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 [**?**], 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 $(a_{i1}, \ldots, a_{it}, b_i)$, 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:** $t \in \{2, 4, 6, \ldots, 28\}$. 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 $t > 20$.

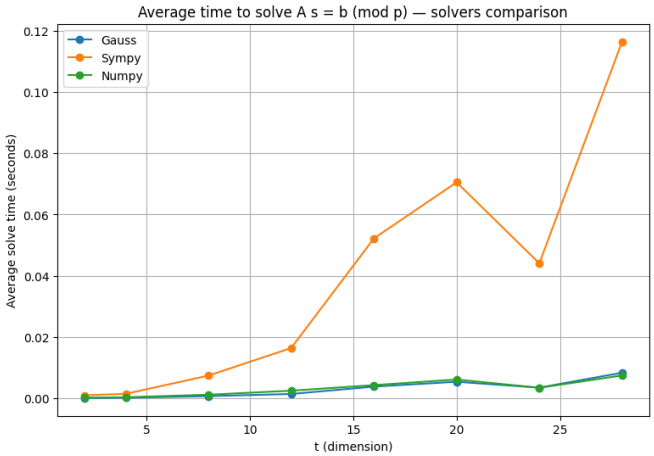

Fig. 1. Solver performance comparison: average time to solve $A\mathbf{s} = \mathbf{b} \pmod{p}$ as t increases. Error bars indicate one standard deviation.

- 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.

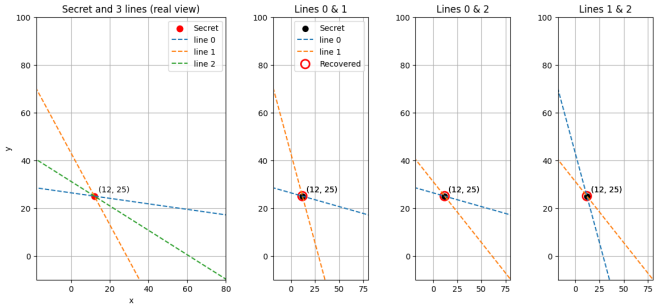

Fig. 2. Geometric interpretation for $t = 2$. Left: three lines intersecting at the secret point in $\mathbb{R}^2$. Right: pairwise intersections showing that any two shares uniquely reconstruct the secret.

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

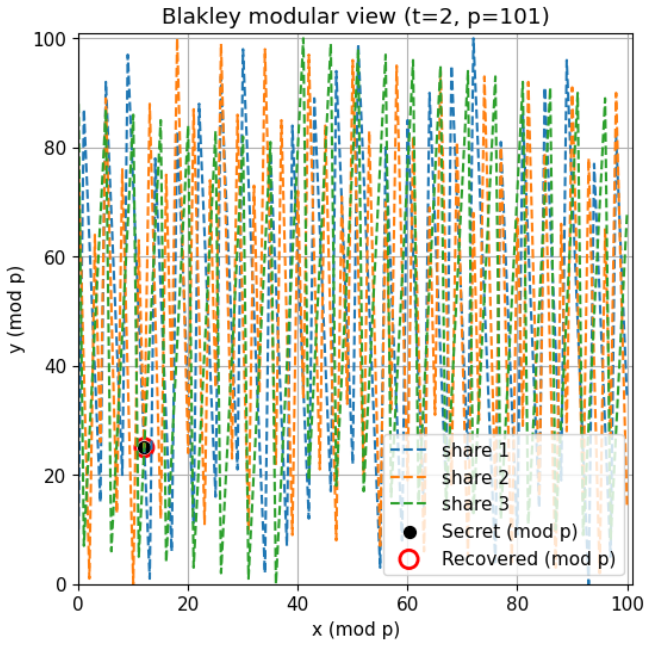

Fig. 3. Modular view of Blakley's scheme ($t = 2, p = 101$). Each dashed line represents a modular hyperplane wrapping periodically in $\mathbb{F}_{101}$. The red circle marks the recovered point, which coincides with the secret (black dot).

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.

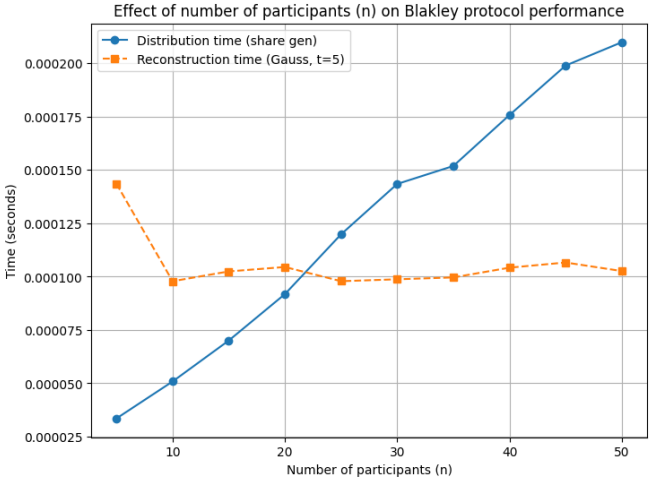

Fig. 4. Scalability analysis: distribution (share generation) and reconstruction time versus number of participants n. The threshold t remains fixed.

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.

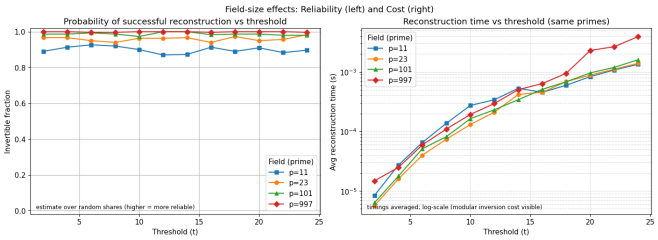

Fig. 5. Field-size analysis. Left: empirical invertible matrix fraction versus threshold t for various primes p. Right: average reconstruction time (log scale) versus t for the same primes.

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.

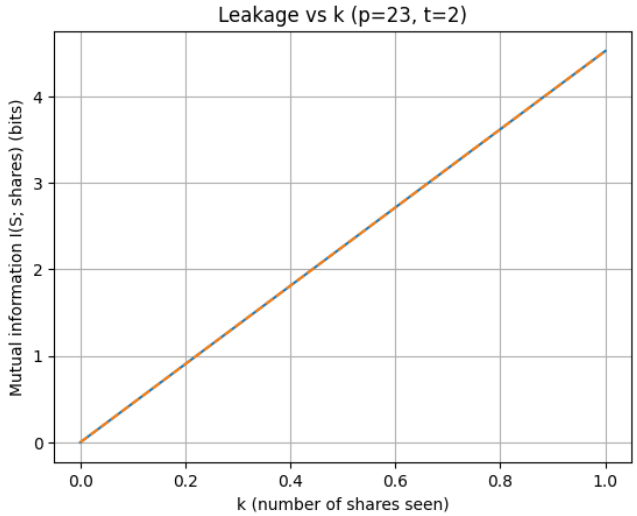

Fig. 6. Empirical information leakage $I(S; \text{shares}_k)$ for $p = 23$, $t = 2$. The dashed line denotes the theoretical slope of $\log_2 p$ bits per share.

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.

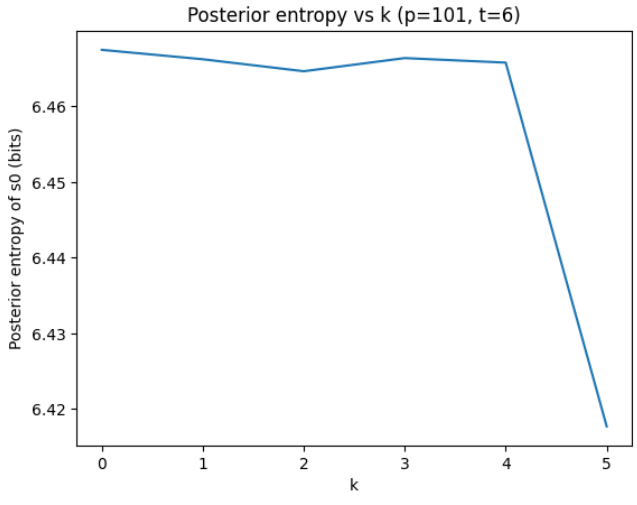

Fig. 7. Posterior entropy of coordinate s_0 as a function of known shares ($p = 101$, $t = 6$). Entropy remains stable until the full threshold set is reached.

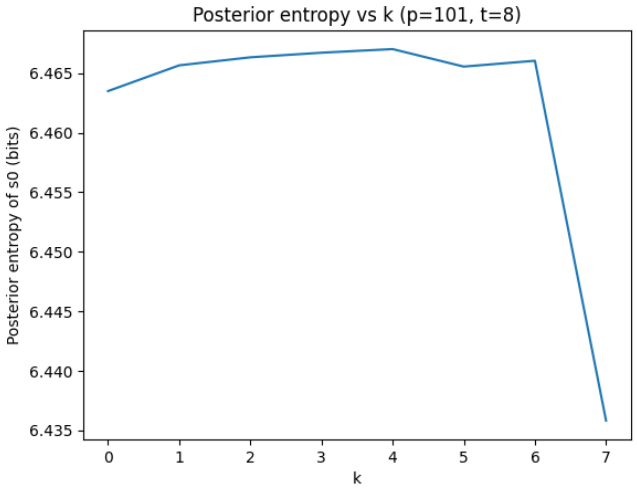

Fig. 8. Posterior entropy of coordinate s_0 for $p = 101$, $t = 8$. Uniform coefficient selection maintains consistent entropy below threshold.

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

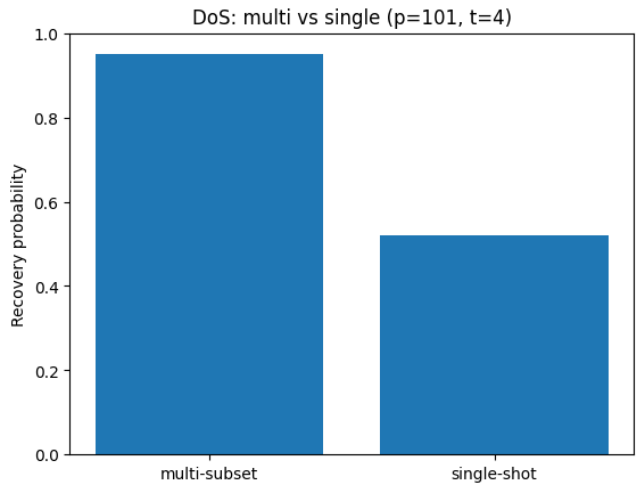

Fig. 9. Comparison of recovery probabilities under 15% share corruption for single-shot and multi-subset reconstruction ($p = 101$, $t = 4$).

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.