# Interpolation Methods and Cryptographic Primitives in Secret Sharing

Thays Rocha Neri Ferreira(✉) and Fábio Borges

Laboratório Nacional de Computação Científica, Petrópolis, RJ, Brazil
academicthaysrocha@gmail.com
http://www.lncc.br/borges/

**Abstract.** Shamir Secret Sharing (SSS) relies fundamentally on polynomial interpolation for secret reconstruction, making the choice of interpolation method a decisive factor for computational performance. Since reconstruction is the core operational phase of the scheme, differences in interpolation strategies directly affect efficiency. This work presents a systematic and comparative evaluation of interpolation techniques applied to SSS, including Lagrange, Newton, barycentric Lagrange, and NTT-based methods. Furthermore, the impact of distinct cryptographic primitives—Pedersen Commitments, Paillier, ElGamal, and RSA—is analyzed within the framework of Pedersen Verifiable Secret Sharing. All approaches were implemented in Python within a unified experimental environment to ensure consistent performance evaluation. The analysis considers execution time and memory usage as the primary evaluation metrics. The experimental results demonstrate that interpolation methods constitute the dominant factor influencing both temporal performance and memory usage. In particular, FFT/NTT-based methods outperform the alternatives for larger threshold values, benefiting from their asymptotic efficiency. Conversely, classical Lagrange interpolation consistently exhibits the lowest memory consumption among the evaluated techniques. With respect to verification mechanisms, RSA-based schemes achieve the best overall performance across the tested scenarios. Collectively, these findings offer concrete and practical guidance for designing efficient and scalable Secret Sharing deployments, especially in contexts where computational resources and performance constraints must be carefully balanced.

**Keyword:** Shamir secret sharing, pedersen verifiable secret sharing, polynomial interpolation, cryptographic primitives

## 1 Introduction

According to the guidelines of the National Institute of Standards and Technology (NIST) [11], recent technological advances increasingly highlight the urgent need to strengthen and modernize currently used cryptographic protocols through the adoption of decentralized strategies capable of increasing

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026 Workshops, LNCS 16759, pp. 99–116, 2027.
https://doi.org/10.1007/978-3-032-30518-3_7

the resilience, robustness, and security of systems against threats. Among the decentralized cryptographic protocols, Secret Sharing schemes stand out, allowing multiple parties to collaborate on a project using their private inputs without the need for undue disclosure of confidential information.

Shamir Secret Sharing (SSS), proposed in [15], stands out among Secret Sharing schemes for offering theoretical information security and a simple mathematical foundation, as cited in [3]. Its construction is based on classical principles of polynomial interpolation over finite fields. These characteristics explain the widespread adoption of SSS in both theoretical and practical applications, and it also serves as a basis for more sophisticated extensions, such as verifiable schemes and applications in secure computing and distributed systems.

However, several polynomial interpolation methods are available in the literature, of which only a limited number of studies systematically investigate their application in the context of the Shamir Secret Sharing scheme. Alternatives have been explored with the aim of improving aspects such as computational efficiency, scalability, and memory usage. The most comprehensive study identified to date is presented in [7], which performs a comparative analysis of the performance of seven different interpolation methods, including Newton, barycentric Lagrange, and the Fast Fourier Transform (FFT) applied to the SSS scheme. Unlike previous studies that focus only on interpolation techniques in Shamir Secret Sharing, this work separates the cost of reconstruction from the cost of verification and provides a joint evaluation of interpolation methods and cryptographic primitives in Verifiable Secret Sharing under a unified experimental framework.

In the context of Verifiable Secret Sharing (VSS) schemes, an additional requirement arises, related not only to the confidentiality of the secret but also to the need to guarantee the integrity and consistency of distributed actions, even in the presence of dishonest participants. As described in [16], these schemes extend Shamir's Secret Sharing by incorporating cryptographic mechanisms that allow participants to efficiently verify whether received shares are valid and consistent with a single underlying polynomial, without revealing the secret.

Despite the practical relevance of VSS schemes, the literature is still limited regarding the systematic analysis of the methods employed in verification. To the best of our knowledge, there are few studies that comparatively evaluate the impact of different cryptographic primitives on the performance of Verifiable Secret Sharing schemes.

Thus, with the aim of addressing the gaps related to the choice of interpolation methods in Shamir Secret Sharing and the cryptographic primitives employed in Verifiable Secret Sharing schemes, this article presents a comprehensive analysis of the main polynomial interpolation techniques, including Lagrange, Newton, barycentric Lagrange, and the Fast Fourier Transform, as well as the cryptographic primitives and commitment schemes including Pedersen commitments and public-key systems such as Paillier, ElGamal, and RSA. All approaches considered were implemented in Python, allowing for a uniform and comparable evaluation of their computational performance.

In general, this article proposes the following main research questions:

(i) Which method is most execution-time efficient for secret reconstruction in large-scale scenarios?
(ii) Which method has the highest memory consumption for secret reconstruction in large-scale scenarios?
(iii) Which cryptographic primitive shows the best time performance in verifying participants' honesty in the Pedersen Verifiable Secret Sharing scheme?
(iv) Which cryptographic primitive exhibits the highest memory consumption in the Pedersen Verifiable Secret Sharing scheme?

These questions motivated and guided the research, forming the basis of this work. Thus, this article contributes primarily to the following:

(i) Systematic comparison of four interpolation techniques for secret reconstruction.
(ii) Evaluation of four cryptographic primitives in Pedersen-based VSS.
(iii) Experimental analysis isolating interpolation cost from verification cost.

The remainder of the article is organized as follows: Sect. 2 presents the essential foundations, covering algebraic theory, Secret Sharing, the considered cryptographic primitives, and the analyzed polynomial interpolation methods. Section 3 describes the experimental setup and the procedures adopted for execution time and memory consumption analysis. Section 4 presents the experimental results obtained. Finally, Sect. 5 concludes the article and outlines directions for future research.

## 2 Preliminaries

This section introduces the preliminary concepts underlying the proposed analysis, covering algebraic foundations, Secret Sharing, selected cryptographic primitives, and polynomial interpolation methods.

### 2.1 Algebraic Theory

This section introduces the algebraic foundations that support the cryptographic schemes and polynomial interpolation techniques employed in this work.

**Finite Field.** A finite field, also known as a Galois Field (GF), is an algebraic structure composed of a finite number of elements. In this set, the operations of addition and multiplication, as well as their inverses, subtraction and division, with the exception of division by zero, are well-defined and satisfy fundamental properties such as associativity, commutativity, and distributivity. The order of a finite field is always a power of a prime number, that is, $p^n$, where $p$ is a prime number and $n$ is a positive integer. For more details such as theorems and axioms, see [8].

**Root of Unity.** The roots of unity are elements of a modular ring that satisfy a property analogous to the classical case: a root of unity of order $n$ is an element $r$ such that $r^n \equiv 1 (\mod M)$, where $M$ is the modulus chosen for the NTT. These roots are fundamental to the definition of the NTT (Number Theoretic Transform), which can be seen as a discrete analogue of the Fourier Transform, but operating in modular arithmetic. This approach allows for efficient transformations on integers, without rounding errors.

**Definition 1.** *Let $\mathbb{F}_p$ be a finite field with prime modulus $p$. An element $\omega \in \mathbb{F}_p$ is called an nth root of unity if it satisfies*

$$\omega^n \equiv 1 \pmod{p}. \tag{1}$$

*Moreover, if $\omega^k \not\equiv 1 \pmod{p}$ for all $0 < k < n$, then $\omega$ is called a primitive nth root of unity. In this case, $\omega$ generates the multiplicative cyclic subgroup*

$$\{1, \omega, \omega^2, \ldots, \omega^{n-1}\} \subset \mathbb{F}_p^*, \tag{2}$$

*whose powers consist of exactly $n$ distinct roots of unity.*

According to [9], in the context of the FFT over finite fields, such roots of unity allow the construction of the Discrete Fourier Transform (DFT) matrix, enabling an efficient evaluation of polynomials at multiple points and resulting in a reduction of computational complexity from $O(n^2)$ to $O(n \log n)$.

### 2.2 Secret Sharing

Secret sharing is a cryptographic scheme that allows a secret, such as a cryptographic key, a password, or any other sensitive data, to be divided into multiple parts, denoting shares. These parts are distributed among different participants. The main characteristic of this scheme is that the original secret can only be reconstructed when a minimum number of participants, known as the threshold, cooperate by combining their actions. If the number of participants is less than this limit, the secret remains unrecoverable, and no relevant information can be compromised by individual actions or any subset that does not meet the established limits.

**The Shamir Secret Sharing Scheme.** The Shamir Secret Sharing scheme (SSS), proposed by Shamir [15], is based on polynomial interpolation over finite fields. The steps of the scheme are described below.

I **Share generation.** The dealer $\mathcal{D}$ selects a large prime number $p$, which defines the finite field, and, given the secret $s$, constructs a polynomial of degree $k-1$ of the form

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{k-1} x^{k-1} \pmod{p}, \tag{3}$$

where $a_0 = s$, $p$ is a prime satisfying $p > n$, with $n$ denoting the total number of participants and $k$ the reconstruction threshold. The coefficients $a_1, a_2, \ldots, a_{k-1} \in \mathbb{Z}_p$ are chosen uniformly at random.

II **Distribution.** The dealer $\mathcal{D}$ computes the $n$ shares of the scheme, $\text{share}_i = (i, f(i))$, and sends them to each participant $\mathcal{P}_i$ through a secure channel, for $i = 1$ to $n$.

III **Reconstruction.** For the reconstruction phase, it is assumed[1] that the participants $\{\mathcal{P}_i\}_{i=1}^{k}$ send their *shares* $\{s_i\}_{i=1}^{k}$. Using Lagrange interpolation over these shares, the secret is recovered as follows:

$$s = f(0) = \sum_{i=1}^{k} y_i \lambda_i \tag{4}$$

with

$$\lambda_i = \prod_{j=1, j\neq i}^{k} \frac{x_j}{x_j - x_i} \tag{5}$$

where the term $y_i$ represents the share value of participant $\mathcal{P}_i$.

**Pedersen's Verifiable Secret Sharing.** Pedersen's Verifiable Secret Sharing (VSS) scheme is a cryptographic method that, in addition to dividing the secret into distributed parts among the participants, allows for verification of the validity and consistency of the shares without revealing the secret itself. It uses commitments based on discrete logarithms to ensure that the parts are consistent and that the distributor is honest, offering robustness against fraud or incorrect parts, as stated by [10]. The steps of Pedersen's Verifiable Secret Sharing scheme are described below.

I **Commitment and share generation.** The dealer $\mathcal{D}$ selects a finite field $\mathbb{Z}_p$, where $p$ is a large prime number, and two generators $g, h \in \mathbb{Z}_p^*$ such that the discrete logarithm of $h$ with respect to base $g$ is unknown. Given the secret $s$, the dealer constructs two polynomials of degree $k-1$:

$$f(x) = a_0 + a_1 x + \cdots + a_{k-1} x^{k-1} \pmod{p}, \tag{6}$$

$$r(x) = b_0 + b_1 x + \cdots + b_{k-1} x^{k-1} \pmod{p}, \tag{7}$$

where $a_0 = s$, $b_0$ is chosen uniformly at random, $p > n$, and all coefficients $a_i, b_i \in \mathbb{Z}_p$ are chosen uniformly at random, with the exception of $a_0$.
The dealer then computes the public commitments to the coefficients:

$$C_j = g^{a_j} h^{b_j} \pmod{p}, \quad j = 0, 1, \ldots, k-1. \tag{8}$$

II **Distribution.** The dealer $\mathcal{D}$ computes the $n$ shares of the secret and the random value, $\text{share}_i = (i, f(i), r(i))$, and sends them to each participant $\mathcal{P}_i$, for $i = 1$ to $n$, through a secure channel. The commitments $\{C_j\}_{j=0}^{k-1}$ are made public.

[1] Note: the notation emphasizes a change in cardinality rather than in the indices of the participants.

III **Share verification.** Each participant $\mathcal{P}_i$ can verify the validity of their share by checking whether the following equality holds:

$$g^{f(i)}h^{r(i)} \equiv \prod_{j=0}^{k-1} C_j^{i^j} \pmod{p}. \tag{9}$$

If the equality does not hold, the participant detects dishonest behavior by the dealer.

IV **Reconstruction.** In the reconstruction phase, it is assumed that the participants $\{\mathcal{P}_i\}_{i=1}^{k}$ send their valid shares $\{f(i)\}_{i=1}^{k}$. Using Lagrange polynomial interpolation over these values, the secret is recovered as

$$s = a_0 = f(0) = \sum_{i=1}^{k} f(i) \prod_{\substack{j=1 \\ j\neq i}}^{k} \frac{-j}{(i-j)} \pmod{p}. \tag{10}$$

### 2.3 Cryptographic Primitives

Cryptographic primitives are fundamental elements in the construction of information security protocols and schemes, providing mathematical mechanisms that ensure properties such as confidentiality, integrity, protection, and verifiability. This section presents the main primitives used in this work: Pedersen Commitments, as well as those associated with the Paillier, ElGamal, and RSA public-key cryptosystems. Each relies on different hard mathematical problems, such as discrete logarithms and integer factorization, and exhibits specific properties.

The cryptographic primitives described below are based on the works of [5,12–14] respectively.

#### Pedersen Commitments

**Definition 2.** *Let $p$ be a large prime, $G$ a cyclic group of order $q$ (with $q \mid p-1$), and let $g, h \in G$ be two generators such that the discrete logarithm of $h$ with respect to base $g$ is unknown. The Pedersen commitment scheme is defined as*

$$C = g^m h^r \pmod{p},$$

*where $m \in \mathbb{Z}_q$ is the message to be committed and $r \in \mathbb{Z}_q$ is a random value.*

#### Paillier Cryptosystem

**Definition 3.** *Let $p$ and $q$ be two large and distinct prime numbers, $n = pq$, and $\lambda = \operatorname{lcm}(p-1, q-1)$. Consider the multiplicative group $\mathbb{Z}_{n^2}^*$ and an element $g \in \mathbb{Z}_{n^2}^*$ such that $\gcd(L(g^\lambda \bmod n^2), n) = 1$, where the function $L$ is defined by*

$$L(u) = \frac{u-1}{n}.$$

*The Paillier cryptosystem is defined as follows:*

- ***Public key:*** $(n, g)$;
- ***Private key:*** $\lambda$.

*The encryption of a message* $m \in \mathbb{Z}_n$ *is given by* $c = g^m r^n \pmod{n^2}$, *where* $r \in \mathbb{Z}_n^*$ *is chosen uniformly at random.*

**ElGamal Cryptosystem**

**Definition 4.** *Let* $G$ *be a cyclic group of order* $q$ *generated by* $g$, *where the discrete logarithm problem is computationally hard. Let* $x \in \mathbb{Z}_q$ *be the private key and let* $h = g^x$ *be the public key.*

*The ElGamal cryptosystem is defined as follows:*

- ***Public key:*** $(G, q, g, h)$;
- ***Private key:*** $x$.

*The encryption of a message* $m \in G$ *is given by* $(c_1, c_2) = (g^r, m \cdot h^r)$, *where* $r \in \mathbb{Z}_q$ *is chosen uniformly at random.*

**RSA Cryptosystem**

**Definition 5.** *Let* $p$ *and* $q$ *be two large and distinct prime numbers,* $n = pq$, *and* $\varphi(n) = (p-1)(q-1)$. *Choose an integer* $e$ *such that* $\gcd(e, \varphi(n)) = 1$, *and let* $d$ *be the multiplicative inverse of* $e$ *modulo* $\varphi(n)$, *that is,*

$$ed \equiv 1 \pmod{\varphi(n)}.$$

*The RSA cryptosystem is defined as follows:*

- ***Public key:*** $(n, e)$;
- ***Private key:*** $d$.

*The encryption of a message* $m \in \mathbb{Z}_n$ *is given by* $c = m^e \pmod{n}$, *and the decryption by* $m = c^d \pmod{n}$.

### 2.4 Polynomial Interpolation Methods

In the context of interpolation over a finite field $\mathbb{F}_p$ (where $p$ is a prime number or a power of a prime), the operations of addition, subtraction, multiplication, and division (except by zero) are performed modulo $p$ or modulo an irreducible polynomial over $\mathbb{F}_p$ (in the case of $p = q^k$).

Several methods have been developed to perform polynomial interpolation, each with its own characteristics, as well as computational advantages and improvements. We will analyze four of them, namely, Lagrange, Newton, barycentric Lagrange, and the Fast Fourier Transform (FFT), known in the finite domain as the Number Theoretic Transform (NTT).

For standardization, given a set of $n + 1$ distinct points

$$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n), \tag{11}$$

where all the values $x_i$ are distinct from each other, the main interpolation methods mentioned are defined below.

**Lagrange.** The Lagrange interpolation polynomial is the unique polynomial of degree at most $n$ that satisfies $P(x_i) = y_i$ for all $i = 0, 1, \ldots, n$.

According to [4], the general form of the classical Naive Lagrange polynomial, also known simply as the Lagrange polynomial, $P(x)$ is given by

$$P(x) = \sum_{j=0}^{n} y_j L_j(x). \tag{12}$$

In this expression, $L_j(x)$ are the Lagrange basis polynomials, defined by

$$L_j(x) = \prod_{\substack{k=0 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}. \tag{13}$$

Each polynomial $L_j(x)$ satisfies the interpolation property $L_j(x_j) = 1$ and $L_j(x_k) = 0$ for all $k \neq j$.

**Newton.** The Newton interpolation polynomial is the unique polynomial of degree at most $n$ that satisfies $P(x_i) = y_i$ for all $i = 0, 1, \ldots, n$. According to [4], the general form of the Newton polynomial $P_n(x)$ is given by

$$\begin{aligned} P_n(x) =& f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \\ & f[x_0, x_1, \ldots, x_n](x - x_0)(x - x_1) \ldots (x - x_{n-1}). \end{aligned} \tag{14}$$

In this expression, $f[x_i, x_{i+1}, \ldots, x_j]$ denotes the divided differences of the function $f$. The divided differences are computed recursively, as described below:

– **Zeroth-order divided difference:**

$$f[x_i] = y_i. \tag{15}$$

– **Divided difference of order $k \geq 1$:**

$$f[x_i, x_{i+1}, \ldots, x_{i+k}] = \frac{f[x_{i+1}, \ldots, x_{i+k}] - f[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}. \tag{16}$$

**Barycentric.** The barycentric interpolation method is a reformulation of Lagrange interpolation, designed to improve numerical stability and computational efficiency. Instead of directly computing the Lagrange basis polynomials, barycentric interpolation uses precomputed weights. As described in [2], the interpolating polynomial $P(x)$ associated with $n + 1$ distinct points can be expressed in barycentric form as

$$P(x) = \frac{\sum_{j=0}^{n} \frac{w_j y_j}{x - x_j}}{\sum_{j=0}^{n} \frac{w_j}{x - x_j}}, \tag{17}$$

where the barycentric weights are defined by

$$w_j = \frac{1}{\prod_{\substack{k=0 \\ k \neq j}}^{n} (x_j - x_k)}. \tag{18}$$

**NTT.** The Fast Fourier Transform ($FFT$) is an efficient algorithm for computing the Discrete Fourier Transform (DFT). The FFT makes this computation significantly faster than the direct DFT approach, especially for large-scale data sets. However, when considered over a finite domain, known as the Number Theoretic Transform ($NTT$), it operates on sequences of numbers embedded in a discrete and finite mathematical system.

According to [9], in a finite field $\mathbb{F}_p$, with $p$ prime, the DFT of a sequence $a = (a_0, a_1, \ldots, a_{n-1})$ is defined as

$$A_k = \sum_{j=0}^{n-1} a_j \, \omega^{jk} \pmod{p}, \quad k = 0, 1, \ldots, n-1, \tag{19}$$

where $\omega \in F_p$ is a primitive $n$th root of unity in $\mathbb{F}_p$, that is, $\omega^n \equiv 1 \pmod{p}$ and $\omega^k \not\equiv 1 \pmod{p}$ for $0 < k < n$.

The inverse transform, which recovers the original sequence $a_j$ from $A_k$, is given by

$$a_j = n^{-1} \sum_{k=0}^{n-1} A_k \, \omega^{-jk} \pmod{p}, \tag{20}$$

where $n^{-1}$ denotes the multiplicative inverse of $n$ modulo $p$.

As in the FFT, the NTT employs a divide-and-conquer strategy, splitting the problem into two smaller NTTs, one associated with even indices and the other with odd indices, and combining the results as follows:

$$A_k = A_k^{(\text{even})} + \omega^k A_k^{(\text{odd})} \pmod{p}, \tag{21}$$

$$A_{k+n/2} = A_k^{(\text{even})} - \omega^k A_k^{(\text{odd})} \pmod{p}, \tag{22}$$

for $k = 0, 1, \ldots, \frac{n}{2} - 1$.

This approach reduces the computational complexity from $O(n^2)$, characteristic of the direct modular DFT, to $O(n \log n)$, making the NTT particularly suitable for applications that require fast and accurate modular operations, such as optimized secret reconstruction in Secret Sharing schemes.

Although the FFT and the NTT share the same algorithmic structure, the NTT imposes additional mathematical constraints that affect its practical application. Implementing the NTT requires extra care to avoid arithmetic overflow and to ensure computational efficiency, especially when large moduli are used. As a result, the NTT is less flexible than the FFT in general-purpose signal processing applications, but extremely valuable in contexts that demand modular arithmetic.

## 3 Experimental Setup and Simulation

The experiments were conducted on a workstation equipped with an Intel Core i9-12900K (16 cores, 3.2 GHz), 125 GB RAM,Ubuntu 22.04, Python 3.12.3. The reported results correspond to the average of 10 independent repetitions of each experiment, with the goal of mitigating the presence of outliers. Execution time was measured using the built-in `time` module, and memory consumption was monitored using the `tracemalloc` library.

For modular inversion, Fermat's Little Theorem was employed. This simple and uniform approach is executed in logarithmic time with respect to $p$, as stated in [17]. For one experiment, we defined the finite field with $p = 2013265921$, the secret $= 987654321$, the generators $g = 31$ and $h = 17$, in addition to (Table 1).

**Table 1.** Values of $n_{\text{sss}}$ (total number of participants) and $k_{\text{sss}}$ (minimum threshold for reconstruction).

| $n_{\text{sss}}$ | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 | 8192 | 16384 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $k_{\text{sss}}$ | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 | 8192 |

### 3.1 Time Analysis

The first topic to be analyzed is the difference in computational performance depending solely on the interpolation method used for secret reconstruction. To this end, we modified the traditional Shamir Secret Sharing method to allow the use of the methods mentioned in Sect. 2.4. Next, for each interpolation method analyzed previously, we added a function to verify the honesty of the participants based on Pedersen's Verifiable Secret Sharing (VSS). For this, we used the cryptographic primitives described in Sect. 2.3. In all analyses performed, the unit of measurement considered is seconds.

### 3.2 Memory Analysis

Similarly to the previous analysis, memory usage was first evaluated solely for the interpolation method. Subsequently, the cryptographic primitives were assessed according to the adopted interpolation method, enabling a combined analysis of actual memory usage throughout the entire process, considering both components. Finally, peak memory usage was also analyzed for all previously described scenarios. The key difference is that peak memory refers to the maximum amount allocated at any point during the monitoring period, regardless of later releases, whereas memory usage represents consumption at the moment of measurement. In all analyses, the unit of measurement used was megabytes (MB).

## 4 Experimental Results

This section presents the experimental results obtained from the evaluation of the considered interpolation methods and cryptographic primitives, focusing on computational performance through execution time and memory consumption analysis. Due to the nature of the results, showing differences on the order of microseconds and very close values, we chose to present all graphs on a logarithmic scale. This allows for a more balanced comparison among the methods and clearer identification of trends that would be difficult to observe in a linear representation.

### 4.1 Time Analysis

As described in Sect. 3, the first aspect analyzed is the variation in computational performance as a function solely of the interpolation method used in the secret reconstruction. Thus, we present Fig. 1. In it, we can see that, among the methods mentioned, the Newton interpolation method, indicated in orange, showed the worst performance, being considered the slowest. Next, the Lagrange method, indicated in blue, and the Barycentric Lagrange, indicated in purple, appear, while the NTT-based method, indicated in pink, obtained the best computational performance.

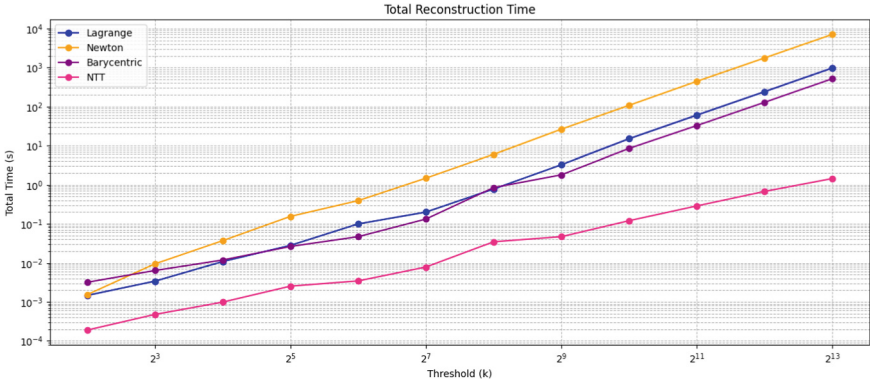


**Fig. 1.** Time analysis for secret reconstruction depends on the interpolation method used.

Comparing the asymptotic complexity of the methods, the NTT approach showed the best performance, which was expected due to its $O(n \log n)$ complexity, in contrast to the $O(n^2)$ complexity of the other methods. Although Newton's method shares the same quadratic complexity, it involves divided difference computations and recursive polynomial evaluation, increasing the number of required operations and, consequently, processing time. Lagrange's method computes the interpolating polynomial directly, requiring multiplications and divisions for each

point. Barycentric interpolation optimizes Lagrange's formulation by reducing the number of operations during polynomial evaluation, making it slightly more efficient.

It is also worth highlighting the most significant occurs at the equivalence threshold of $2^5$, which is when the Lagrange and Barycentric methods intersect. At $2^8$, they become equivalent, but at $2^9$ the Barycentric method once again stands out as faster than the other two.

Now, in Fig. 2, we present an analysis of the impact of verifying the honesty of participants using Pedersen's VSS with the aforementioned cryptographic primitives, on the interpolation methods. This analysis shows that the behavior of the curves, regardless of the cryptographic primitive used, is very similar. Consequently, we can conclude that the type of primitive used does not significantly influence the temporal performance.

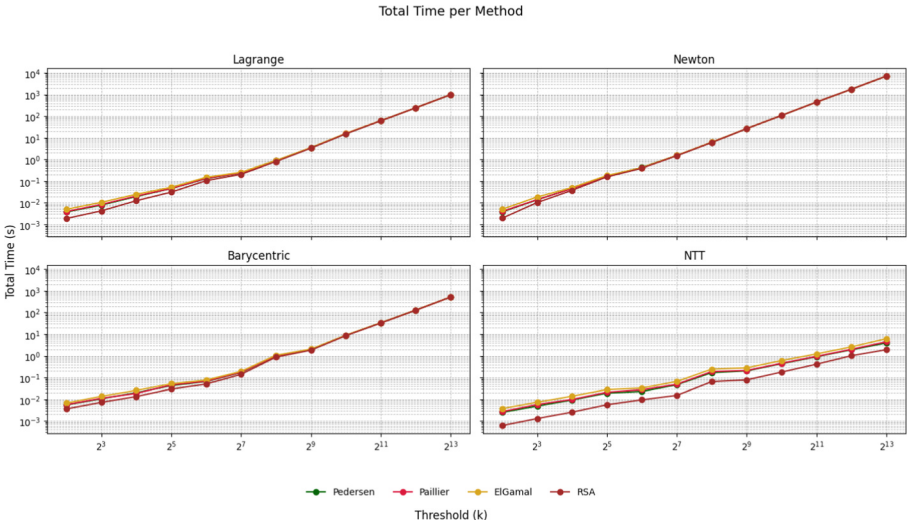


**Fig. 2.** Time analysis for share verification depends on the interpolation method and the cryptographic primitive used.

### 4.2 Memory Analysis

As with the previous analysis, the first aspect analyzed is the variation in memory usage as a function solely of the interpolation method used in secret reconstruction. Thus, we present Fig. 3. In it, we can see that, among the methods analyzed, NTT presented the worst performance, being the algorithm that consumes the most memory throughout the entire process. Next, Newton's method stands out, exhibiting oscillatory behavior in the interval from $2^2$ to $2^6$ interval, stabilizing from $2^7$ onwards, when it begins to show practically constant memory

consumption. In turn, the Barycentric and Lagrange methods showed constant behavior, regardless of the threshold value $k$. Lagrange's method stands out, having obtained the best overall performance, registering the lowest memory consumption among the methods evaluated.

NTT performs complete transformations and complex modular operations, consuming temporary memory during the process. Newton's method constructs a divided difference table, requiring the storage of all intermediate subresults, which significantly increases memory usage. The oscillation observed in the interval from $2^2$ to $2^6$ can be attributed to the incremental pattern of construction of this table: in smaller sizes, small variations in the number of points cause internal reorganizations and reallocations of data structures, temporarily impacting memory usage. From $2^7$ onwards, the structure tends to stabilize, operating with a more uniform allocation pattern and, consequently, with practically constant consumption.

The Barycentric Lagrange is an optimized version of Lagrange interpolation, designed to reduce the cost of evaluating polynomials at multiple points. However, it requires storing the interpolation weights, which increases temporary memory allocation. In contrast, the direct Lagrange method calculates each point independently, without the need to store large intermediate tables, resulting in minimal memory usage and more straightforward calculations.

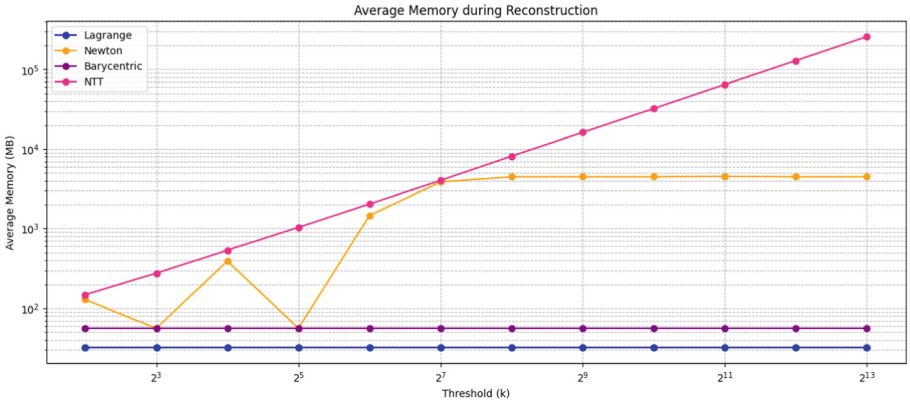


**Fig. 3.** Memory analysis used to reconstruct the secret depends on the interpolation method used.

Now, analyzing only the cryptographic primitives, regardless of the interpolation method, we observe a quite similar behavior among them. In Fig. 4, it can be observed that Pedersen, Paillier, ElGamal, and RSA present practically overlapping curves in all scenarios, with marginal and stable differences as the threshold $k$ increases.

This pattern indicates that memory consumption is strongly influenced by the structure of the reconstruction method adopted and not by the cryptographic

primitive itself. The choice of interpolation technique has a much more significant impact on memory usage than the cryptographic scheme used for verification or protection of shares. In summary, we can present the memory usage of the cryptographic primitives as shown in Table 2 and Fig. 5 below, with the maximum difference between the primitives being 100 MB (Fig. 4).

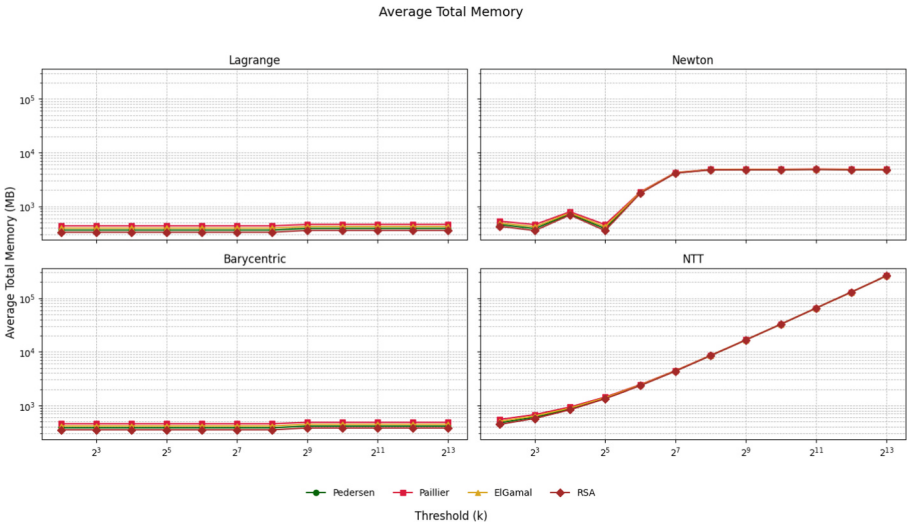


**Fig. 4.** Memory analysis as a function of the cryptographic primitive used.

Using Fig. 5 as a basis and comparing the primitives among themselves, RSA consistently exhibits the lowest memory usage. This is expected, as its internal operations essentially involve modular exponentiation with few auxiliary objects. However, it is worth noting that this is also a result of the parameters used in the simulation. The public exponent equivalent to $e = 65537$ was used, which is commonly used in RSA cryptosystems because it provides a balance between security and computational efficiency, but small enough to allow efficient cryptography, as stated by [6]. Furthermore, the values of $p$ and $q$ have a size of 16 bits, and the modulus of $N$ has a size of 32 bits. These parameter sizes are intentionally insecure and were selected exclusively to evaluate computational performance rather than cryptographic security. According to [1], these values are considered insecure, but adequate for performance tests, such as evaluating execution time and memory usage, where security is not the primary focus.

Pedersen Commitments come next, with a slightly higher cost. This increase is explained by the need to store commitments, generators, and auxiliary values for verification. The ElGamal algorithm consumes more memory than the Pedersen and RSA algorithms, reflecting the fact that it operates with pairs of group elements and additional random values, which duplicates part of the internal structures during encryption and verification. Paillier's algorithm, on the other

**Table 2.** Summary of average memory usage (in MB) for different encryption primitives.

| $k$ | Pedersen | Paillier | ElGamal | RSA |
|---|---|---|---|---|
| $2^2$ | 328 | 400 | 360 | 296 |
| $2^3$ | 328 | 400 | 360 | 296 |
| $2^4$ | 328 | 400 | 360 | 296 |
| $2^5$ | 328 | 400 | 360 | 296 |
| $2^6$ | 328 | 400 | 360 | 296 |
| $2^7$ | 328 | 400 | 360 | 296 |
| $2^8$ | 356 | 428 | 388 | 324 |
| $2^9$ | 356 | 428 | 388 | 324 |
| $2^{10}$ | 356 | 428 | 388 | 324 |
| $2^{11}$ | 356 | 428 | 388 | 324 |
| $2^{12}$ | 356 | 428 | 388 | 324 |
| $2^{13}$ | 356 | 428 | 388 | 324 |

hand, is the most memory-intensive in all scenarios. This is directly related to the use of large integers, operations on larger moduli ($n^2$), and the maintenance of multiple intermediate values during homomorphic encryption and decryption, which significantly increases the pressure on memory.

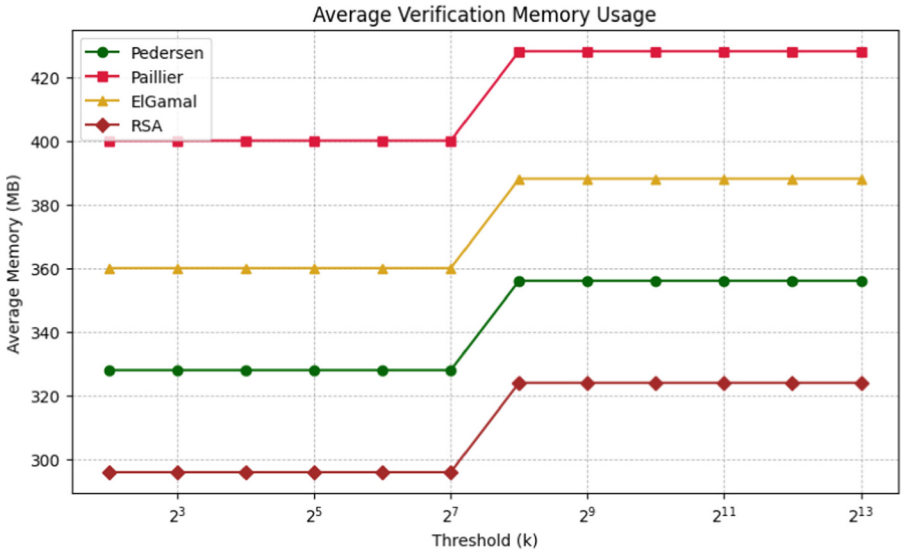


**Fig. 5.** Average memory usage (in MB) for different encryption primitives.

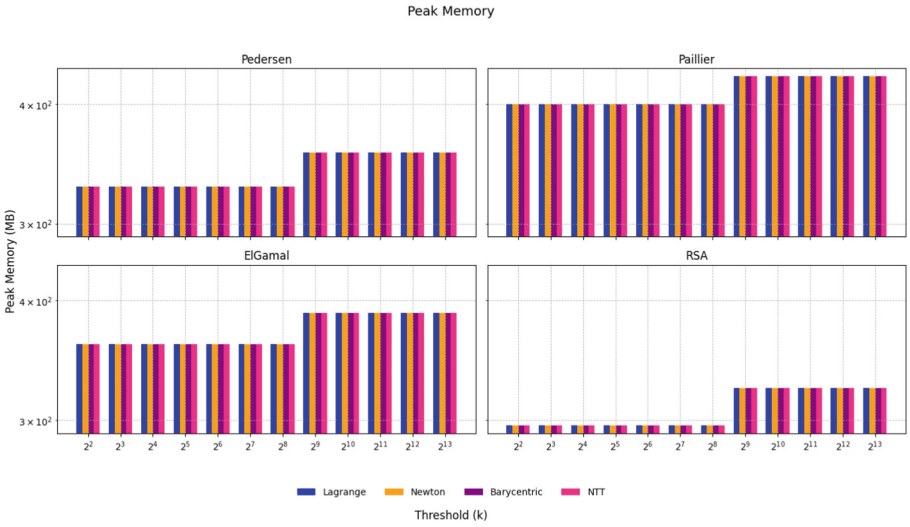


**Fig. 6.** Analysis of the peak memory required for secret reconstruction as a function of the interpolation method and the cryptographic primitive.

We also analyzed the memory peak, shown in Fig. 5, to reveal the most critical moment of execution. The memory peak is what determines the execution prediction on devices with limited memory, such as smart meters, for example. In all cryptographic primitives, the memory peak was constant, with jumps between limits equivalent to $2^8$ and $2^9$. This indicates that the peak does not gradually increase with the number of participants, but is controlled by internal allocation limits, typical of vectors (Fig. 6).

The influence of primitive cryptography, on the other hand, is quite clear in the absolute value of the peak. RSA consistently shows the lowest memory peaks, reflecting its simpler structure. Pedersen and ElGamal occupy an intermediate range, while Paillier shows the highest memory peaks for all values of $k$.

## 5 Conclusion

First, this work analyzes the performance of the Lagrange interpolation method in Shamir's Secret Sharing compared to other approaches. Although the security of the scheme stems from the polynomial construction over a finite field, the cost of reconstructing the secret depends directly on the interpolation method adopted. By comparing them, it is possible to dissociate the intrinsic cost of the cryptographic scheme from that introduced by the reconstruction algorithm, allowing for a more precise and reproducible evaluation.

Similar logic applies to the verification mechanism in Pedersen's VSS. Considering cryptographic primitives alternative to Pedersen's commitments allows for evaluating how different security assumptions impact performance and resource

consumption. This makes it possible to analyze the trade-offs between security, efficiency, and implementability, isolating the effect of the commitment primitive and selecting the most suitable option without compromising the guarantees of the VSS.

The study, therefore, separates two layers that are usually combined: interpolation cost and verification cost. In interpolation, the NTT-based method presents the best time. In contrast, NTT requires higher memory consumption. This difference is relevant in practice: NTT is more suitable for scenarios requiring low latency and frequent reconstructions, such as aggregation in smart meters, while Lagrange is preferable in memory-constrained environments, such as embedded devices and IoT sensors.

Regarding primitives, the RSA-based scheme consistently presents the shortest execution times for all values of k and interpolation methods, followed by Pedersen, ElGamal, and Paillier, the latter with the highest computational cost. From $2^8$ onwards, a discrete and uniform increase in memory is observed across all primitives, suggesting the allocation of auxiliary structures. In general, the impact of k on memory is reduced when compared to the influence of the chosen primitive.

Despite the measurable differences between the primitives, the dominant factor for time and memory is the interpolation method. Thus, the inclusion of verification steps is technically justified, as its cost does not significantly alter the overall performance profile. Its adoption enhances the reliability and robustness of the system, mitigating malicious behavior or failures without significant penalties, strengthening the scheme for applications that require explicit guarantees of correctness and security.

Future work includes extending the analysis to larger cryptographic parameters, implementing the algorithms in compiled languages, and evaluating their performance in distributed environments and real-world applications such as secure multiparty computation and federated learning systems.

**Acknowledgments.** We thank the National Laboratory for Scientific Computing (LNCC) and CAPES for their institutional support, as well as the team of collaborators for their valuable contributions that enriched the development of this work.

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

## References

1. Amalarethinam, D.: Enhanced RSA algorithm with varying key sizes for data security in cloud. 2017 World Congress on Computing and Communication Technologies (WCCCT) pp. 172–175 (2017). https://doi.org/10.1109/wccct.2016.50
2. Berrut, J.P., Trefethen, L.N.: Barycentric Lagrange interpolation. SIAM Rev. **46**(3), 501–517 (2004). https://doi.org/10.1137/S0036144502417715
3. Bogdanov, D.: Foundations and properties of Shamir's Secret Sharing Scheme – Research Seminar in Cryptography. University of Tartu, Institute of Computer Science **1** (2007)

4. Burden, R.L., Faires, J.D.: Numerical Analysis, 10th edn. Cengage Learning, Boston, MA, USA (2021)
5. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory **31**(4), 469–472 (1985)
6. Feng, Y., Nitaj, A., Pan, Y.: Small public exponent brings more: Improved partial key exposure attacks against RSA. IACR Cryptol. ePrint Arch **2024**, 1329 (2024). https://doi.org/10.62056/ahjbhey6b
7. Ferreira, T.R.N., Borges, F.: Performance evaluation of interpolation techniques in shamir secret sharing scheme. IEEE Access (2026). https://doi.org/10.1109/ACCESS.2026.3679331
8. Garling, D.J.H.: Finite fields. In: Galois Theory and Its Algebraic Background, Cambridge University Press (2021)
9. Lidl, R., Niederreiter, H.: Introduction to Finite Fields and Their Applications, 2nd edn. Cambridge University Press, Cambridge (1997)
10. Mahmoud, Q.A.: Polynomial differential-based information-theoretically secure verifiable secret sharing. International Journal of Information Technology and Computer Science **6**, 18–23 (2014). https://doi.org/10.5815/ijitcs.2014.12.03
11. National Institute of Standards and Technology: Nist threshold cryptography project. https://csrc.nist.gov/Projects/threshold-cryptography (2025), accessed: Jul. 21, 2025
12. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: International conference on the theory and applications of cryptographic techniques, pp. 223–238. Springer (1999)
13. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Annual international cryptology conference, pp. 129–140. Springer (1991)
14. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978)
15. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979). https://doi.org/10.1145/359168.359176
16. Stadler, M.: Publicly verifiable secret sharing pp. 190–199 (1996). https://doi.org/10.1007/3-540-68339-9_17
17. Xu, S., Gu, H., Wang, L., Guo, Z., Liu, J., Lu, X., Gu, D.: Efficient and constant time modular inversions over prime fields. 2017 13th International Conference on Computational Intelligence and Security (CIS) pp. 524–528 (2017). https://doi.org/10.1109/cis.2017.00122