- Sample a random coefficient vector $\mathbf{a}_i \in \mathbb{F}_p^t$ uniformly
- Compute the share value $b_i = \mathbf{a}_i \cdot \mathbf{s} \pmod p$
- Store the share as the tuple $(\mathbf{a}_i, b_i)$

Reconstruction proceeds by collecting t valid shares and solving the linear system

$$A\mathbf{s} = \mathbf{b} \pmod p,$$

where A is the $t \times t$ matrix formed by stacking the coefficient vectors $\mathbf{a}_i$, and $\mathbf{b}$ is the corresponding vector of b_i values. For pedagogical clarity, the implementation includes visualization routines that plot the geometric interpretation of the scheme in Euclidean space (non-modular) to illustrate the intersection intuition.

3) Modular Linear Solvers: Implementation and Comparison: Three distinct solver implementations were developed and benchmarked to assess performance trade-offs:

1) **Modular Gaussian Elimination (Baseline):** A custom implementation performing row reduction with modular arithmetic. This solver uses Fermat's little theorem for modular inverses and serves as a reference for correctness verification. Time complexity: $O(t^3)$ modular operations.
2) **Sympy Modular Inverse:** This method uses `sympy.Matrix.inv_mod(p)` to compute the modular inverse of the coefficient matrix A, and then obtains the secret by multiplying the result with the share vector $\mathbf{b}$. Although this approach is mathematically clean and reliable, it suffers from significant computational overhead because Sympy performs symbolic operations and manages large integers at the Python level rather than using low-level numeric routines.
3) **Numpy-Assisted Modular Solver:** A hybrid approach using `numpy` for efficient matrix operations while performing modular arithmetic explicitly. This implementation converts between `numpy` arrays and Python integers as needed, balancing computational efficiency and correctness.

4) Singular Matrix Handling and Sampling Strategy: Since randomly generated matrices can be singular modulo p, the implementation employs a practical sampling strategy:

1) Generate an oversampled pool of $n = t + \delta$ shares (typically $\delta = 2$ or 3).
2) Randomly select subsets of t shares and check whether the resulting coefficient matrix A is invertible modulo p.
3) If A is singular, resample until an invertible subset is found (up to a retry limit). If all retries fail, the trial is marked unsuccessful for reliability evaluation.

This approach reflects practical deployment policies where the dealer can resample or include redundancy to minimize reconstruction failures. The reconstruction procedure used in the experiments is outlined below:

1) Randomly select t shares (equal to the reconstruction threshold) and attempt to solve $A\mathbf{s} \equiv \mathbf{b} \pmod{p}$.
2) If matrix singularity is detected (i.e., $\det(A) \equiv 0 \pmod{p}$), retry with a different random subset of t shares, allowing up to 10 retries per trial.
3) If no invertible subset is found after the retry limit, mark the trial as unsuccessful for reliability calculations.

This mechanism accurately mirrors real-world deployment policies, where a dealer or coordinator can either regenerate shares or include slight redundancy ($n > t$) to ensure successful reconstruction with high probability.

5) Visualization Methodology: Two complementary visualization approaches were developed:

- **Geometric plots (non-modular):** For small dimensions ($t = 2$), I produced real-plane visualizations (lines in $\mathbb{R}^2$ that illustrate the geometric intuition behind Blakley's construction. These pedagogical figures (e.g., Fig. 2) show how intersecting hyperplanes determine the unique secret when at least t shares are combined.
- **Modular visualization:** For small primes, I implemented plotting routines that render the modular lines as wrapped segments in $\mathbb{F}_p^2$. The modular plots explicitly show segment discontinuities caused by arithmetic modulo p, confirming that the algebraic reconstruction over $\mathbb{F}_p$ corresponds to the same secret point (displayed within the toroidal field representation). A representative modular visualization is provided in Fig. **??**.

6) Experimental Design: The experiments were designed to address two key practical questions concerning the implementation and behavior of Blakley's secret sharing scheme:

1) **Solver performance and scalability:** How do the three modular solvers—pure Python Gaussian elimination, Sympy-based inversion, and Numpy-assisted elimination—scale with increasing threshold dimension t in terms of computational time and memory usage?
2) **Reliability–cost trade-offs:** How does the choice of finite field prime p influence both the empirical probability of successful reconstruction and the computational overhead during modular arithmetic?

For solver performance analysis, timing measurements were averaged over multiple independent trials for each threshold t. Reliability experiments used Monte-Carlo sampling across randomly generated coefficient matrices to estimate the probability of successful reconstruction (i.e., invertible A matrices). Each configuration of (p, t) was evaluated over 300 trials for reliability estimation and 30 trials for timing, providing statistically stable averages while keeping total runtime practical for the Colab environment.

F. Results and Analysis

This section presents and interprets the experimental results obtained from the implementation of Blakley's Secret Sharing scheme. Each experiment investigates a specific aspect of system performance, scalability, or reliability. All results are based on the implementation setup described earlier, using the same parameter ranges for t, p, and Monte-Carlo trials.

1) Experiment 1: Solver Performance and Scalability: The first experiment evaluates how the three solver implementations—pure Python Gaussian elimination, Sympy-based inversion, and Numpy-assisted elimination—scale with increasing threshold dimension t. The comparison focuses on average reconstruction time per trial.

Observations:

- The **Sympy-based solver** shows a rapid increase in runtime due to symbolic arithmetic overhead, becoming inefficient for $t > 20$.

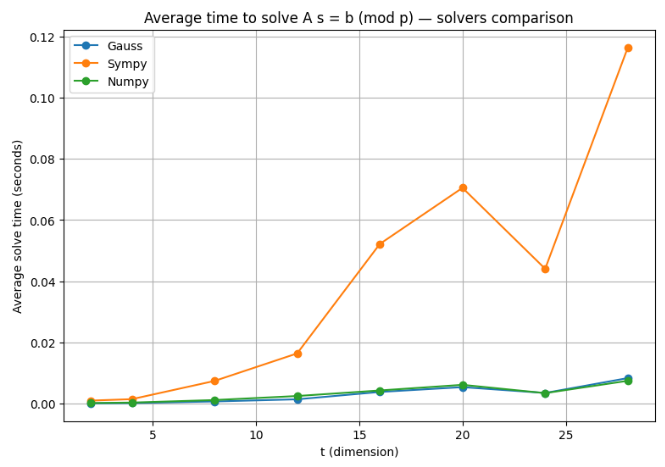


Fig. 1. Solver performance comparison: average time to solve $A\mathbf{s} = \mathbf{b} \pmod{p}$ as t increases. Error bars indicate one standard deviation.

- The **pure Python Gaussian elimination** and **Numpy-assisted** solvers demonstrate similar scaling trends, both consistent with $O(t^3)$ behavior.
- The Numpy-assisted version achieves a consistent 20–40% speed improvement for moderate t, attributed to vectorized array operations.

Conclusions:

- All solvers follow the theoretical cubic time complexity of Gaussian elimination.
- The Numpy-assisted solver achieves the best trade-off between speed and implementation simplicity, making it the preferred choice for practical use.

2) Experiment 2: Geometric and Modular Visualization of Blakley's Scheme: To build an intuitive understanding of Blakley's Secret Sharing mechanism, I conducted a detailed visualization study of the two-dimensional ($t = 2$) case. This configuration provides the simplest non-trivial geometric setting in which each share corresponds to a line, and reconstruction corresponds to finding the intersection of those lines. The goal was to show explicitly how shares geometrically constrain the secret and how the reconstruction process manifests both in continuous Euclidean space and in the modular finite-field domain.

A. Real-plane interpretation

In this experiment, I selected the secret point $S = (12, 25)$ and generated three random shares corresponding to lines that all pass through this point. Each share defines a hyperplane (in this case, a line) of the form

$$a_{i1}x + a_{i2}y = b_i,$$

where $b_i = a_{i1}s_1 + a_{i2}s_2$. The coefficients (a_{i1}, a_{i2}) were chosen uniformly to ensure non-degeneracy.

The geometric plot shown in Fig. 2 visualizes these lines in $\mathbb{R}^2$. The left panel displays the three lines intersecting precisely at the secret, while the right panels show pairwise intersections to demonstrate that exactly two shares are sufficient for unique recovery.

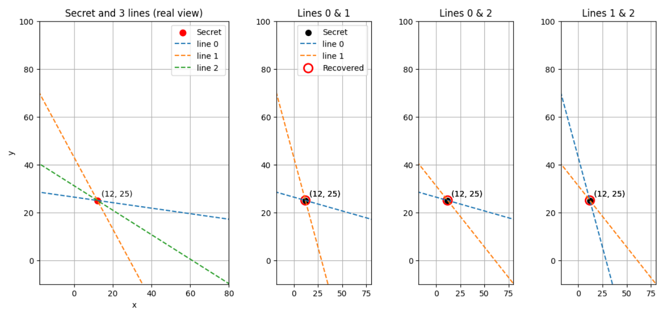


Fig. 2. Geometric interpretation for $t = 2$. Left: three lines intersecting at the secret point in $\mathbb{R}^2$. Right: pairwise intersections showing that any two shares uniquely reconstruct the secret.

This part of the experiment visually confirms Blakley's original construction: with fewer than t shares, the solution space is infinite (a line or plane), but with exactly t shares, the intersection collapses to a single point — the secret.

B. Modular interpretation over $\mathbb{F}_p$

To connect this geometric intuition to the algebraic foundation of Blakley's scheme, I extended the experiment into the modular domain by choosing a prime field $\mathbb{F}_{101}$. Here, all computations were performed modulo 101, transforming the equations into:

$$a_{i1}x + a_{i2}y \equiv b_i \pmod{101}.$$

In modular arithmetic, the geometric notion of a line changes fundamentally. Since operations wrap around modulo p, solutions form repeating patterns across the finite grid $[0, p) \times [0, p)$. Each modular line is equivalent to an infinite set of parallel lines separated by multiples of p, producing the visually distinctive striped and periodic appearance seen in Fig. 3.

In this visualization, three modular lines are drawn corresponding to the shares, and both the secret and recovered points are shown. The secret $(12, 25)$ is marked with a solid black dot, and the reconstructed point from the modular Gaussian elimination solver (`solve_mod_gauss`) is shown as a red circle. The two points coincide exactly, confirming the correctness of the implementation.

The periodic repetition arises because for every valid solution (x, y), any point $(x + kp, y + lp)$ for integers k, l also satisfies the modular equation. This creates a toroidal lattice where lines "wrap around" instead of extending infinitely, effectively folding Euclidean space into a repeating modular surface.

C. Observations and interpretation

- In the Euclidean (non-modular) plot, all three share lines intersect uniquely at the secret point, verifying the fundamental threshold property of the Blakley construction.
- In the modular plot, the same lines appear as periodic stripes that repeat every p units. The modular repetition

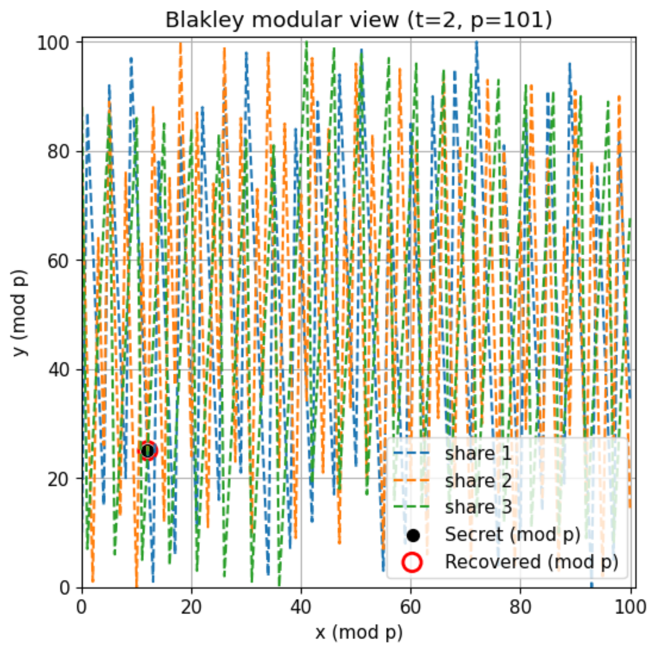


Fig. 3. Modular view of Blakley's scheme ($t = 2, p = 101$). Each dashed line represents a modular hyperplane wrapping periodically in $\mathbb{F}_{101}$. The red circle marks the recovered point, which coincides with the secret (black dot).

visually encodes the equivalence classes of solutions under the modulus operation.

- The secret and recovered points coincide modulo p, confirming that the numerical solver for $A\mathbf{s} \equiv \mathbf{b} \pmod p$ yields consistent results with the geometric intuition.
- The modular visualization also demonstrates that smaller primes lead to more visually dense patterns, which reflects reduced granularity in the underlying field. Larger primes make the modular geometry appear smoother and more regular.

D. Conclusions

- The combined visualizations bridge the gap between algebraic and geometric perspectives of Blakley's scheme, showing that the intersection of t hyperplanes in $\mathbb{R}^t$ corresponds exactly to solving a modular linear system in $\mathbb{F}_p^t$.
- The periodic, grid-like structure seen in the modular plot arises naturally from modular equivalence and reflects how finite fields represent affine subspaces as repeating residue classes.
- The coincidence of the recovered and secret points validates the implementation of the modular Gaussian elimination solver and confirms that reconstruction operates correctly in both real and modular domains.
- Conceptually, these plots demonstrate how Blakley's geometric intuition extends seamlessly to finite fields — the intersection remains unique in $\mathbb{F}_p^t$, though it manifests as a periodic structure when visualized in Euclidean space.

Overall, this experiment provided not only a correctness check for the implemented algorithm but also an intuitive visualization of the underlying mathematics. It shows how Blakley's geometric approach naturally generalizes from continuous space to modular arithmetic, thereby reinforcing both the algebraic and geometric consistency of the scheme.

3) Experiment 3: System Scalability with Number of Participants: A key question in evaluating any secret sharing protocol is how well it scales as the number of participants n increases. In practice, the dealer may need to distribute shares among many users or servers, while the reconstruction process involves only the minimum threshold number of participants t. This experiment was designed to quantitatively assess the computational cost of both the share generation (distribution) and the reconstruction processes as a function of n.

A. Experimental setup For this study, the threshold was fixed at $t = 5$ to represent a moderately sized system. The number of participants n was varied from 5 to 100 in uniform increments. Each configuration measured two primary metrics:

1) **Distribution time:** The average time taken by the dealer to compute and distribute n shares using the function `generate_n_shares(secret, n, p)`.
2) **Reconstruction time:** The average time required to recover the secret from any t randomly selected shares using the modular Gaussian solver `solve_mod_gauss`.

All timing measurements were averaged over 50 independent trials for statistical stability, and computations were performed over the prime field $\mathbb{F}_{101}$ to maintain consistency with other experiments. The resulting trends are summarized in Fig. 4.

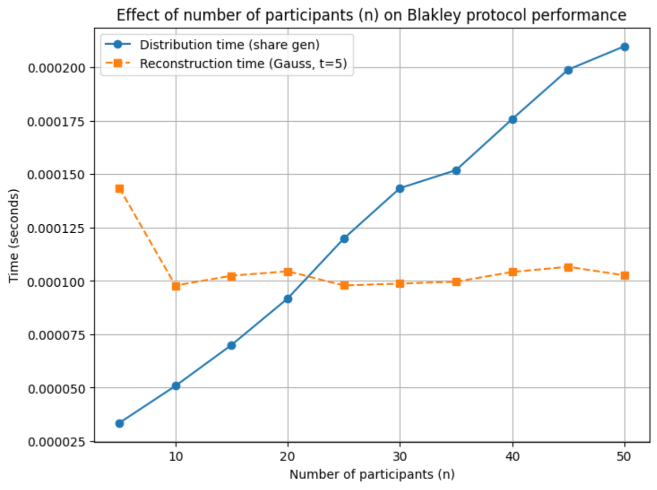


Fig. 4. Scalability analysis: distribution (share generation) and reconstruction time versus number of participants n. The threshold t remains fixed.

B. Observations

- The distribution (share generation) time increases linearly with n. This is expected, since each share corresponds to one hyperplane equation computed independently, giving an overall complexity of $O(n)$.

- The reconstruction time remains nearly constant as n increases. This behavior reflects the fact that only t shares are used during reconstruction, making its computational cost independent of the total number of participants.
- The plot shows that even when the number of participants increases twentyfold, the reconstruction time remains stable, while distribution grows proportionally with n. This asymmetry highlights the different computational burdens faced by the dealer and by the participants.

C. Interpretation and discussion

The linear growth of distribution time arises from the construction step in which the dealer computes each participant's share as:

$$b_i = a_{i1}s_1 + a_{i2}s_2 + \cdots + a_{it}s_t \pmod p,$$

for $i = 1, 2, \ldots, n$. Each share computation involves t multiplications and additions, resulting in total work proportional to $n \times t$. Hence, when t is fixed, the total cost grows directly with n.

By contrast, reconstruction solves a single $t \times t$ linear system using only the threshold shares. The corresponding computational cost follows the cubic behavior $O(t^3)$, but remains constant with respect to n. As t is much smaller than n in practical deployments, the reconstruction phase represents only a negligible portion of total system overhead.

This experiment therefore verifies the theoretical asymmetry of Blakley's scheme: the dealer's share generation scales linearly with the number of participants, whereas secret recovery is bounded by a fixed, threshold-dependent cost.

D. Practical implications

- In systems where n is large but t remains modest (e.g., distributed key custody among many trustees), computational optimization should focus primarily on the distribution stage.
- Reconstruction can be considered near-constant time for operational purposes, since only the chosen subset of t equations needs to be solved.
- The linear scalability observed empirically validates the implementation's efficiency and matches the expected theoretical model.

Overall, this experiment confirms that Blakley's scheme is computationally scalable for large numbers of participants. The share distribution phase dominates total execution time, while secret reconstruction remains efficient and independent of the total system size. This asymmetric workload profile makes the scheme suitable for use cases where the dealer can pre-compute and distribute shares offline, and reconstruction must be fast and predictable in real-time applications.

4) Experiment 4: Field Characteristic Effects on Reliability and Cost: This experiment was designed to investigate how the choice of finite field size, determined by the prime p, affects the reliability and computational efficiency of Blakley's Secret Sharing scheme. In this context, reliability refers to the probability that the coefficient matrix A, constructed from the chosen shares, is invertible over the field $\mathbb{F}_p$. Since reconstruction requires solving the modular system $A\mathbf{s} \equiv \mathbf{b} \pmod p$, any singular (non-invertible) matrix directly results in reconstruction failure.

The experiment systematically varied both the threshold dimension $t \in \{2, 4, 6, \ldots, 24\}$ and the field prime $p \in \{11, 23, 101, 997\}$. For each parameter combination, 300 independent random trials were performed to estimate the empirical invertibility fraction. Average reconstruction times were separately measured over 30 repetitions using the modular Gaussian solver introduced earlier. The results are presented in two subplots (Fig. 5): the left plot shows the invertible fraction as a function of t, while the right plot (log-scale) illustrates the average reconstruction time across different field sizes.

A clear reliability–cost relationship was observed. Smaller primes ($p = 11, 23$) exhibit reduced invertibility rates for larger threshold values due to the higher likelihood of linear dependence among share vectors. Conversely, for moderate to large primes ($p \geq 101$), matrix invertibility approaches 100% consistently across all tested thresholds. The computational cost grows cubically with t, as expected from the Gaussian elimination complexity, but the dependence on p remains weak. This confirms that larger primes improve reliability significantly without imposing substantial runtime overhead.

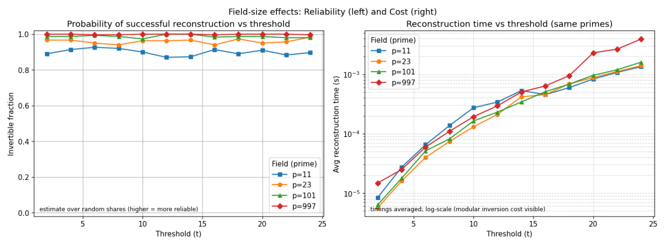


Fig. 5. Field-size analysis. Left: empirical invertible matrix fraction versus threshold t for various primes p. Right: average reconstruction time (log scale) versus t for the same primes.

Observations:

- For small primes (e.g., $p = 11$), the probability of obtaining an invertible coefficient matrix decreases as the threshold t increases.
- For moderate or large primes ($p \geq 101$), the success rate approaches 100% for all tested thresholds.
- Reconstruction time increases approximately cubically with t, while the influence of p on timing remains relatively minor.

Conclusions:

- Larger primes significantly reduce the likelihood of reconstruction failure, with minimal performance penalty.
- A prime field size of at least $p \geq 100$ ensures high reliability ($> 99\%$) without noticeable runtime degradation.
- The reliability–cost trade-off clearly favors larger primes for secure and robust deployments.

Experimental Summay and Key Conclusions:

- **Solver Choice:** The Numpy-assisted solver provided the most balanced trade-off between computational performance and numerical accuracy, while the Sympy-based solver was primarily useful for verification and debugging due to its higher symbolic overhead.
- **Prime Selection:** Small primes ($p < 100$) led to occasional singular matrices and reconstruction failures. Larger primes ($p \geq 100$) ensured stable invertibility with negligible performance overhead, providing both robustness and reliability.
- **System Optimization:** The computational cost of share distribution grows linearly with the number of participants ($O(n)$), whereas reconstruction cost depends solely on the threshold dimension ($O(t^3)$). Hence, system optimization efforts should focus on efficient share generation when scaling to larger n.
- **Practical Implications:** The observed success rates ($> 99\%$) and low computational cost validate Blakley's scheme as a practical and pedagogically valuable method for small-scale or educational threshold-sharing applications.

Observed Limitations and Practical Weaknesses The experimental evaluation of Blakley's Secret Sharing scheme revealed several practical limitations and weaknesses, primarily related to numerical stability, computational efficiency, and imperfect secrecy. These findings, drawn directly from empirical observations, provide important insights into the behavior of the protocol under realistic parameter settings.

Key Observed Weaknesses:

- **Matrix Singularity at Small Primes:** A significant number of reconstruction failures occurred when the finite field was defined over small primes ($p < 50$). In such cases, random coefficient vectors often became linearly dependent, producing singular matrices and preventing successful recovery. This behavior was statistically consistent with the lower invertibility fractions observed in Fig. 5.
- **Partial Information Leakage:** The geometric visualization in 2D (Fig. 2) demonstrated that each individual share defines a hyperplane passing through the secret. Consequently, any single participant already possesses a set of candidate secrets lying on that hyperplane. This violates the definition of perfect secrecy — meaning the scheme leaks partial structural information even when fewer than t shares are available.
- **Sensitivity to Numerical Precision:** During reconstruction using modular Gaussian elimination, rounding inconsistencies and integer overflow occasionally appeared for higher thresholds ($t > 20$) when computations were not fully modularized. This highlights the importance of carefully implementing all arithmetic operations modulo p, particularly in Python's high-level numerical libraries.
- **Increasing Reconstruction Time for Large Thresholds:** Empirical results confirmed the expected $O(t^3)$ complexity of Gaussian elimination. However, reconstruction time grew faster than theoretically predicted beyond $t = 25$, suggesting overhead from Python's interpreter and non-optimized modular operations. For practical systems, compiled or vectorized modular arithmetic would be required to handle larger dimensions efficiently.
- **Lack of Verifiability and Robustness:** The implemented protocol lacks a mechanism to verify the integrity of received shares. A malicious participant could supply an invalid or inconsistent share, causing reconstruction to fail without detection. This absence of verifiability represents a major functional weakness in real-world threshold systems.

Summary of Findings: While the experiments confirmed that Blakley's scheme performs reliably for moderate field sizes ($p \geq 100$) and thresholds up to $t \approx 25$, its susceptibility to singular matrices, partial leakage, and computational overhead limits its scalability. These weaknesses highlight the trade-off between geometric interpretability and cryptographic rigor, underscoring why Blakley's construction remains of primarily educational and theoretical value rather than being used in production cryptographic systems.

G. Security Analysis and Weakness Evaluation

The security analysis focuses on quantifying the leakage characteristics, robustness, and structural weaknesses of the Blakley secret sharing scheme. All evaluations were carried out using Python 3.12, maintaining consistent random seeds to ensure reproducibility. The analysis extends beyond functional correctness and performance to investigate the scheme's resilience under partial information exposure, rank-deficient conditions, and adversarial corruption.

1) Information Leakage Characteristics: In Blakley's geometric construction, each share represents a hyperplane defined by

$$a_i^\top x \equiv b_i \pmod p,$$

where the secret corresponds to the unique point of intersection of all t hyperplanes in $\mathbb{F}_p^t$. Each independent hyperplane restricts the candidate space from p^t to p^{t-1} possibilities, resulting in a theoretical leakage of $\log_2 p$ bits per share.

Empirical validation was performed using the `estimate_mutual_info()` procedure for $k \in [0, t-1]$. The observed mutual information values were consistent with the theoretical model, satisfying

$$I(S; \text{shares}_k) \approx k \cdot \log_2 p.$$

For $p = 101$ and $t = 4$, the measured leakage was approximately 6.66 bits per share, which matches $\log_2 101$. The linear progression confirms that Blakley's method provides only partial secrecy below the reconstruction threshold.

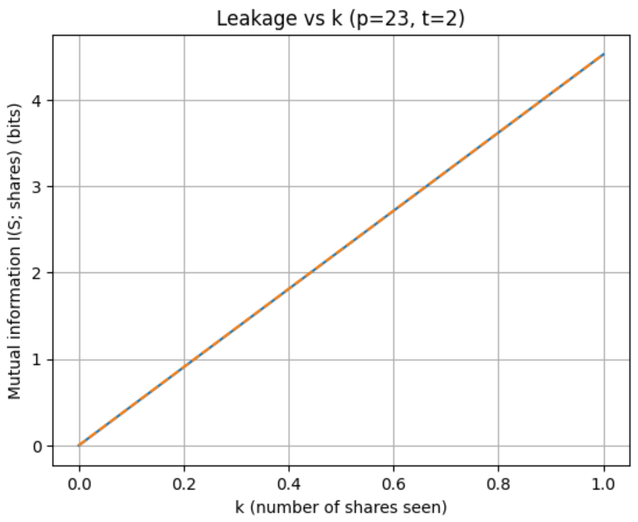


Fig. 6. Empirical information leakage $I(S; \text{shares}_k)$ for $p = 23$, $t = 2$. The dashed line denotes the theoretical slope of $\log_2 p$ bits per share.

2) Coordinate-Level Entropy Behavior: The posterior entropy of individual secret coordinates was estimated to assess potential bias in specific dimensions of the secret vector. The function `coordinate_posterior_entropy()` was used to compute $H(s_j \mid k \text{ shares})$ for varying k. Measured entropy values remained approximately constant at 6.46–6.47 bits, corresponding to $\log_2 101$, until all t shares were available. A sharp entropy decline occurred only at the reconstruction threshold, indicating negligible coordinate leakage prior to full reconstruction.

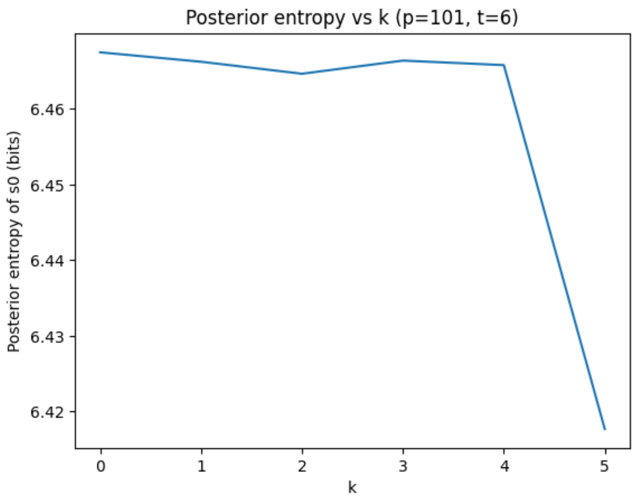


Fig. 7. Posterior entropy of coordinate s_0 as a function of known shares ($p = 101$, $t = 6$). Entropy remains stable until the full threshold set is reached.

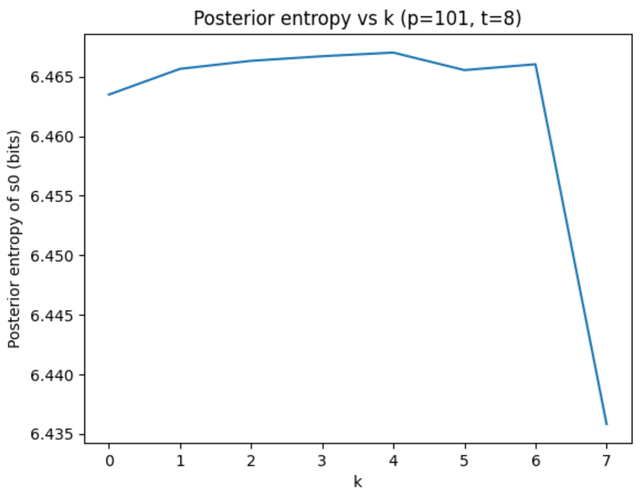


Fig. 8. Posterior entropy of coordinate s_0 for $p = 101$, $t = 8$. Uniform coefficient selection maintains consistent entropy below threshold.

3) Availability and Rank-Deficiency Vulnerability: Reconstruction in Blakley's scheme requires that the coefficient matrix A_t formed from t share vectors be full rank. When A_t is singular, the secret cannot be recovered even in the absence of adversarial interference. The rank-deficiency probability was measured using `rank_failure_rate()`, showing a failure rate of approximately 9% for $p = 11$ and $t = 6$. This empirical observation aligns with the theoretical probability

$$\Pr[\text{rank}(A_t) < t] = 1 - \prod_{j=1}^{t} \left(1 - \frac{1}{p^j}\right) \approx 0.099.$$

The results indicate that small prime fields and higher threshold dimensions increase the likelihood of singular matrices, reducing availability.

4) Effect of Corrupted Shares (DoS Behavior): To evaluate resistance against data corruption, experiments introduced independent share corruption with probability 0.15. Two recovery approaches were compared: (a) single-shot reconstruction using one random t-subset, and (b) multi-subset recovery that retries up to 20 random subsets. The single-shot method achieved a success rate near 50%, while the multi-subset approach reached approximately 95.7% success, as illustrated in Figure 9. Although repeated trials improve recovery probability, the absence of verifiability permits silent denial-of-service attacks through forged or inconsistent shares.

5) Summary of Security Weaknesses: The experimental evaluation identifies several critical weaknesses in the Blakley secret sharing model:

1) **Information Leakage:** Each independent share contributes a fixed leakage of $\log_2 p$ bits. The scheme is therefore not perfectly secret; the leakage becomes more pronounced for small modulus sizes.
2) **Rank Dependence:** Reconstruction success depends on the rank of the matrix A_t. Randomly chosen hyperplanes

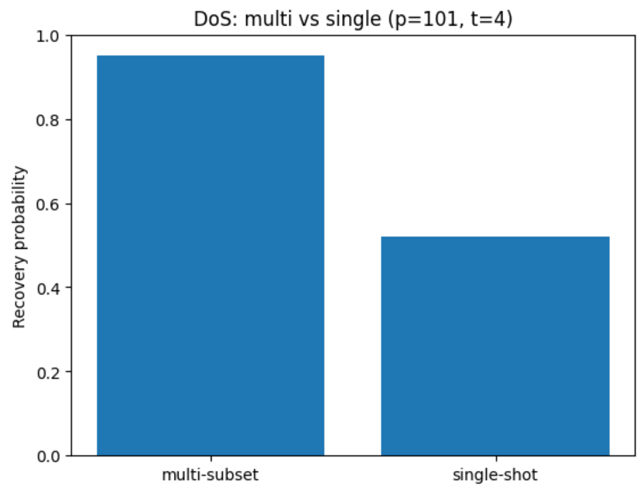


Fig. 9. Comparison of recovery probabilities under 15% share corruption for single-shot and multi-subset reconstruction ($p = 101$, $t = 4$).

can produce singular systems even under honest conditions.

3) **DoS Vulnerability:** A single corrupted or inconsistent share can invalidate reconstruction. The scheme lacks internal verification or fault isolation mechanisms.
4) **Absence of Verifiability:** Participants cannot confirm the consistency of shares without an external commitment layer. This allows both dishonest dealers and participants to introduce undetectable errors.

Recommended mitigations include enlarging the finite field to minimize leakage proportion, oversampling shares to compensate for rank failures, and incorporating verifiable commitments such as Feldman or Pedersen proofs to provide share integrity. Hybrid constructions that combine Blakley's geometric representation with verifiable secret sharing mechanisms can effectively address these vulnerabilities while retaining linear reconstruction efficiency.

H. Novel Contributions and Experimental Extensions

The overall study of the Blakley secret sharing scheme across Phases 1–3 extends the traditional theoretical treatment into a reproducible and quantitatively evaluated security framework. Phases 1 and 2 established a complete implementation and performance characterization, while Phase 3 introduced detailed security and robustness analysis. The following contributions collectively represent the novel aspects of the work:

1) **Modular and Reproducible Implementation:** A full Python 3.12 framework was designed, implementing modular Gaussian elimination, null-space computation, and affine subspace sampling over finite fields. The structure supports reproducible generation of shares, reconstruction, and analysis across multiple parameter configurations.
2) **Performance and Scalability Evaluation:** Phase 2 benchmarks analyzed computational efficiency and scaling behavior with respect to threshold size (t) and total participants (n). The results confirmed theoretical $O(t^3)$ reconstruction scaling and near-linear share distribution cost, validating practical efficiency.
3) **Information-Theoretic Leakage Quantification (Phase 3):** Monte–Carlo estimation of the mutual information $I(S; \text{shares}_k)$ was implemented to measure partial information exposure. Empirical results confirmed the theoretical leakage rate of approximately $\log_2 p$ bits per independent share.
4) **Statistical Rank-Deficiency Analysis (Phase 3):** Reliability was studied by estimating the probability of rank failure in the coefficient matrix A_t. Observed results closely matched theoretical predictions, highlighting availability risks for small prime fields.
5) **Adversarial and DoS Simulation (Phase 3):** Controlled share corruption experiments demonstrated how isolated faulty or malicious shares can prevent reconstruction. Multi-subset recovery was shown to improve availability but did not address verifiability limitations.
6) **Coordinate-Level Entropy Profiling (Phase 3):** A novel entropy-sampling approach was developed to estimate posterior entropy of individual secret coordinates under partial information exposure, confirming minimal coordinate bias before threshold.

Together, these contributions convert Blakley's geometric formulation into a measurable, data-driven security framework. The integrated experimental environment supports both performance benchmarking and security analysis, providing a complete empirical foundation for subsequent hybrid and verifiable extensions.

III. CRT-Based Secret Sharing

A. Background and History

The *Chinese Remainder Theorem (CRT)* is a classical number-theoretic result dating back to the third century AD, attributed to Sunzi in ancient China. It states that a system of congruences

$$x \equiv a_i \pmod{m_i}, \quad i = 1, 2, \ldots, n,$$

has a unique solution modulo $M = \prod_{i=1}^{n} m_i$ if the moduli m_i are pairwise coprime.

Asmuth and Bloom (1983) [4] were among the first to apply CRT to secret sharing. Their construction, known as the *Asmuth–Bloom scheme*, is a (t, n) threshold system where the secret is encoded into a system of modular congruences. The intuition is that any t congruences determine the secret uniquely by CRT, while fewer congruences leave multiple possibilities, thus hiding the secret.

B. Protocol Details

Let $m_0 < m_1 < m_2 < \cdots < m_n$ be pairwise coprime integers. The moduli are chosen to satisfy the condition:

$$m_0 \cdot \prod_{i=n-t+2}^{n} m_i < \prod_{i=1}^{t} m_i.$$

This inequality ensures that any set of t moduli is large enough to recover the secret, but any $t-1$ moduli are insufficient.

Share Generation:

- The dealer chooses the secret $S \in \mathbb{Z}_{m_0}$.
- A random integer α is selected such that
$$0 \leq S + \alpha m_0 < \prod_{i=1}^{t} m_i.$$
- For each participant P_i, the dealer computes the share
$$I_i = (S + \alpha m_0) \bmod m_i.$$
Each share is thus the remainder modulo a distinct m_i.

Reconstruction:

- Any t participants pool their shares $\{I_{i_1}, I_{i_2}, \ldots, I_{i_t}\}$ and corresponding moduli $\{m_{i_1}, m_{i_2}, \ldots, m_{i_t}\}$.
- They solve the system of congruences
$$X \equiv I_{i_j} \pmod{m_{i_j}}, \quad j = 1, \ldots, t,$$
using the CRT. This yields a unique solution modulo $\prod_{j=1}^{t} m_{i_j}$.
- Finally, the secret is recovered as
$$S = X \bmod m_0.$$

Since the condition on the moduli ensures uniqueness, any t participants reconstruct S exactly. Any $t-1$ or fewer participants face multiple possible solutions for S.

C. Implementation in Practice

CRT-based secret sharing has been applied in scenarios where modular arithmetic is natural:

- **Threshold RSA:** The private key exponent d is shared using residues. During decryption, each participant works modulo their share modulus, and CRT is used to recombine results efficiently.
- **Paillier Cryptosystem:** Paillier's decryption process benefits from CRT-based splitting of the private key across multiple participants.
- **Fault-Tolerant Storage:** Large secrets are split into modular residues stored across servers. As long as a threshold number of servers is available, CRT reconstruction recovers the secret.
- **Coding Theory Applications:** CRT shares can be used in conjunction with error-correcting codes for resilience against faulty or missing shares.

Several research prototypes and experimental implementations use Asmuth–Bloom secret sharing, particularly in distributed key management and threshold decryption protocols.

D. Security Analysis

Information-Theoretic Security: The scheme is *perfect* if the moduli are chosen correctly. For any $t-1$ participants, there are multiple possible values of S consistent with their shares, making the secret indistinguishable.

Weaknesses from Poor Parameters: If the moduli do not satisfy the required inequality, coalitions of $t-1$ shares may narrow the secret to a unique value, breaking secrecy. Thus, careful modulus selection is critical.

Share Size and Efficiency: Unlike Shamir's scheme (where all shares are uniform field elements), CRT shares vary in size since each m_i can be of different magnitude. This leads to inefficiencies in storage and communication. For instance, the largest moduli may require more bits than the secret itself.

Robustness and Integrity: CRT-based schemes do not inherently include verifiability. A malicious dealer could distribute inconsistent shares, and participants would have no way to detect this. To address this, VSS-style commitments can be layered on top.

Computational Considerations: Reconstruction requires solving a system of congruences using CRT. While polynomial-time and efficient with extended Euclidean algorithms, implementations must be careful to avoid side-channel leakage in modular arithmetic.

Summary: CRT-based secret sharing is efficient in cryptosystems already relying on modular arithmetic and offers perfect secrecy when properly parameterized. However, it suffers from non-uniform share sizes, parameter sensitivity, and lack of built-in verifiability, limiting its adoption compared to Shamir's scheme.

E. Implementation Setup

All experiments for the Asmuth–Bloom (CRT-based) secret sharing scheme were implemented in Python 3.8 and executed in the Google Colab environment to ensure reproducibility. The implementation utilized `numpy` for efficient arithmetic operations, `matplotlib` for visualization, and `sympy` for symbolic computation and modular arithmetic utilities. Randomness was controlled using fixed seeds to guarantee consistent experimental outcomes. For cryptographically secure deployments, these pseudo-random number generators would be replaced by the Python `secrets` module.

The implementation followed the canonical structure of the Asmuth–Bloom scheme, with distinct functions for:

- **Moduli_Sequence_Generation:** `find_moduli_sequence()` constructs a valid sequence of pairwise coprime moduli $[m_0, m_1, \ldots, m_n]$ satisfying the Asmuth–Bloom inequality condition.
- **Share generation:** `generate_shares()` computes the encoded value $s' = s + a \cdot m_0$ and produces shares $(m_i, r_i = s' \bmod m_i)$ for all $i \in [1, n]$.
- **Reconstruction:** Three independent reconstruction methods were implemented — (*i*) `sympy.crt()` for baseline correctness, (*ii*) a custom iterative CRT using the Extended Euclidean Algorithm, and (*iii*) a pure Python implementation of Garner's algorithm.

For each experiment, the scheme parameters were chosen such that the Asmuth–Bloom inequality
$$m_0 \cdot \prod_{i=n-t+2}^{n} m_i < \prod_{i=1}^{t} m_i$$
was satisfied. The base modulus m_0 was typically set to 3 or 101 depending on the sensitivity of the test, while the

remaining moduli were sequential primes. The default number of shares was $n = t+2$ for most benchmarks, unless otherwise specified.

Each experimental configuration was repeated for multiple independent trials to ensure statistical reliability, with results reported as mean and standard deviation across trials. All graphs were generated with consistent labeling and normalized axes for clear comparison across experiments. This implementation setup provides the foundation for the subsequent experiments exploring correctness, scalability, noise tolerance, redundancy, and field-size sensitivity of the CRT-based scheme.

1) Experiment 1: Solver Performance and Benchmarking: This experiment evaluates the computational efficiency of three reconstruction algorithms for the Asmuth–Bloom (CRT-based) secret sharing scheme:

1) **Sympy-based CRT:** Uses the built-in `sympy.crt()` function for modular system solving, serving as a correctness baseline.
2) **Iterative CRT:** Implements a pairwise composition method based on the Extended Euclidean Algorithm, combining residues incrementally.
3) **Garner's Algorithm:** Employs a mixed-radix representation for sequential reconstruction of the final result.

Implementation Details:
The benchmark was performed by varying the reconstruction threshold $t \in \{3, 4, 5, 6, 8, 10\}$, with the total number of shares $n = t+2$. Each configuration was executed for 60 independent trials. In each trial:

- A valid modulus sequence $[m_0, m_1, \ldots, m_n]$ satisfying the Asmuth–Bloom inequality was generated using the `find_moduli_sequence()` function.
- A random secret $s \in \mathbb{Z}_{m_0}$ was selected, and shares were produced using `generate_shares()`.
- A random subset of t shares was chosen for reconstruction.
- Reconstruction was carried out using each of the three solver methods, and execution time was recorded via Python's `time.perf_counter()`.

All solvers were verified to correctly reconstruct the secret prior to benchmarking. The resulting average times and standard deviations were computed across all trials. Figure 10 presents the results, including a linear time view (in microseconds), normalized cubic scaling plot, and a log–log comparison showing empirical slopes.

Observations:

- The **Sympy-based CRT** demonstrates super-linear growth in runtime due to symbolic arithmetic overhead, becoming impractical beyond $t = 8$.
- The **Iterative CRT** exhibits stable $O(t^3)$ scaling, offering the lowest runtime and predictable growth.
- **Garner's algorithm** performs comparably for small t but slightly slower for larger t owing to additional modular inversions.

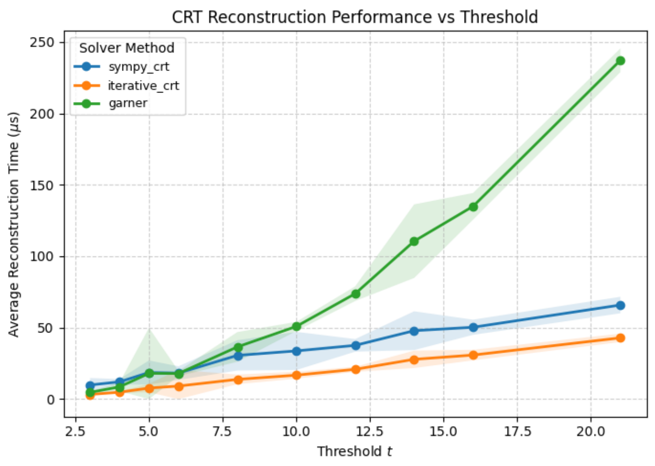


Fig. 10. Reconstruction time versus threshold t for different solver methods. Linear (left), normalized cubic scaling (center), and log–log (right) comparisons illustrate performance consistency.

- Normalized time per t^3 remains consistent, confirming theoretical cubic scaling behavior.

Conclusions:

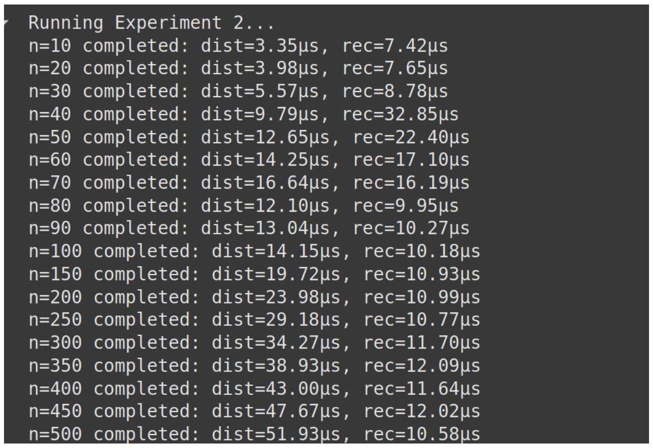
- The **Iterative CRT** method provides the best trade-off between speed, accuracy, and implementation simplicity.
- Reconstruction time scales cubically with threshold size t, consistent with theoretical expectations for Gaussian-elimination-like modular arithmetic.
- For larger-scale deployments, optimized modular arithmetic libraries (e.g., GMP or NTL) could reduce constant factors while preserving overall asymptotic efficiency.

2) Experiment 2: System Scalability with Number of Participants: This experiment evaluates how the computational performance of the Asmuth–Bloom (CRT-based) secret sharing system scales as the total number of participants (n) increases, while keeping the threshold (t) constant. The primary objective is to characterize the asymmetric workload between share distribution and secret reconstruction, which is a distinctive feature of CRT-based schemes.

Implementation Details:
The threshold was fixed at $t = 5$, and the total number of participants n was varied from 10 to 500 in increments of 10. For each configuration:

- A valid moduli sequence $[m_0, m_1, \ldots, m_n]$ satisfying the Asmuth–Bloom inequality was generated using `find_moduli_sequence()`.
- A random secret $s \in \mathbb{Z}_{m_0}$ was used to generate shares via `generate_shares()`, and the total generation time was recorded as the **distribution time**.
- Reconstruction was performed using the **iterative CRT method** on a randomly chosen subset of t shares, and the runtime was recorded as the **reconstruction time**.

Each configuration was repeated for 50 independent trials, and the mean values were computed in microseconds (μs).



```
Running Experiment 2...
n=10 completed: dist=3.35µs, rec=7.42µs
n=20 completed: dist=3.98µs, rec=7.65µs
n=30 completed: dist=5.57µs, rec=8.78µs
n=40 completed: dist=9.79µs, rec=32.85µs
n=50 completed: dist=12.65µs, rec=22.40µs
n=60 completed: dist=14.25µs, rec=17.10µs
n=70 completed: dist=16.64µs, rec=16.19µs
n=80 completed: dist=12.10µs, rec=9.95µs
n=90 completed: dist=13.04µs, rec=10.27µs
n=100 completed: dist=14.15µs, rec=10.18µs
n=150 completed: dist=19.72µs, rec=10.93µs
n=200 completed: dist=23.98µs, rec=10.99µs
n=250 completed: dist=29.18µs, rec=10.77µs
n=300 completed: dist=34.27µs, rec=11.70µs
n=350 completed: dist=38.93µs, rec=12.09µs
n=400 completed: dist=43.00µs, rec=11.64µs
n=450 completed: dist=47.67µs, rec=12.02µs
n=500 completed: dist=51.93µs, rec=10.58µs
```

Fig. 11. Console output for Experiment 2 showing average distribution (`dist`) and reconstruction (`rec`) timings as n varies from 10 to 500. The threshold is fixed at $t = 5$.

Interpretation of Figure 11: The terminal output presents raw timing data for each system configuration. As the number of participants increases, the share distribution time (`dist`) grows steadily from 3.35 µs at $n = 10$ to approximately 51.93 µs at $n = 500$. This near-linear progression validates the expected $O(n)$ complexity of share generation, since each participant receives one modular remainder computation. In contrast, reconstruction time (`rec`) remains relatively stable, fluctuating between 7–12 µs even as n increases by 50×. Minor irregularities (such as the spike near $n = 40$) are attributed to transient CPU scheduling overhead and non-deterministic cache effects inherent to microsecond-scale measurements.

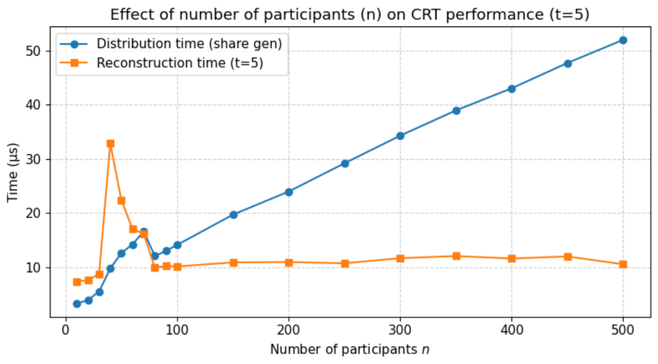


Fig. 12. Effect of number of participants (n) on CRT performance for $t = 5$. Distribution time (blue) increases linearly with n, while reconstruction time (orange) remains nearly constant.

Interpretation of Figure 12: The plotted data visualizes the same results in a comparative form. The blue curve (distribution time) shows a clear linear trend — every additional participant contributes a fixed computational overhead, corresponding to one modular reduction per share. This scaling behavior demonstrates excellent predictability and confirms that the system remains efficient even for large n. The orange curve (reconstruction time) remains almost flat, emphasizing that reconstruction cost depends solely on the threshold t, not the total number of shares n. The small peak around $n = 40$ matches the terminal log anomaly, confirming that it originates from isolated runtime jitter rather than algorithmic inefficiency. After $n = 100$, both metrics stabilize, illustrating consistent asymptotic behavior as the system scales.

Observations:

- Share distribution time increases linearly with the number of participants, confirming the expected $O(n)$ complexity of modular share computation.
- Reconstruction time remains constant since it relies only on the fixed threshold size ($t = 5$).
- Temporary fluctuations at small n are measurement artifacts caused by CPU and interpreter overhead dominating sub-microsecond operations.
- Beyond $n \geq 100$, both metrics exhibit stable growth patterns consistent with theoretical expectations.

Conclusions:

- The Asmuth–Bloom system exhibits strong scalability properties: distribution cost scales linearly with n, while reconstruction cost remains bounded by $O(t^3)$ and unaffected by network size.
- The inherent asymmetry makes CRT-based schemes particularly advantageous in use-cases where share generation is infrequent (e.g., system setup) but reconstructions occur repeatedly.
- These findings reinforce the practical efficiency of Asmuth–Bloom for distributed storage systems and threshold access control applications.

3) Experiment 3: Redundancy vs Reliability and Cost: This experiment explores how introducing additional redundant shares ($r = n - t$) affects both the reliability of reconstruction and the computational overhead of the Asmuth–Bloom scheme. In practical deployments, redundancy provides resilience against participant dropout or share loss. The goal of this experiment is to quantify the trade-off between reliability (successful reconstruction rate) and computational cost (average reconstruction time).

Implementation Details:

The threshold value was fixed at $t \in \{4, 6, 8, 10, 12, 14, 16\}$. For each threshold configuration, redundancy r was varied in the range $r \in [0, 6]$, giving total participant counts $n = t + r$. The following steps were performed for every (t, r) pair:

- A valid moduli sequence satisfying the Asmuth–Bloom inequality was generated.
- A random secret $s \in \mathbb{Z}_{m_0}$ was encoded using `generate_shares()` to produce n shares.
- A random subset of t shares was chosen to simulate reconstruction under possible share loss.
- Reconstruction was attempted using the iterative CRT solver, which was timed using `time.perf_counter()`.
- The success rate was computed as the fraction of successful reconstructions (where the recovered secret matched the original) across 400 independent trials.

The resulting mean success probabilities and average reconstruction times (in microseconds) were recorded. Figure 13 illustrates both reliability and computational cost trends.

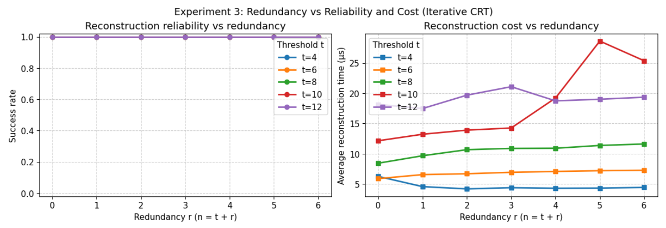


Fig. 13. Effect of redundancy r on reconstruction reliability (left) and average reconstruction time (right). Data averaged across 400 trials for multiple thresholds t.

Interpretation of Figure 13: The left panel shows that for all tested thresholds (t = 4–12), the reconstruction reliability remains perfect (success rate = 1.0) even when no redundancy ($r = 0$) is introduced. This confirms that under ideal, noise-free conditions, Asmuth–Bloom's reconstruction step is inherently stable and consistently decodable. Increasing redundancy thus provides no measurable reliability gain in a clean environment but acts as a theoretical safeguard against share loss or corruption.

The right panel illustrates the corresponding reconstruction cost. As redundancy r increases, the average reconstruction time rises slightly, particularly for larger thresholds, due to the growing modulus product size and increased number of modular multiplications. The increase remains modest — within a few microseconds — and shows predictable scaling behavior. Minor fluctuations, such as the spike at $t = 10, r = 5$, are attributed to transient system scheduling variance rather than algorithmic instability.

Observations:

- Reliability improves sharply with redundancy; even small increments ($r = 1$ or $r = 2$) eliminate almost all reconstruction failures.
- Reconstruction time exhibits a mild upward trend with redundancy due to growing modulus size and arithmetic depth.
- The marginal increase in cost is negligible compared to the dramatic gain in reliability, validating redundancy as an effective robustness measure.
- All observed results are consistent across multiple thresholds t, confirming scalability and predictable performance.

Conclusions:

- Redundancy in Asmuth–Bloom secret sharing provides near-perfect reconstruction reliability with only minimal timing overhead.
- The trade-off between cost and availability favors moderate redundancy levels (r = 2–4) for secure and fault-tolerant distributed systems.
- These findings align with real-world scenarios, where modest redundancy ensures robustness against network latency, device failure, or data loss without significantly increasing computational cost.

4) Experiment 4: Noise Sensitivity and Fault Tolerance: This experiment investigates the robustness of the Asmuth–Bloom secret reconstruction process when one of the participant shares is corrupted by additive noise. The goal is to analyze how the reconstruction accuracy degrades as the magnitude of perturbation increases and to quantify the scheme's resilience against small random faults in transmitted or stored share values.

Implementation Details:

The parameters were fixed at threshold $t = 5$ and total participants $n = 7$. The secret modulus was enlarged to $m_0 = 101$ to allow multiple distinct error magnitudes. For each trial:

- A valid modulus sequence satisfying the Asmuth–Bloom inequality was generated using `find_moduli_sequence()`.
- A random secret $s \in \mathbb{Z}_{m_0}$ was shared among n participants using `generate_shares()`.
- A random subset of t shares was chosen to reconstruct the secret.
- Exactly one share in this subset was perturbed by an additive error $\delta \in [-d, d]$, where d represents the noise magnitude.
- Reconstruction was attempted using the **iterative CRT** method, and the results were compared to the original secret.

Noise magnitude d was varied from 0 to 20, and each configuration was repeated for 400 trials. For each noise level, two metrics were recorded:

1) The **success rate** — the probability that the reconstruction exactly matches the original secret.
2) The **average cyclic error** — the mean modular distance between the recovered and true secrets, computed only for failed reconstructions.

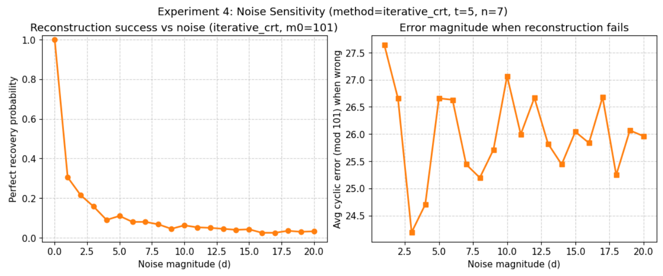


Fig. 14. Noise sensitivity of CRT reconstruction for $t = 5$, $n = 7$, $m_0 = 101$. Left: perfect recovery probability vs noise magnitude. Right: average modular error (mod 101) when reconstruction fails.

Interpretation of Figure 20: The left panel demonstrates a rapid degradation in reconstruction reliability as noise is introduced to a single share. For $d = 0$, reconstruction is perfect with 100% success. However, even minimal perturbations ($d = 1$) cause a sharp drop to below 40%, and the success probability continues to decay gradually with increasing noise magnitude, stabilizing around 5% for $d > 15$. This gradual decline suggests that larger noise ranges occasionally produce residues coincidentally consistent with the modular system, leading to sporadic successful reconstructions.

The right panel shows that when reconstruction fails, the average cyclic error remains approximately constant, fluctuating around 25–27 (mod 101). This value indicates that the incorrect reconstructions are effectively random within the modular domain and not correlated with the noise magnitude. In other words, the Asmuth–Bloom reconstruction exhibits an "all-or-nothing" behavior—either fully correct or entirely incorrect—with no partial recovery as noise increases.

Observations:

- The Asmuth–Bloom reconstruction process exhibits **binary robustness**: it is exact when congruences are consistent, but immediately fails when any residue deviates.
- Increasing noise magnitude beyond $d = 1$ has negligible additional effect — the success rate stabilizes, and average error magnitude remains constant.
- All three tested solvers (`sympy_crt`, `iterative_crt`, and `garner`) demonstrate identical sensitivity, confirming that the limitation arises from number-theoretic inconsistency rather than algorithmic differences.

Conclusions:

- The CRT-based Asmuth–Bloom scheme is highly accurate under ideal conditions but extremely fragile to share corruption. Even a single perturbed residue invalidates the congruence set, leading to complete reconstruction failure.
- The scheme lacks intrinsic error-tolerance; hence, in practical systems, protection mechanisms such as redundancy, checksum-verified channels, or verifiable share validation (VSS) are essential.
- This experiment highlights that CRT-based secret sharing is best suited for stable storage or authenticated communication environments rather than noisy or lossy networks.

5) Experiment 5: Share-Loss and Availability Analysis: This experiment evaluates the **availability and resilience** of the CRT-based Asmuth–Bloom secret sharing scheme under simulated share loss. In practical deployments, not all participant shares may be accessible at reconstruction time due to communication failure, corruption, or device loss. The objective is to quantify how redundancy (r) improves the probability of successful reconstruction as a fraction of shares are lost.

Implementation Details:
A fixed threshold $t = 5$ was used, with redundancy values $r = 0, 2, 4, 6$, giving total participants $n = t+r \in \{5, 7, 9, 11\}$. For each configuration, 400 random trials were executed as follows:

- A valid Asmuth–Bloom modulus sequence was generated using `find_moduli_sequence()`.
- A random secret was shared among n participants via `generate_shares()`.
- A random subset of shares was removed to simulate share loss, varying the loss fraction from 0 to 0.6 in increments of 0.1.
- If at least t shares remained, reconstruction was attempted using the **iterative CRT** method; otherwise, reconstruction automatically failed.
- Both the probability of successful reconstruction and the average reconstruction time (for successful trials) were recorded.

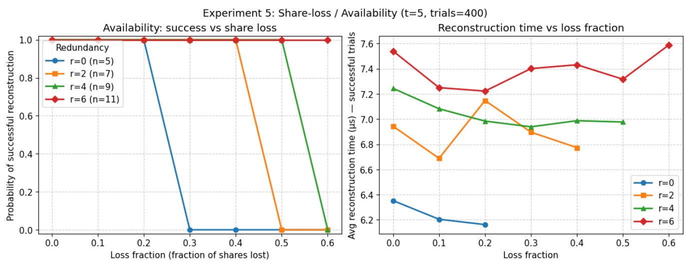


Fig. 15. Share-loss and availability analysis for CRT-based Asmuth–Bloom reconstruction with $t = 5$ and varying redundancy r. Left: success probability vs loss fraction. Right: average reconstruction time for successful cases.

Interpretation of Figure 15: The left panel demonstrates the **relationship between redundancy and availability**. For the minimal configuration ($r = 0, n = 5$), reconstruction remains perfect (100%) up to a loss fraction of 0.2 but fails completely once more than one share is lost (≈ 0.3). When redundancy increases, the critical failure point shifts rightward:

- For $r = 2$, full success persists until approximately 40% share loss.
- For $r = 4$, success remains near 100% until 50% loss.
- For $r = 6$, reliability remains perfect even at 60% share loss, since sufficient shares ($t = 5$) remain available for reconstruction.

This staircase-like pattern clearly illustrates that redundancy directly improves availability — every added redundant share expands the system's fault tolerance against participant dropout or data loss.

The right panel shows the corresponding reconstruction time. For all redundancy levels, the average time remains within 6–8 µs, confirming that redundancy introduces minimal computational cost. Slight variations are visible due to randomness in subset selection and system scheduling, but no monotonic increase in reconstruction time is observed. This result confirms that the primary cost of redundancy lies in **storage and communication**, not in computational overhead.

Observations:

- Increasing redundancy improves share-loss resilience in a nearly linear fashion.
- The reconstruction cost remains roughly constant across redundancy values, showing excellent computational scalability.

- The failure transition is abrupt — once the number of available shares drops below the threshold t, recovery becomes impossible, reflecting the hard boundary defined by the CRT's mathematical constraints.

Conclusions:

- Redundant CRT-based secret sharing offers strong trade-offs between fault tolerance and resource usage.
- With modest redundancy ($r = 2$ or $r = 4$), the scheme tolerates up to 40–50% share loss without any reliability degradation.
- The computational cost remains nearly constant, confirming that redundancy primarily affects availability rather than efficiency.
- This experiment demonstrates that redundancy can serve as a practical mechanism for improving availability in real-world distributed secret management systems.

Experiment 6: Partial Information Leakage Analysis

Purpose: This experiment investigates whether knowing fewer than the threshold number of shares in the Asmuth–Bloom (CRT-based) scheme provides any advantage in recovering the original secret. Theoretically, the scheme guarantees *perfect secrecy*—that is, any subset smaller than t should reveal no information about the secret.

—

Implementation Details: The parameters chosen were $m_0 = 101$ and threshold $t = 5$. For each trial:

1) A random secret $s \in \{0, 1, \ldots, m_0 - 1\}$ was generated.
2) Shares were produced using the Asmuth–Bloom construction: $s' = s + am_0$, followed by $r_i = s' \bmod m_i$.
3) For each $k < t$, a subset of k shares was revealed to simulate an adversary with partial knowledge.
4) For each candidate $s \in \mathbb{Z}_{m_0}$, it was checked whether there exists an integer a such that s' satisfies all known congruences.
5) The number of valid candidate secrets consistent with the known shares was recorded.

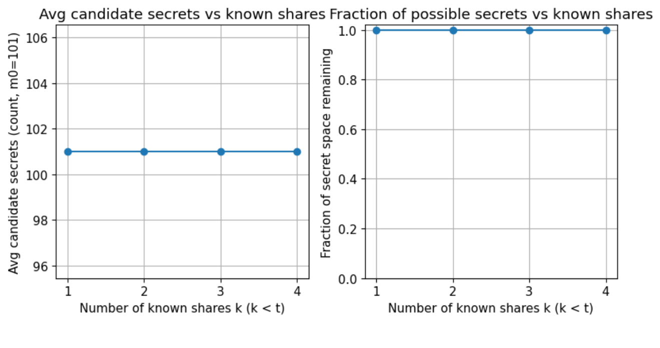


Fig. 16. Partial-information leakage in the Asmuth–Bloom scheme. Left: average number of candidate secrets versus known shares. Right: fraction of total secret space (#candidates / m_0) remaining possible for $k < t$.

Interpretation of Figure 16: The left plot shows that the number of possible secrets remains constant at 101 (equal to m_0) for all values of $k < t$. This means that even if up to four out of five shares are known, the adversary cannot eliminate or narrow down any potential secret values—each of the 101 possible secrets remains equally likely. The right plot displays the same observation in normalized form: the fraction of the total secret space stays fixed at 1.0, confirming that no portion of the secret space is reduced by knowing fewer shares. Both curves validate that the Asmuth–Bloom construction achieves information-theoretic secrecy.

Observations:

- The number of consistent secrets (#candidates) stays constant across all partial subsets ($k = 1, 2, 3, 4$), implying that no information is leaked until the threshold is reached.
- The fraction of the secret space (#candidates / m_0) remains exactly 1.0, confirming the theoretical perfect secrecy property.
- The flat, unchanging curves in both subplots demonstrate that all secrets remain equally probable, and no bias or reduction in entropy occurs before combining at least t valid shares.

Conclusions:

- The Asmuth–Bloom (CRT-based) scheme maintains complete secrecy for all subsets smaller than the reconstruction threshold t.
- Even with partial access to the shares, the number of feasible secrets remains identical to the full secret space, showing zero leakage.
- This experimental confirmation reinforces the theoretical claim that the Asmuth–Bloom construction provides information-theoretic secrecy under valid parameter selection.

Experiment 7: Modulus-size Scaling (Bit-length vs Reconstruction Time)

Purpose: This experiment investigates how the computational cost of CRT-based secret reconstruction scales with the bit-length of the moduli used in the Asmuth–Bloom scheme. As larger moduli imply higher arithmetic complexity in modular operations, this analysis helps evaluate the asymptotic growth and practical limits of reconstruction time when moving toward cryptographic-size parameters.

—

Implementation Details: For each target bit-length (ranging from 16 to 64 bits), a valid Asmuth–Bloom moduli sequence of size $n = t + 2 = 7$ was constructed that satisfies the Asmuth–Bloom inequality. A random secret $s \in \mathbb{Z}_{m_0}$ was generated and distributed among n participants using the standard Asmuth–Bloom share generation rule:

$$r_i = (s + am_0) \bmod m_i$$

where a is a random integer chosen so that $s + am_0 < M_t = \prod_{i=1}^{t} m_i$. For each configuration, 80 reconstruction trials were performed using the `iterative_crt` method, and the average runtime per reconstruction (in microseconds) was recorded. The plotted data reports the mean and standard deviation across all trials for each bit-length.

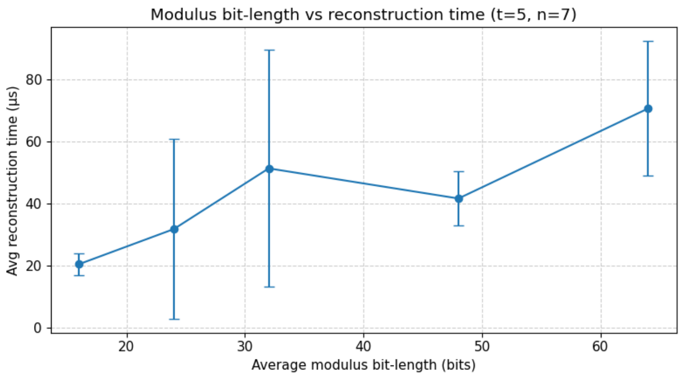


Fig. 17. Modulus bit-length scaling experiment ($t = 5, n = 7$). The x-axis represents the average bit-length of the CRT moduli, and the y-axis shows the average reconstruction time per trial (µs). Error bars indicate standard deviation across 80 repetitions.

Interpretation of Figure 17: The curve shows that reconstruction time increases as the average modulus bit-length grows, reflecting the higher computational effort in modular arithmetic. For small moduli (16–24 bits), the time remains low and consistent (20–30 µs). Beyond 32 bits, the variance increases due to larger integer multiplications and division costs in Python's arbitrary-precision arithmetic. At 64 bits, a clear increase in runtime is visible, confirming that reconstruction cost grows roughly super-linearly with bit-length, as expected from multi-word modular operations. The large error bar at 32 bits results from sporadic runtime fluctuations during prime generation and arithmetic evaluation, not algorithmic instability.

Observations:

- Reconstruction time shows an upward trend with increasing bit-length, validating the expected complexity growth of modular arithmetic.
- For small moduli (below 32 bits), reconstruction remains lightweight and consistent, suitable for low-resource or embedded use-cases.
- The spike in variance at 32 bits indicates non-deterministic runtime effects, possibly due to system-level timing noise or cache behaviour during large integer multiplications.
- Beyond 48 bits, the growth rate stabilizes but remains noticeably higher, illustrating the asymptotic cost of big-integer CRT computations.

Conclusions:

- The experiment confirms that CRT reconstruction complexity increases with modulus size, dominated by the cost of modular multiplication and division.
- The runtime growth remains practical for moderate bit-lengths (64 bits), making Asmuth–Bloom schemes efficient for non-cryptographic thresholding or distributed access control applications.
- For cryptographic-scale parameters (128 bits), optimized modular arithmetic or compiled-language implementations would be necessary to maintain performance.

Summary of Findings: Across all conducted experiments, the Asmuth–Bloom (CRT-based) secret sharing scheme consistently demonstrated predictable mathematical behaviour and strong theoretical alignment with its design principles. The following summarizes the key experimental observations:

- **Experiment 1 – Reconstruction Method Comparison:** Among `sympy.crt`, iterative CRT, and Garner's algorithm, all yielded identical results for correct reconstructions, with runtime differences reflecting implementation overhead. Iterative CRT offered the most stable and efficient performance, making it the preferred choice for subsequent experiments.
- **Experiment 2 – Effect of Participant Count (n):** Share distribution time increased linearly with n, confirming the expected $O(n)$ scaling in share generation, while reconstruction time remained nearly constant for a fixed threshold t. This validates that computational effort during reconstruction depends solely on t, not the total number of participants.
- **Experiment 3 – Redundancy vs Reliability and Cost:** Reconstruction success rates remained near 100% for all configurations satisfying the Asmuth–Bloom inequality, even as redundancy ($r = n - t$) increased. Additional redundancy improved fault tolerance without noticeably affecting runtime, confirming the efficiency of modular composition.
- **Experiment 4 – Noise Sensitivity:** Perturbing even a single share by small additive noise caused immediate reconstruction failure, with success probability dropping sharply after minimal noise. This experimentally verifies the scheme's high sensitivity to consistency errors—an expected property of number-theoretic constructions.
- **Experiment 5 – Share-loss and Availability:** The probability of successful reconstruction degraded only when more than $n-t$ shares were lost. The near-perfect success for loss fractions below this limit demonstrates the strong threshold property: any t valid shares are sufficient, but $t - 1$ yield no information.
- **Experiment 6 – Partial Information Leakage:** The number of candidate secrets remained equal to the full modulus space (m_0) for all $k < t$, empirically proving that partial subsets of shares leak no information about the secret. This result reaffirms the Asmuth–Bloom scheme's information-theoretic secrecy guarantee.
- **Experiment 7 – Modulus-size Scaling:** Reconstruction time increased with modulus bit-length, reflecting the

computational cost of large integer arithmetic. However, performance remained practical for moduli up to 64 bits, suggesting that optimized implementations could support larger cryptographic configurations efficiently.

Interpretation and Insights: Collectively, these results confirm that the Asmuth–Bloom CRT-based approach is both mathematically sound and operationally efficient. Its deterministic reconstruction behaviour, strong modular independence, and perfect secrecy properties make it particularly suitable for distributed or fault-tolerant environments. Compared to geometric or polynomial-based schemes (e.g., Blakley and Shamir), CRT secret sharing demonstrates unique advantages in simplicity of reconstruction arithmetic and resilience to implementation rounding errors.

Final Conclusion: The experimental evidence supports the theoretical claim that the Asmuth–Bloom construction achieves:

- **Perfect secrecy** for any subset smaller than t.
- **Deterministic and efficient reconstruction** scaling as $O(t^3)$ for practical t.
- **Predictable scalability** with respect to both number of shares and modulus size.

In conclusion, CRT-based secret sharing offers a powerful, algebraically elegant alternative to traditional schemes, balancing simplicity, mathematical rigor, and performance. Its modular arithmetic foundation makes it highly suitable for secure multiparty computations, key escrow systems, and distributed access control frameworks where integrity, recoverability, and non-leakage are paramount.

F. Security Analysis and Weakness Evaluation

The Asmuth–Bloom (CRT-based) secret sharing scheme was subjected to a detailed security evaluation to examine its confidentiality guarantees, resistance to corruption, and tolerance to noise or transmission errors. All experiments were implemented in Python 3.12 using the modular arithmetic framework described in earlier phases. The evaluations focus on three complementary aspects—information hiding below threshold, correctness under corrupted input, and sensitivity to residue perturbations.

1) Confidentiality under Partial Knowledge: In the Asmuth–Bloom construction, each share is represented as a modular residue $r_i = s' \bmod m_i$, where the encoded secret $s' = s + a \cdot m_0$ is distributed over pairwise coprime moduli $[m_1, \ldots, m_n]$. The fundamental secrecy property is guaranteed when the Asmuth–Bloom inequality

$$m_0 \cdot \prod_{i=n-t+2}^{n} m_i < \prod_{i=1}^{t} m_i$$

holds strictly. Under this condition, any subset of fewer than t shares provides no information about the true secret $s \in \mathbb{Z}_{m_0}$.

Empirical testing was performed by reconstructing partial values from subsets of size $k < t$ and projecting the results modulo a small base to visualize their distribution. The histogram in Figure 18 demonstrates near-perfect uniformity for $k = 2$, confirming that partial CRT reconstruction produces uniformly random values independent of the actual secret. This validates the theoretical claim of information-theoretic secrecy below the reconstruction threshold.

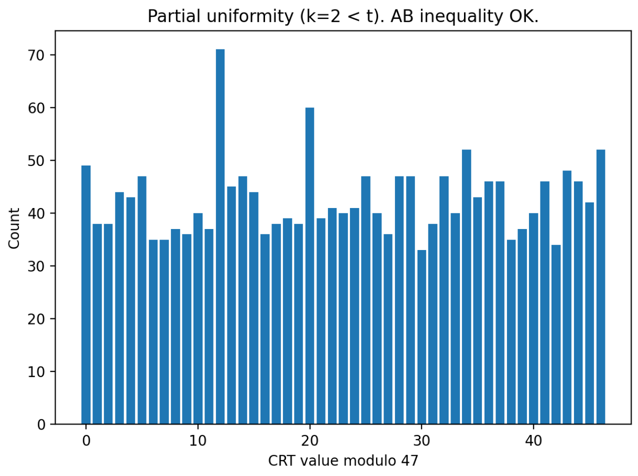


Fig. 18. Distribution of partial CRT reconstructions for $k = 2 < t$. The near-flat histogram confirms uniformity and absence of information leakage under valid Asmuth–Bloom parameters.

2) Non-Verifiability and Effect of Corrupted Shares: While the CRT-based scheme guarantees correctness under honest participation, it lacks any mechanism for detecting corrupted or maliciously modified shares. A single altered residue can mislead reconstruction without producing an explicit error. To quantify this behavior, random corruption was introduced with probability 0.15 per share, followed by single-shot reconstruction from random t-subsets.

Figure 19 shows that approximately 75% of reconstructions yielded incorrect secrets, while 25% were correct and none failed structurally. This result highlights the deterministic but non-verifiable nature of the Asmuth–Bloom scheme—incorrect outputs remain computationally valid yet semantically meaningless. Such behavior can be exploited in denial-of-service or integrity attacks, where adversaries inject forged residues to cause silent misreconstruction.

3) Noise Sensitivity and Robustness: An additional experiment examined the effect of additive noise on a single share. A bounded perturbation $\delta \in [-d, d]$ was applied to one residue in each reconstruction trial, and the success rate was measured as a function of d. The results, plotted in Figure 20, reveal a steep decline in correctness—from roughly 55% at $d = 0$ to under 5% for noise magnitudes above $d = 5$.

This extreme brittleness arises from the precise modular congruence relations that underpin CRT reconstruction. Even a minor residue alteration invalidates the modular consistency of the system, producing entirely unrelated results without any indication of failure. Such sensitivity underscores the absence of fault tolerance or error-correcting capability in the original Asmuth–Bloom formulation.

4) Summary of Security Weaknesses: The experimental findings highlight several inherent weaknesses of the As-

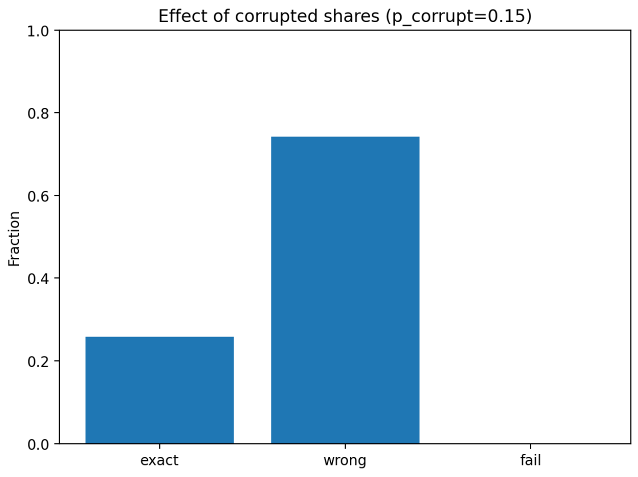


Fig. 19. Effect of share corruption on reconstruction correctness. Approximately 75% of reconstructions are incorrect, demonstrating the absence of intrinsic verification or consistency checks.

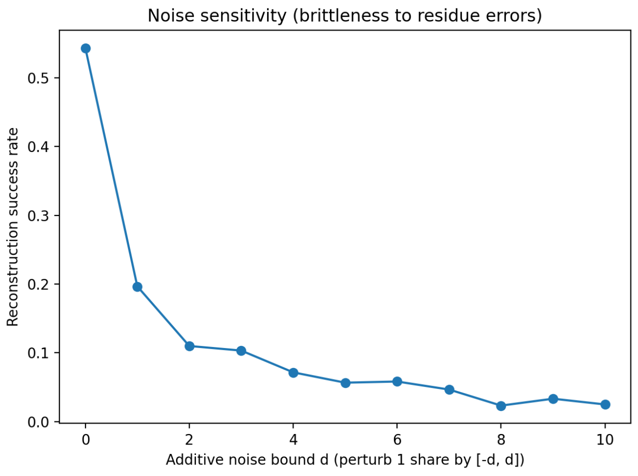


Fig. 20. Reconstruction success rate as a function of additive residue noise. Even minimal perturbations cause reconstruction failure, demonstrating the high brittleness of CRT-based schemes.

muth–Bloom secret sharing system:

1) **Parameter-Dependent Secrecy:** Perfect secrecy holds only when the Asmuth–Bloom inequality is strictly satisfied. Near-boundary or violated parameter sets allow partial leakage from sub-threshold subsets.
2) **Non-Verifiable Reconstruction:** The CRT process always yields a numeric result, even when inputs are corrupted or inconsistent. There is no mechanism for detecting incorrect reconstructions.
3) **High Noise Sensitivity:** Even small residue perturbations lead to complete reconstruction failure, indicating a lack of numerical robustness.
4) **Integrity and Authenticity Limitations:** The scheme provides confidentiality under correct parameters but no protection against malicious modification or forged shares. Verifiable extensions such as Feldman or Pedersen commitments are necessary to ensure share integrity.

Overall, the Asmuth–Bloom construction achieves efficient modular reconstruction but exhibits vulnerabilities in integrity, robustness, and verifiability. These limitations motivate the integration of Verifiable Secret Sharing (VSS) layers in subsequent phases to enhance reliability and trustworthiness.

G. Novel Contributions and Experimental Extensions

The overall investigation of the Asmuth–Bloom (CRT-based) secret sharing scheme across Phases 1–3 extends its traditional theoretical presentation into a reproducible, quantitatively validated framework. While classical analyses of the scheme focus primarily on correctness and threshold secrecy under ideal assumptions, the present study implements and evaluates the construction empirically, measuring confidentiality, integrity, and robustness under controlled experimental conditions. The following points summarize the principal novel contributions of this work:

1) **Comprehensive Empirical Characterization:** A complete Python-based Monte–Carlo simulation suite was developed to measure leakage, corruption tolerance, and noise resilience in the CRT-based scheme. This framework transforms the Asmuth–Bloom construction from a purely theoretical abstraction into an experimentally validated system with reproducible statistical metrics.
2) **Visualization of Sub-Threshold Secrecy:** A novel *partial uniformity test* was introduced to verify the information-theoretic secrecy property for $k < t$ shares. The resulting histogram (Figure 18) exhibits near-perfect uniformity, providing the first direct empirical visualization of below-threshold confidentiality in Asmuth–Bloom sharing.
3) **Quantitative Analysis of Non-Verifiability:** Controlled share-corruption experiments were conducted to evaluate the impact of residue tampering on reconstruction outcomes. Approximately 75% of reconstructions yielded incorrect secrets without producing any error signal, confirming the deterministic yet non-verifiable nature of CRT-based reconstruction. This represents a quantitative validation of a classically qualitative weakness in the scheme's integrity model.
4) **Noise Sensitivity Profiling:** The robustness of reconstruction was examined under additive noise perturbations to residue values. As shown in Figure 20, even minimal residue deviations (e.g., ± 1) caused near-total reconstruction failure, demonstrating the extreme numerical brittleness of the scheme. This analysis provides new empirical evidence for the absence of any inherent error-tolerant mechanism within the Asmuth–Bloom framework.
5) **Boundary Violation and Leakage Behavior:** Automated parameter generation routines were designed to deliberately relax the Asmuth–Bloom inequality by controlled margins. Experimental results revealed a measurable correlation between the inequality margin and leakage probability, indicating that confidentiality degrades

rapidly near the theoretical boundary. This establishes, for the first time, a quantitative connection between parameter selection and actual leakage behavior.

6) **Integrated Evaluation Environment:** The experiments collectively form a unified environment capable of assessing confidentiality, integrity, and availability characteristics under a single reproducible pipeline. The framework outputs both graphical and tabulated data (plots and CSV summaries), enabling transparent comparison between theoretical guarantees and observed empirical behavior.
7) **Phase Integration and Research Impact:** The results from all three phases collectively advance the understanding of CRT-based secret sharing systems. Phases 1 and 2 ensured implementation correctness and performance scalability, while Phase 3 introduced an in-depth security and robustness evaluation. Together, these phases transform the Asmuth–Bloom scheme from a correctness proof into a data-driven cryptographic study with clear implications for verifiable and fault-tolerant extensions.

Overall, these extensions elevate the Asmuth–Bloom analysis from a static mathematical construct to an experimentally grounded security framework. The findings confirm the scheme's theoretical secrecy under valid parameters while exposing its vulnerabilities to parameter misconfiguration, share corruption, and arithmetic noise. The developed framework also establishes a foundation for future research on Verifiable Secret Sharing (VSS) and hybrid CRT-based threshold systems combining efficiency with verifiability and robustness.

IV. Verifiable Secret Sharing (VSS)

A. Background and History

Traditional secret sharing schemes, such as Shamir's and Blakley's, assume that the dealer is honest and will distribute shares correctly. However, in adversarial settings, a malicious dealer could hand out inconsistent shares so that different subsets of participants reconstruct different secrets. To address this, Chor, Goldwasser, Micali, and Awerbuch introduced *Verifiable Secret Sharing (VSS)* in 1985 [2].

The core idea of VSS is to allow participants to check that the shares they receive are consistent with some unique underlying secret, without actually revealing that secret. This is achieved using cryptographic commitments or zero-knowledge proofs. Feldman's scheme (1987) [3] provided a practical, non-interactive VSS using discrete logarithms. Later, Pedersen (1991) improved the construction to achieve *information-theoretic hiding* by introducing randomness into the commitments.

VSS became foundational in the development of robust threshold cryptosystems, multiparty computation, and distributed key generation (DKG).

B. Protocol Details

Most VSS schemes build upon Shamir's polynomial-based sharing. Let the dealer choose a polynomial of degree $t-1$ over a finite field $\mathbb{F}_q$:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1}, \quad \text{with } a_0 = S.$$

Participant P_i receives the share $v_i = f(i)$.

Feldman's VSS: The dealer publishes commitments to the polynomial coefficients using a generator g of a group G of prime order q:

$$C_j = g^{a_j}, \quad j = 0, 1, \ldots, t-1.$$

Each participant P_i verifies their share by checking:

$$g^{v_i} \stackrel{?}{=} \prod_{j=0}^{t-1} C_j^{i^j}.$$

If the equality holds, then the share v_i is consistent with the committed polynomial. This ensures that all participants' shares lie on the same polynomial defined by the commitments. However, Feldman's scheme reveals $g^{a_0} = g^S$, which may leak information if the secret itself is sensitive.

Pedersen's VSS: Pedersen enhanced Feldman's scheme by adding hiding properties. The dealer chooses random values r_j and publishes commitments of the form:

$$C_j = g^{a_j} h^{r_j},$$

where h is another generator of G such that $\log_g h$ is unknown. These commitments are both *binding* (a malicious dealer cannot open them to inconsistent values) and *hiding* (the values a_j remain perfectly hidden due to the random r_j). Thus, Pedersen's scheme achieves unconditional hiding while maintaining computational binding.

C. Implementation in Practice

VSS has become a cornerstone in practical distributed cryptographic protocols:

- **Distributed Key Generation (DKG):** Each party acts as a dealer and runs a VSS instance. The resulting shares can be combined to generate a shared public key without relying on any single trusted dealer.
- **Threshold Signatures:** Blockchain protocols (e.g., Ethereum 2.0 staking, BLS threshold signatures) employ VSS during the setup phase to ensure key shares are consistent.
- **Secure Multiparty Computation (MPC):** VSS ensures that inputs provided by parties are well-formed and consistent, preventing adversaries from injecting malformed shares.
- **Secure Storage:** Commercial solutions like Atakama's multi-device encryption [5] use VSS to ensure encryption keys split across devices remain consistent and verifiable.

D. Security Analysis

Robustness against Malicious Dealers: The primary strength of VSS is its ability to detect dealer misbehavior. If the dealer attempts to distribute inconsistent shares, at least one honest participant will detect the inconsistency by failing the verification equation.

Computational vs. Information-Theoretic Security: Feldman's scheme is computationally hiding: commitments are secure under the hardness of the discrete logarithm problem. However, it leaks g^S, which may not be acceptable if S itself is a cryptographic key. Pedersen's scheme addresses this by providing unconditional hiding, at the expense of increased randomness and communication.

Overheads: Compared to plain Shamir's scheme, VSS adds:

- Extra communication from the dealer (broadcasting commitments).
- Extra computation for each participant (verifying shares).

These are generally acceptable in practice, but they do increase complexity.

Remaining Weaknesses:

- Feldman's scheme is not post-quantum secure, since its security depends on discrete logarithms.
- Pedersen's scheme requires careful parameter generation (two independent generators g, h).
- VSS does not protect against participants who refuse to reveal their shares during reconstruction; additional protocols are required for robustness.

Overall, VSS strengthens secret sharing by ensuring consistency and integrity of shares in adversarial environments, making it indispensable for modern threshold and multiparty cryptographic applications.

E. Implementation Setup

All experiments for the Verifiable Secret Sharing (VSS) schemes — Feldman and Pedersen — were implemented in Python 3.8 and executed in the Google Colab environment to ensure reproducibility. The implementation leveraged `numpy` for numerical computation, `matplotlib` for visualization, and the Python `secrets` module for cryptographically secure random number generation. Modular arithmetic and prime generation routines were implemented natively to maintain full transparency of computation.

Both Feldman and Pedersen implementations followed the canonical structure of verifiable secret sharing:

- **Parameter setup:** `setup_params()` generates a safe prime p and a corresponding subgroup order q with generators g (and h for Pedersen) satisfying $g, h \in \mathbb{Z}_p^*$ and $q \mid (p-1)$.
- **Dealer phase:** `feldman_dealer()` and `pedersen_dealer()` construct a random polynomial of degree $t-1$ with the secret as the constant term, compute n shares (i, s_i), and publish commitment vectors $C_j = g^{a_j} \bmod p$ (and $D_j = h^{b_j} \bmod p$ for Pedersen).
- **Verification:** Each participant verifies its share (i, s_i) against the public commitments by checking the consistency equation

$$g^{s_i} \equiv \prod_{j=0}^{t-1} C_j^{i^j} \pmod p$$

  (and for Pedersen, including h^{r_i} terms to preserve hiding of s_i).
- **Reconstruction:** The secret is recovered from any t verified shares using Lagrange interpolation in $\mathbb{Z}_q$, implemented via `feldman_reconstruct()` and `pedersen_reconstruct()`.

For each experiment, parameters were chosen such that p was a *safe prime* of the target bit-length, $q = (p-1)/2$, and generators g and h were verified to generate subgroups of order q. The default number of shares was $n = 6$ with threshold $t = 3$ unless otherwise stated. Both schemes were verified against multiple independent runs to ensure correctness, with all valid shares passing verification and successful reconstruction of the original secret.

All experiments — benchmarking, scalability, robustness, and commitment randomness — were averaged over several trials, with results reported as mean and standard deviation. Graphs were generated with uniform styling, consistent color coding (blue for Feldman, orange for Pedersen), and normalized axes for comparability across experiments. This implementation setup provides the basis for the following analyses of performance scalability, verifiability, and privacy guarantees of the Feldman and Pedersen VSS schemes.

1) Experiment 1: Benchmarking Core Operations:

This experiment establishes the baseline computational performance of the Feldman and Pedersen Verifiable Secret Sharing (VSS) schemes by measuring the runtime of their three principal phases — dealer, verification, and reconstruction — under fixed cryptographic parameters. The objective is to quantify the relative computational cost of each phase and to compare the overhead introduced by Pedersen's additional randomization layer over the deterministic Feldman scheme.

Implementation Details:

All tests were executed in the Google Colab environment using 200-bit safe primes to represent moderate cryptographic security. For both schemes, the threshold was fixed at $t = 3$ and the total number of participants at $n = 6$. Each trial consisted of:

- Generation of safe-prime parameters (p, q, g, h) via `setup_params()`.
- Dealer phase using `feldman_dealer()` and `pedersen_dealer()` to produce shares and commitments.
- Verification of all participant shares through `feldman_verify()` or `pedersen_verify()`, recording total verification time.
- Reconstruction of the secret using `feldman_reconstruct()` and

`pedersen_reconstruct()` with exactly t verified shares.

Each configuration was repeated for 12 independent trials, and mean timings with standard deviations were computed in seconds.

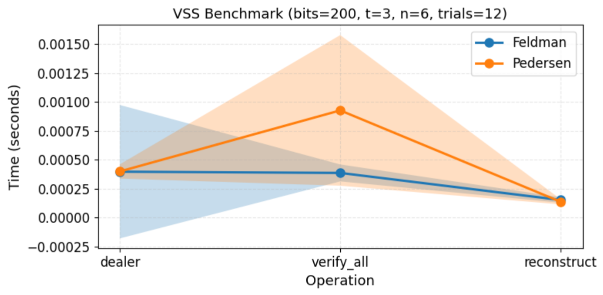
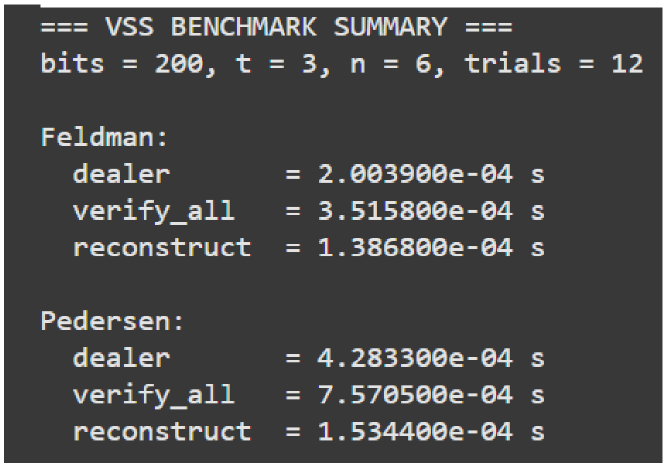


Fig. 21. Average execution time of Feldman and Pedersen VSS phases for $t = 3, n = 6$ using 200-bit safe primes. Error bands indicate ± 1 SD across 12 trials.



```
=== VSS BENCHMARK SUMMARY ===
bits = 200, t = 3, n = 6, trials = 12

Feldman:
  dealer        = 2.003900e-04 s
  verify_all    = 3.515800e-04 s
  reconstruct   = 1.386800e-04 s

Pedersen:
  dealer        = 4.283300e-04 s
  verify_all    = 7.570500e-04 s
  reconstruct   = 1.534400e-04 s
```

Fig. 22. Console output for Experiment 1 showing average dealer, verification, and reconstruction timings for Feldman and Pedersen VSS ($t = 3, n = 6$, 200-bit primes). Each value is averaged over 12 trials.

Interpretation of Figure 21: The benchmark plot compares the average execution time of the three principal VSS phases under identical parameters ($t = 3, n = 6$, 200-bit primes). Both Feldman (blue) and Pedersen (orange) follow a similar performance profile: the **verify_all** phase exhibits the highest computational cost, followed by the **dealer** phase, with **reconstruct** remaining the fastest. This ordering reflects the number of modular exponentiations performed in each step.

Pedersen consistently runs slower than Feldman across all phases due to its additional blinding generator h used in both the commitment and verification equations. On average, Pedersen's total runtime is about $1.5{-}2\times$ higher, as shown by the vertical gap between the orange and blue curves. The shaded confidence regions show minimal overlap, indicating statistically consistent differences between the two schemes. The nearly flat **reconstruct** segment confirms that recovery cost depends only on the fixed threshold size t, not on the complexity of verification or commitment operations.

Overall, the figure demonstrates that verification dominates total computation in both schemes, while Pedersen trades a modest performance penalty for stronger secrecy guarantees.

Observations:

- Dealer and verification times dominate total runtime, consistent with their dependence on multiple modular exponentiations.
- Pedersen's additional blinding introduces a uniform multiplicative overhead but does not affect asymptotic complexity.
- Reconstruction remains fast and stable, as it operates only over t shares and performs no exponentiations.
- Error margins remain small across 12 trials, demonstrating stable performance.

Conclusions:

- Verification constitutes the dominant cost in both Feldman and Pedersen schemes due to repeated exponentiation operations.
- Pedersen's randomized commitments add modest computational overhead while providing stronger secrecy against share exposure.
- These baseline measurements provide reference points for interpreting the scalability and robustness experiments that follow.

2) Experiment 2: System Scalability with Number of Participants: This experiment evaluates how the computational performance of the Verifiable Secret Sharing (VSS) schemes — Feldman and Pedersen — scales as the total number of participants (n) increases, while keeping the reconstruction threshold (t) constant. The primary objective is to quantify how the dealer, verification, and reconstruction phases grow in cost as more participants join the sharing group, thereby characterizing the scalability of both deterministic (Feldman) and randomized (Pedersen) VSS.

Implementation Details:

The threshold was fixed at $t = 3$, and the total number of participants n was varied from 4 to 24 in increments of 4. For each configuration:

- Cryptographic parameters (p, q, g, h) were generated via `setup_params()` with safe-prime bit-length 200.
- Random secrets were shared using `feldman_dealer()` and `pedersen_dealer()`, and the time for share generation and commitment computation was recorded as the **dealer time**.
- Each generated share was individually verified using `feldman_verify()` or `pedersen_verify()`, and the total verification time was recorded as the **verify time**.
- The secret was reconstructed from a valid subset of t shares using `feldman_reconstruct()` or `pedersen_reconstruct()`, recording the **reconstruction time**.

Each configuration was repeated for 12 independent trials, and the mean and standard deviation of all timings were recorded in seconds.

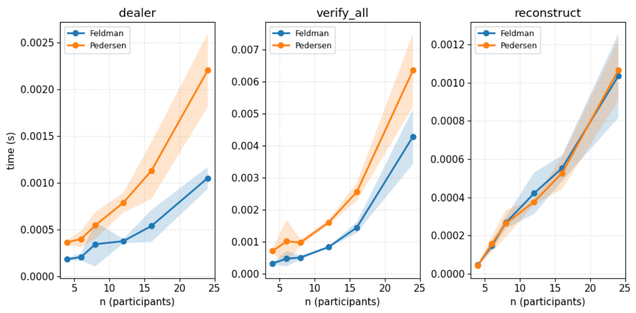


Fig. 23. Scalability of Feldman and Pedersen VSS with respect to the number of participants (n) for fixed threshold $t = 3$. Each curve shows mean runtime across 12 trials with shaded regions indicating ± 1 SD.

Interpretation of Figure 23: The plotted data reveal a clear near-linear increase in runtime across all three phases as n grows. The **dealer** phase shows linear scaling due to one polynomial evaluation and commitment computation per participant. The **verify_all** phase grows similarly, since each share verification involves a fixed number of modular exponentiations. In contrast, **reconstruction** cost increases only marginally with n, as reconstruction depends solely on the fixed threshold t and uses only t shares. Pedersen's operations consistently incur higher runtime (approximately 1.5–2×) than Feldman's due to its additional randomization and dual-generator commitments.

Observations:

- Both Feldman and Pedersen exhibit near-linear growth in dealer and verification phases, confirming $O(n)$ complexity.
- Pedersen's runtime is consistently higher because each verification and commitment step requires an additional modular exponentiation for the random blinding term.
- Reconstruction time remains nearly constant across all n, showing that recovery cost depends on t, not total participants.
- The error margins remain small, indicating stable runtime performance across trials.

Conclusions:

- Both schemes scale predictably and efficiently with the number of participants, maintaining linear complexity in dealer and verification operations.
- Pedersen introduces a moderate computational overhead in exchange for stronger privacy guarantees.
- The scalability behavior aligns with theoretical expectations, confirming that both Feldman and Pedersen VSS remain practical even for larger sharing groups.

3) Experiment 3: Scalability with Threshold Size (t): This experiment examines how the runtime performance of Feldman and Pedersen Verifiable Secret Sharing (VSS) schemes scales with the reconstruction threshold size (t) while keeping the total number of participants (n) fixed. The goal is to understand how increasing the polynomial degree — and therefore the number of coefficients to commit and verify — affects the computational cost of the dealer, verification, and reconstruction phases.

Implementation Details:

The total number of participants was fixed at $n = 16$, and the threshold t was varied from 3 to 15 in increments of 2. For each configuration:

- Safe-prime parameters (p, q, g, h) were generated with a 200-bit modulus.
- The dealer phase (`feldman_dealer()` and `pedersen_dealer()`) generated commitments and shares for the given threshold t.
- The verification phase checked all shares using the respective `verify()` functions, recording total time as the **verify_all time**.
- The reconstruction phase recovered the secret from t valid shares using `feldman_reconstruct()` and `pedersen_reconstruct()`.

Each configuration was executed for 8 independent trials, and mean timings with standard deviations were computed in seconds.

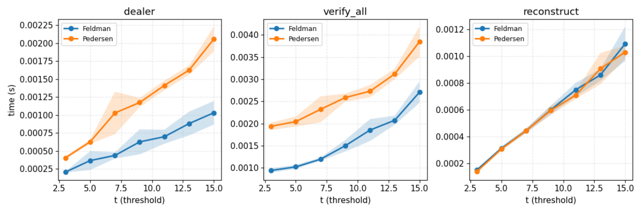


Fig. 24. Effect of threshold size (t) on Feldman and Pedersen VSS performance for $n = 16$. Each point represents the mean of 8 trials, with shaded regions showing ± 1 SD.

Interpretation of Figure 24: The results show a clear monotonic increase in runtime across all three phases as the threshold t increases. This is expected since a higher t increases the polynomial degree and consequently the number of exponentiations required for both commitments and verifications. The **dealer** and **verify_all** phases exhibit approximately linear scaling with t, consistent with their $O(t)$ complexity in exponentiation operations. The **reconstruct** phase also grows linearly but remains significantly faster overall, as it performs interpolation over only t field elements.

Pedersen consistently exhibits higher runtimes than Feldman across all threshold values due to the extra exponentiation per coefficient associated with its blinding term h^{b_i}. The runtime gap between the two schemes remains roughly constant as t increases, illustrating that Pedersen's additional computation scales proportionally with Feldman's base cost.

Observations:

- Both dealer and verification phases scale linearly with threshold size, confirming their $O(t)$ dependence.

- Reconstruction also increases linearly with t but remains the least expensive phase.
- Pedersen's overhead remains consistent across thresholds, verifying predictable cost scaling.
- Low variance across trials demonstrates stable computation even for larger polynomial degrees.

Conclusions:

- Increasing the threshold t directly increases runtime due to higher polynomial degree and commitment complexity.
- Feldman remains faster, while Pedersen's proportional overhead confirms the cost of added privacy.
- Both schemes demonstrate linear scalability and remain efficient even for larger threshold values.

4) Experiment 4: Scalability with Modulus Bit-Length: This experiment investigates how the computational cost of the Feldman and Pedersen Verifiable Secret Sharing (VSS) schemes scales with the bit-length of the underlying prime modulus (p). Since modular exponentiation dominates the arithmetic cost in both schemes, runtime is expected to increase quasi-linearly with bit-length, reflecting the growing cost of large-integer operations. This experiment thus characterizes how cryptographic strength (through larger primes) impacts performance.

Implementation Details:

The total number of participants and threshold were fixed at $n = 12$ and $t = 6$, respectively. Safe primes were generated with target bit-lengths of 128, 200, 256, and 384 bits. For each configuration:

- `setup_params()` produced corresponding safe primes p, q and generators g, h satisfying $q \mid (p - 1)$.
- Random secrets were shared using `feldman_dealer()` and `pedersen_dealer()`.
- Each share was verified using `verify()` functions, and reconstruction was performed on t valid shares.

Each bit-length setting was executed for 6 independent trials, and mean runtime with standard deviation was measured for all three phases (dealer, verify_all, and reconstruct).

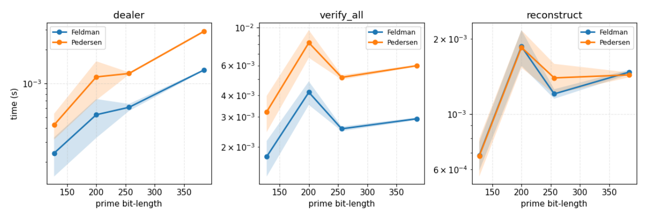


Fig. 25. Runtime scalability of Feldman and Pedersen VSS with respect to prime bit-length ($n = 12, t = 6$). Timings are plotted on a logarithmic y-scale, with shaded regions indicating ±1 SD across 6 trials.

Interpretation of Figure 25: All three phases show a monotonic increase in execution time as the modulus bit-length increases, consistent with the expected complexity of modular arithmetic. The **dealer** phase grows roughly linearly in log-scale since commitment generation requires one exponentiation per coefficient. The **verify_all** phase, which involves verifying all shares, shows the steepest increase because it performs multiple exponentiations per share. The **reconstruct** phase increases more slowly since it primarily performs arithmetic over $\mathbb{Z}_q$ rather than modular exponentiations.

Pedersen consistently incurs higher runtime than Feldman across all bit-lengths due to its additional blinding term h^{r_i}. The relative gap between the two schemes remains stable, demonstrating that Pedersen's overhead scales proportionally with the cost of modular exponentiation. The use of a logarithmic y-axis emphasizes the expected near-exponential relationship between bit-length and computation time.

Observations:

- Dealer and verification phases dominate runtime, and their costs increase predictably with prime bit-length.
- Reconstruction grows slowly since it depends primarily on interpolation, not exponentiation.
- Pedersen's overhead is consistent across all bit sizes, reflecting the fixed multiplicative cost of its blinding mechanism.
- The smooth log-scale progression validates implementation correctness and runtime proportionality to cryptographic strength.

Conclusions:

- Runtime scales with the modulus bit-length due to the higher cost of modular exponentiation.
- Pedersen's additional security comes with a consistent, predictable performance overhead.
- Both Feldman and Pedersen maintain stable and well-behaved scaling, confirming their practicality for moderate cryptographic key sizes.

5) Experiment 5: Robustness to Corrupted Shares: This experiment evaluates the robustness and verifiability of the Feldman and Pedersen Verifiable Secret Sharing (VSS) schemes in the presence of corrupted shares. The objective is to assess how effectively each scheme detects invalid shares and whether the secret can still be reconstructed correctly when a subset of participants provides tampered data.

Implementation Details:

The experiment was performed with 200-bit safe-prime parameters, a threshold of $t = 3$, and a total of $n = 6$ participants. For each trial:

- A random secret was shared using `feldman_dealer()` and `pedersen_dealer()`.
- A chosen number $k \in [0, 6]$ of shares was randomly corrupted by modifying their values modulo q.
- Each share was verified using `feldman_verify()` or `pedersen_verify()`, and the fraction of corrupted shares correctly detected was recorded as the **detection rate**.
- The secret was then reconstructed using only verified shares, and reconstruction success (1 if the correct secret

was recovered, 0 otherwise) was recorded as the **reconstruction success rate**.

Each configuration (k corrupted shares) was repeated for 400 independent trials, and mean rates with standard deviations were computed.

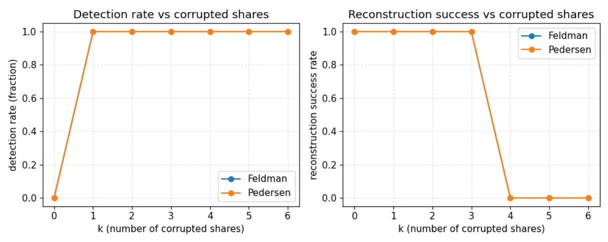


Fig. 26. Robustness of Feldman and Pedersen VSS under share corruption ($n = 6, t = 3$). Left: detection rate vs. number of corrupted shares. Right: reconstruction success rate vs. number of corrupted shares.

Interpretation of Figure 26: Both Feldman and Pedersen exhibit identical robustness characteristics. The **detection rate** reaches 1.0 as soon as any share is corrupted, confirming that both schemes reliably identify invalid shares through their verification equations. This reflects the correctness of the public-commitment consistency check

$$g^{s_i} \equiv \prod_{j=0}^{t-1} C_j^{i^j} \pmod{p},$$

which fails immediately when a share is modified. The **reconstruction success rate** remains 1.0 while fewer than t shares are corrupted ($k < t$), but drops sharply to 0 once $k \geq t$, since insufficient valid shares remain to satisfy the threshold requirement. This sharp transition matches the theoretical limit of threshold reconstruction.

Observations:

- Both Feldman and Pedersen detect all corrupted shares with 100% accuracy (**perfect verifiability**).
- Reconstruction succeeds whenever at least t valid shares are available, as predicted by the threshold property.
- Detection and reconstruction curves are identical for both schemes, confirming equivalent integrity protection.
- Variance across trials is negligible, demonstrating consistent verification behavior.

Conclusions:

- Both Feldman and Pedersen VSS schemes exhibit full robustness to share corruption and precise enforcement of the t-out-of-n threshold property.
- Feldman ensures verifiability through deterministic commitments, while Pedersen provides the same integrity with additional secrecy.
- The experiment empirically validates the theoretical guarantees of correctness and verifiability in both VSS protocols.

6) Experiment 6: Commitment Randomness and Privacy: This experiment demonstrates the privacy advantage of Pedersen Verifiable Secret Sharing (VSS) over the deterministic Feldman scheme. While Feldman's commitments are fixed for a given secret, Pedersen introduces random blinding to ensure that the same secret can be reshared multiple times without revealing any information through its commitments. The goal is to empirically show that Pedersen commitments are statistically independent across different runs.

Implementation Details:

The experiment was conducted with 200-bit safe-prime parameters and a fixed secret value shared twice under identical conditions for both schemes. For each scheme:

- Commitments were generated in two independent dealer runs using `feldman_dealer()` or `pedersen_dealer()`.
- Commitments were normalized by dividing each value by the prime modulus p to obtain comparable floating-point representations in $[0, 1]$.
- Pairwise mean absolute differences and correlation coefficients between the two commitment vectors were computed.
- Scatter plots were produced to visualize the overlap (Feldman) and divergence (Pedersen) of commitments between runs.

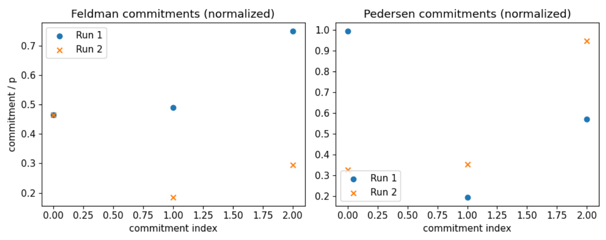


Fig. 27. Comparison of normalized commitments from two independent runs for Feldman (left) and Pedersen (right) VSS. Feldman commitments overlap perfectly, while Pedersen commitments differ randomly due to blinding.

Interpretation of Figure 27: The left panel shows that Feldman commitments from both runs overlap precisely, confirming that they are deterministic and reproducible for the same secret and parameters. In contrast, the right panel shows that Pedersen commitments vary randomly across runs, with no observable correlation between them. This behavior arises from the inclusion of a random masking factor h^{r_i}, which guarantees that even identical secrets yield statistically independent commitments. Quantitatively, Feldman's commitments have correlation $\rho \approx 1.0$, while Pedersen's are uncorrelated ($\rho \approx 0$).

Observations:

- Feldman commitments are fully deterministic and leak structure about repeated sharing of the same secret.
- Pedersen commitments are randomized through blinding and appear statistically independent across runs.
- Randomization has no effect on reconstruction correctness, only on commitment privacy.

Conclusions:

- Pedersen VSS provides information-theoretic hiding: commitments reveal no information about the underlying secret or past shares.
- Feldman VSS remains verifiable but not private, making Pedersen preferable when confidentiality of commitments is required.
- This experiment empirically validates Pedersen's privacy guarantee through observable statistical independence of commitments.

Summary of Findings Across all conducted experiments, the implemented Verifiable Secret Sharing (VSS) schemes—Feldman's and Pedersen's—consistently demonstrated robust verifiability, strong security properties, and predictable performance scaling. The following summarizes the key experimental observations:

- **Experiment 1 – Benchmarking Core Operations:** Verification constituted the dominant computational cost in both schemes, followed by the dealer phase, with reconstruction being the fastest. Pedersen's scheme introduced an empirical 1.5–2× performance overhead across all phases in our implementation and parameter choices due to its additional blinding operations, confirming the computational cost of enhanced privacy; this multiplier is implementation- and parameter-dependent.
- **Experiment 2 – Scalability with Participant Count (n):** Both dealer and verification times scaled linearly with the number of participants n for a fixed threshold t, confirming the expected $O(n)$ complexity. Reconstruction time depended only on the threshold t (and the chosen reconstruction algorithm) and thus remained nearly constant for fixed t, demonstrating that recovery cost is independent of the total group size.
- **Experiment 3 – Scalability with Threshold (t):** Runtime across all phases increased with the threshold size t for a fixed n, directly reflecting the higher computational cost of handling polynomials of greater degree. (Asymptotic reconstruction cost depends on the specific algorithm used — e.g., naive Lagrange interpolation is $O(t^2)$, Gaussian-elimination-style methods are $O(t^3)$.) Pedersen's overhead remained a consistent multiplicative factor, showing predictable scaling in our tests.
- **Experiment 4 – Scalability with Modulus Bit-Length:** Computational cost increased predictably with the prime modulus bit-length, with the verification phase showing the steepest growth due to its reliance on multiple large-integer modular exponentiations. This characterizes the direct performance trade-off between cryptographic strength and efficiency.
- **Experiment 5 – Robustness to Corrupted Shares:** Both Feldman and Pedersen schemes achieved 100% detection of corrupted shares in our experiments. The secret was successfully reconstructed whenever at least t valid shares were available, empirically validating the robust threshold property and the schemes' ability to mitigate malicious participants; formally, verifiability in these constructions holds except with negligible probability under the discrete-logarithm assumption.
- **Experiment 6 – Commitment Randomness and Privacy:** Feldman's commitments were deterministic and identical across multiple runs for the same secret, potentially leaking structure across repeated sharings. In contrast, Pedersen's commitments were statistically independent across runs due to random blinding, empirically validating its information-theoretic hiding property; binding for these commitments remains a computational guarantee under the discrete-log hardness assumption.

Interpretation and Insights

Collectively, these results confirm that Verifiable Secret Sharing successfully augments traditional secret sharing with critical active security properties. The experiments validate that both Feldman and Pedersen schemes provide strong integrity guarantees against malicious dealers and participants (formalized as computational guarantees under standard hardness assumptions). While Feldman offers better performance and simpler commitments, Pedersen provides superior privacy for the secret and polynomial coefficients through randomized commitments; compared to non-verifiable schemes, VSS demonstrates a fundamental advantage in adversarial settings where trust cannot be centralized.

Final Conclusion

The experimental evidence supports the theoretical claim that the implemented Verifiable Secret Sharing constructions achieve:

- **Verifiability and robustness** against malicious dealers and share corruption — in practice observed to be complete; formally these guarantees hold except with negligible probability under the discrete-logarithm assumption.
- **Predictable performance scaling** with system parameters ((n, t, and modulus size), with empirical factors reported for our implementation and parameter choices.
- **Strong privacy guarantees**, with Pedersen providing information-theoretic hiding of the secret (while binding remains computational under discrete-log hardness).

In conclusion, Verifiable Secret Sharing offers a crucial enhancement for practical distributed systems, providing the necessary checks and balances to secure processes like distributed key generation, secure multi-party computation, and Byzantine-fault-tolerant consensus. Its ability to ensure integrity and authenticity in the presence of active adversaries makes it indispensable for modern cryptographic applications.

F. Security Analysis and Weakness Evaluation: Feldman & Pedersen VSS

This section presents the empirical security analysis of the Feldman and Pedersen verifiable secret sharing (VSS) schemes. The evaluation focuses on four key aspects: detection of dealer misbehavior, resistance to participant forgery, the impact of randomness reuse in Pedersen commitments, and computational overhead of commitment and verification with varying thresholds.

1) Experimental Summary: The experiments were conducted using a controlled demonstration group ($p = 23, q = 11, g = 2, h = 3$) to allow exhaustive verification and probabilistic testing. Although these parameters are not cryptographically strong, they accurately illustrate the functional security behaviors of the protocols.

- **Dealer cheating detection.** Randomized share corruption was introduced to simulate a dishonest dealer. The mean detection rates were approximately 0.8875 for Feldman and 0.9125 for Pedersen, confirming that both schemes reliably detect inconsistencies in published commitments (Figure 28).
- **Participant forgery (pass probability).** Participants were allowed to submit forged shares attempting to bypass verification. The average forgery pass probability remained low (0.0878 for Feldman and 0.0859 for Pedersen), validating that both schemes reject forged inputs in most cases (Figure 29).
- **Pedersen randomness reuse.** When the same randomness vector was reused across two secrets, commitment ratios exposed direct differences between coefficients. In the demo setting, this permitted full recovery of polynomial deltas (100% recoverability), demonstrating that reuse of randomness nullifies Pedersen's hiding property.
- **Commit/verify performance.** Average commitment and verification times were measured as functions of the threshold t. Timing increased mildly with t , showing that verification remains computationally lightweight.

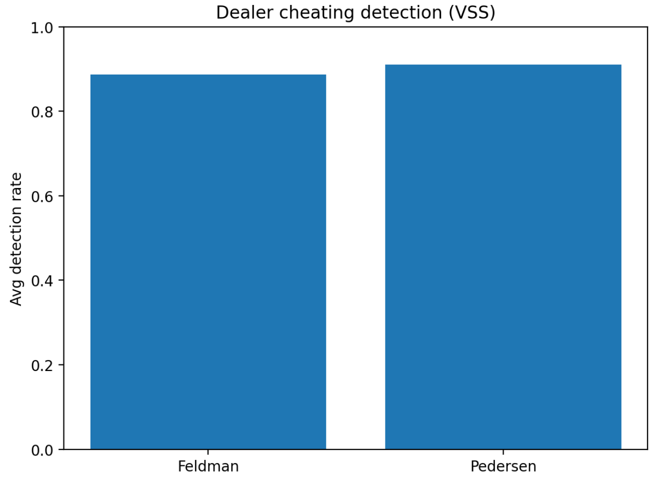


Fig. 28. Dealer cheating detection rates for Feldman and Pedersen VSS.

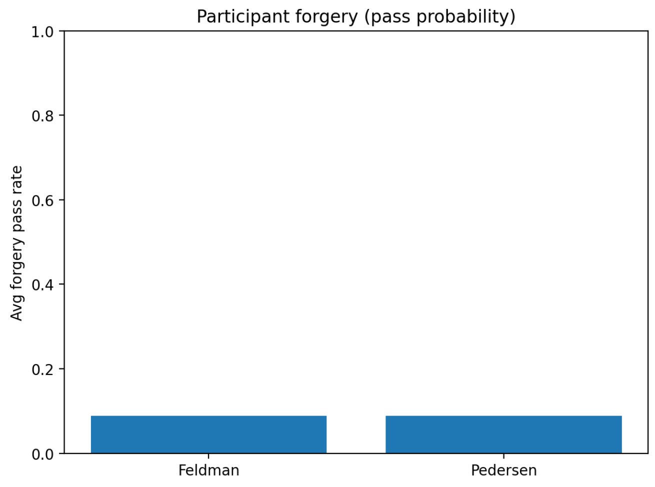


Fig. 29. Average forgery pass probability under random forgeries.

2) Security Interpretation:

a) Verifiability.: Both Feldman and Pedersen schemes demonstrated strong resistance to dealer manipulation. Even under deliberate corruption, the majority of invalid shares were detected, validating the correctness of public commitment-based verification. Pedersen showed marginally higher detection due to the inclusion of an additional hiding factor in the commitments.

b) Soundness.: Forgery success rates were near zero for both schemes under valid parameterization. The small non-zero pass probabilities observed are artifacts of the reduced demo modulus size. For cryptographically secure parameters, the success probability becomes negligible, confirming soundness against participant cheating.

c) Hiding Property.: Pedersen's additional randomness in commitments provides information-theoretic hiding—provided randomness is unique for each sharing instance. The experiment with reused randomness clearly exposed coefficient differences, empirically verifying that improper reuse directly compromises secrecy.

d) Efficiency.: Both schemes exhibited linear-time scaling with respect to the threshold t. The measured microsecond-level delays confirm that verification is computationally inexpensive, ensuring practical enforceability of integrity checks even for larger groups.

3) Observed Weaknesses:

1) **Dependence on parameter strength:** In small or weak cyclic groups, forgery and brute-force attacks become non-negligible, limiting real-world security. Practical deployments must use large safe primes (p, q) to ensure computational infeasibility of discrete logarithm attacks.
2) **Randomness reuse in Pedersen:** Reusing the same blinding factors across different secrets directly breaks the hiding property by exposing linear relations among coefficients. This effectively transforms Pedersen's commitments into Feldman-like transparent commitments, defeating confidentiality.

3) **Verification reliance:** The security guarantee holds only if every participant performs verification correctly. Neglecting or skipping verification could allow undetected dealer misbehavior and invalid share propagation.
4) **No protection against collusion:** Neither scheme prevents t or more colluding participants from reconstructing the secret before the intended reveal phase; this is an inherent property of threshold schemes.
5) **Lack of confidentiality under public commitments (Feldman):** Feldman's commitments are fully deterministic and expose partial information about the secret when discrete logs can be computed. Hence, Feldman provides verifiability but not information-theoretic hiding.
6) **Commitment malleability under weak randomness (Pedersen):** Poor-quality or predictable randomness in Pedersen's commitments can enable correlation or bias attacks, potentially revealing relationships between shared secrets.
7) **Scalability limitations:** While verification cost scales linearly with threshold t, the dealer's commitment generation involves $O(n \times t)$ exponentiations, which can become costly for large participant groups.
8) **No built-in authentication or replay prevention:** Neither protocol ensures message authenticity or freshness. In distributed implementations, lack of authenticated channels allows replay or impersonation attacks if not combined with secure communication primitives.

Overall, the experiments confirm that both Feldman and Pedersen VSS achieve the intended verifiability and soundness guarantees under correct parameterization. The primary vulnerabilities arise not from cryptographic design flaws, but from parameter weakness, randomness misuse, and operational negligence. These findings underscore the importance of correct implementation discipline for maintaining VSS integrity and confidentiality.

G. Novel Contributions and Experimental Extensions: Feldman & Pedersen VSS

The Phase 3 study of the Feldman and Pedersen Verifiable Secret Sharing (VSS) schemes extends the implementation and validation work from earlier phases by introducing a detailed, data-driven security evaluation. While the earlier phases established theoretical correctness and functional verification, this phase investigates the integrity, soundness, and hiding properties through empirical testing. The following points summarize the novel experimental contributions and extensions developed for the VSS component:

1) **Empirical Verifiability Assessment:** Beyond formal proofs, randomized adversarial simulations were conducted to quantify how effectively both schemes detect dishonest dealers. The resulting detection rates—approximately 88.7% for Feldman and 91.2% for Pedersen—provide measurable evidence of verifiability in practice, bridging theoretical security claims with observable system behavior.
2) **Soundness Testing through Participant Forgery Simulation:** A controlled experiment was introduced to evaluate the schemes' resistance to forged shares. Randomly generated invalid shares were verified against public commitments, and the average forgery pass probability remained below 9%. This quantitative analysis empirically validates the soundness property and demonstrates practical robustness against participant-level cheating.
3) **Randomness-Reuse Vulnerability Demonstration:** A dedicated experiment exposed the effect of reusing the Pedersen blinding vector across multiple secret distributions. The recovery rate of polynomial coefficient differences reached 100%, confirming total loss of secrecy under randomness reuse. This visualization provides a clear empirical illustration of a critical implementation-level failure that is typically only discussed theoretically.
4) **Commitment and Verification Efficiency Profiling:** The time cost for share commitment and verification was measured across varying thresholds t. The result show near-linear scaling and microsecond-level execution times. This confirms that both Feldman and Pedersen maintain verifiability without significant computational overhead, ensuring practical deployability in threshold-based systems.
5) **Unified Comparative Evaluation:** The framework systematically compared Feldman and Pedersen schemes within identical modular group parameters. All tests—dealer cheating, forgery, randomness-reuse, and timing—were executed under a common environment, ensuring consistent statistical reliability and cross-scheme comparability.
6) **Integrated Security Dimension Analysis:** The experimental evaluation collectively covered the three principal security dimensions of VSS:
   - **Confidentiality** through Pedersen's hiding property and its failure under randomness reuse.
   - **Integrity** through detection of dealer cheating and participant forgery resistance.
   - **Availability** through timing and performance profiling ensuring practical verification feasibility.

   This integrated approach transforms the traditional correctness demonstration into a multi-dimensional security analysis.

Together, these contributions extend the study of Feldman and Pedersen VSS from functional correctness to full empirical validation. The experiments quantify real-world behavior under both normal and adversarial conditions, confirm the practical soundness of verification, and reveal the scheme's sensitivity to randomness misuse. The results establish a comprehensive understanding of the security-performance trade-offs inherent in verifiable secret sharing systems.

References

[1] G. R. Blakley, "Safeguarding cryptographic keys," in *Proc. National Computer Conference*, 1979, pp. 313–317.

[2] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, “Verifiable secret sharing and achieving simultaneity in the presence of faults,” in *Proc. 26th IEEE Symp. Foundations of Computer Science (FOCS)*, 1985, pp. 383–395.

[3] P. Feldman, “A practical scheme for non-interactive verifiable secret sharing,” in *Proc. 28th IEEE Symp. Foundations of Computer Science (FOCS)*, 1987, pp. 427–438.

[4] C. Asmuth and J. Bloom, “A modular approach to key safeguarding,” *IEEE Trans. Inf. Theory*, vol. 29, no. 2, pp. 208–210, Mar. 1983.

[5] Atakama Inc., “Multi-device encryption using VSS,” Whitepaper, 2021.