Survey of Polynomial Commitment Schemes

CS797: MTP1

Submitted in partial fulfillment of the requirements for the degree of

Master of Technology

by

Farhan Jawaid
(Roll No. 24M0801)

Under the supervision of

Prof. Sruthi Sekar

Computer Science and Engineering Department

INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

Mumbai - 400076, India

October, 2025

Acknowledgement

I would like to express my deepest gratitude to my supervisor, **Prof. Sruthi Sekar**, for her continuous guidance, insightful discussions, and encouragement throughout the course of this thesis. Her expertise and patience were invaluable in shaping my understanding of cryptographic protocols and their mathematical foundations.

I am also grateful to all the faculty members and staff of the **Department of Computer Science and Engineering, IIT Bombay**, for providing an excellent academic environment and resources that enabled me to carry out this work.

Finally, I extend my sincere thanks to my friends for their constant support, stimulating conversations, and for making my time at IIT Bombay both enjoyable and enriching.

Farhan Jawaid
24M0801
Mumbai
October, 2025

Abstract

Polynomial commitment schemes are essential cryptographic primitives that allow a prover to commit to a polynomial and later prove its evaluation at specific points without revealing the polynomial. They form the foundation of modern succinct proof systems such as zero-knowledge proofs, verifiable computation, blockchain scalability protocols.
This dissertation surveys major polynomial commitment schemes, analyzing their constructions, efficiency, and trust assumptions. The KZG scheme achieves constant-size proofs using bilinear pairings but depends on a trusted setup. Its multilinear extension generalizes the approach to multivariate polynomials, improving flexibility while increasing computational cost.
To overcome setup limitations, transparent schemes such as Pedersen commitments, Bulletproofs, and Dory are examined. Pedersen commitments rely on the discrete logarithm assumption, providing perfect hiding; Bulletproofs use logarithmic-sized inner product arguments for efficient, trustless proofs; and Dory offers a pairing-free, fully transparent protocol with succinct verification.
This study emphasizes the transition from trusted to transparent constructions and discusses open directions in efficiency, aggregation, and post-quantum secure polynomial commitments.

Contents

Chapter 1

Background

Modern cryptography relies heavily on mathematical tools that allow one party (the *prover*) to convince another (the *verifier*) of the validity of a statement without revealing any unnecessary information. Among these tools, **commitment schemes** play a foundational role in constructing zero-knowledge proofs, secure multiparty computation, and verifiable computation protocols. This chapter presents the theoretical foundations necessary to understand polynomial commitment schemes, including the basic idea of commitments, their desired properties, and the mathematical background that underlies their security.

1.1 Commitment Schemes

A **commitment scheme** enables a prover to commit to a value while keeping it hidden from others, yet ensuring that it cannot be altered once committed. It is often described as the digital equivalent of sealing a message in an envelope—one can reveal the content later, but cannot change it after sealing.
Formally, a commitment scheme consists of two algorithms:

1. **Commit**$(m; r)$**:** On input message m and randomness r, outputs commitment C.
2. **Open**(C, m, r)**:** Reveals m and verifies that C corresponds to (m, r).

A secure commitment scheme satisfies two fundamental properties:

- **Hiding:** The commitment C reveals no information about the message m. This ensures privacy until the prover chooses to open it.
- **Binding:** Once the commitment C is published, the prover cannot change m without being detected.

These properties are often achieved under computational assumptions such as the hardness of discrete logarithms or collision resistance of cryptographic hash functions. Classical examples include Pedersen commitments and hash-based commitments.

1.2 Polynomial Commitments

While standard commitments work for fixed values, many cryptographic systems require commitments to *polynomials*. A **polynomial commitment scheme** allows a prover to commit to a polynomial $f(x)$ in such a way that:

- The commitment hides the coefficients of $f(x)$.
- Later, the prover can efficiently prove that the polynomial evaluates to a claimed value y at some point z, i.e., that $f(z) = y$.

The commitment and proof must be:

- **Succinct:** The commitment and proof sizes should be much smaller than the polynomial degree.
- **Efficiently Verifiable:** The verifier should check proofs faster than recomputing $f(z)$ directly.
- **Sound:** A dishonest prover cannot convince the verifier of a false evaluation.

Polynomial commitments are the foundation for efficient **verifiable computation** and **succinct zero-knowledge proofs**, including well-known SNARKs such as Groth16, PLONK, and Marlin.

1.3 Mathematical Prerequisites

To understand how these schemes achieve efficiency and security, a few mathematical concepts are essential.

1.3.1 Finite Fields

A finite field $\mathbb{F}_p$ (where p is prime) is a set of integers $\{0, 1, \ldots, p-1\}$ equipped with addition and multiplication modulo p. Polynomials are usually defined over finite fields, and operations such as evaluation or interpolation are performed modulo p. The algebraic structure of fields provides the foundation for defining commitments and their arithmetic consistency.

1.3.2 Groups and Discrete Logarithms

Many commitment schemes rely on the hardness of the **Discrete Logarithm Problem (DLP)** in cyclic groups. Given a generator g and element $h = g^x$, it is computationally infeasible to determine x. This hardness assumption ensures the *binding* property in commitments like Pedersen and forms the security basis for protocols such as Bulletproofs and Dory.

1.3.3 Bilinear Pairings

Certain polynomial commitments, notably the KZG scheme, rely on **bilinear pairings** on elliptic curves. A pairing is a map:

$$e : G_1 \times G_2 \to G_T$$

with the property:

$$e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$$

for all integers a, b. Pairings enable efficient verification of relationships between group elements without revealing underlying exponents. They are essential for constructing constant-size proofs in the KZG commitment scheme.

1.3.4 Homomorphic Properties

Commitments often exploit homomorphic properties, allowing operations on committed values without revealing them. For example, in Pedersen commitments:

$$C(m_1 + m_2, r_1 + r_2) = C(m_1, r_1) \cdot C(m_2, r_2)$$

This linearity simplifies the construction of more complex cryptographic protocols where arithmetic operations on hidden data are required.

1.4 From Commitments to Polynomial Commitments

A standard commitment hides a single value; however, many cryptographic applications involve verifying polynomial relations. For instance, in a zkSNARK or a verifiable computation protocol, a prover may need to prove that a certain computation (represented as a polynomial equation) holds for specific inputs. Instead of committing to each coefficient individually, polynomial commitment schemes allow committing to the entire polynomial once and proving any number of evaluations succinctly.
The key design challenge is balancing the following:

- **Efficiency:** Small proofs and fast verification.
- **Security:** Strong hiding and binding guarantees.
- **Transparency:** Avoiding the need for trusted setup.

Applications

Polynomial commitments are integral to:

- **Zero-Knowledge Proofs:** Provers demonstrate that certain polynomials satisfy constraints without revealing them.
- **Blockchain Scalability:** Verifiable rollups and succinct proofs depend on polynomial evaluation checks.
- **Verifiable Computation:** Systems like zkSNARKs and STARKs require efficient polynomial evaluation proofs.

Chapter 2

KZG Polynomial Commitment Scheme

The Kate–Zaverucha–Goldberg (KZG) polynomial commitment scheme, proposed in 2010, is one of the most efficient and widely used constructions in modern cryptography. It is built upon **bilinear pairings** on elliptic curves and enables constant-size commitments and proofs, independent of the degree of the polynomial. This makes it an attractive choice for systems that require succinct verifiable computations, such as zkSNARKs and blockchain rollup proofs.

The KZG scheme allows a prover to commit to a polynomial $f(x)$ and later provide a short proof that it evaluates to a specific value y at a point z. The verifier can check this proof efficiently using a pairing equation without learning the underlying polynomial. However, this efficiency comes at the cost of requiring a **trusted setup**, a process that generates public parameters based on a secret trapdoor.

2.1 Mathematical Setup

The KZG scheme is constructed over a bilinear group setting:

$$G_1, G_2, G_T \text{ of prime order } p,$$

with generators $g_1 \in G_1$, $g_2 \in G_2$, and a bilinear pairing

$$e : G_1 \times G_2 \to G_T$$

satisfying the property:

$$e(g_1^a, g_2^b) = e(g_1, g_2)^{ab} \quad \forall a, b \in \mathbb{Z}_p.$$

A trusted setup generates public parameters for a secret value $\alpha \in \mathbb{Z}_p$:

$$\text{PP} = (g_1, g_1^{\alpha}, g_1^{\alpha^2}, \ldots, g_1^{\alpha^n}, g_2, g_2^{\alpha})$$

where n is the maximum degree of polynomials to be committed. The value α must be securely discarded after setup (the "toxic waste"), as knowledge of α breaks the binding property.

2.2 Commitment Phase

Let the prover hold a polynomial of degree n:

$$f(x) = f_0 + f_1 x + f_2 x^2 + \cdots + f_n x^n.$$

The commitment is computed as:

$$C = g_1^{f(\alpha)} = \prod_{i=0}^{n} (g_1^{\alpha^i})^{f_i}.$$

This produces a constant-size commitment in G_1, independent of n. The commitment is **hiding** if a blinding factor is added (for example, $C = g_1^{f(\alpha)} h^r$ for random r), though the base KZG scheme is typically binding but not hiding.

2.3 Evaluation and Proof Generation

To prove that the polynomial $f(x)$ evaluates to a specific value y at point z, the prover constructs a quotient polynomial:

$$Q(x) = \frac{f(x) - y}{x - z}.$$

Since $(x - z)$ divides $(f(x) - y)$, this division is valid in the field $\mathbb{F}_p$.
The proof is then:

$$\pi = g_1^{Q(\alpha)}.$$

2.4 Verification

The verifier checks that the provided y corresponds to the evaluation $f(z)$ using a pairing equation. Given (C, π, z, y), the verifier checks:

$$e(C/g_1^y, g_2) \stackrel{?}{=} e(\pi, g_2^{\alpha - z}).$$

If the equality holds, the verifier is convinced that $f(z) = y$ for the committed polynomial f.

Intuition

The correctness of the verification equation follows from the bilinearity of the pairing:

$$C = g_1^{f(\alpha)} = g_1^{Q(\alpha)(\alpha - z) + y}.$$

Rearranging gives:

$$g_1^{f(\alpha) - y} = (g_1^{Q(\alpha)})^{\alpha - z}.$$

Applying the pairing to both sides yields:

$$e(g_1^{f(\alpha)-y}, g_2) = e(g_1^{Q(\alpha)}, g_2^{\alpha - z}),$$

which is precisely the equation used in verification.

2.5 Formal Binding Analysis

The **binding** property of the KZG commitment scheme ensures that a committer cannot open a single commitment to two distinct polynomial evaluations. In this section, we present the formal argument demonstrating that any adversary capable of doing so would effectively break the **Strong Diffie–Hellman (SDH)** assumption in the underlying bilinear group.

Assume the committer produces a valid commitment $c \in G_1$ and two distinct openings (v, y) and (v', y') such that both verify correctly:

$$e(c \cdot g^{-v}, g) = e(y, g^{\tau - z}) \quad \text{and} \quad e(c \cdot g^{-v'}, g) = e(y', g^{\tau - z}),$$

where τ is the secret evaluation point used in the setup. For simplicity, write $c = g^{r_1}$, $y = g^{r_2}$, and $y' = g^{r_3}$, even though the committer may not know the discrete logarithms r_1, r_2, r_3.

By the bilinearity of e, these two equations imply:

$$g^{r_1 - v} = g^{r_2(\tau - z)} \quad \text{and} \quad g^{r_1 - v'} = g^{r_3(\tau - z)}.$$

Combining them yields:

$$g^{v - v'} = g^{(r_3 - r_2)(\tau - z)}.$$

Rearranging, we obtain:

$$\left(y' \cdot y^{-1}\right)^{1/(v-v')} = g^{1/(\tau - z)}. \tag{15.3}$$

The left-hand side can be computed efficiently by anyone who knows (y, y', v, v'), since the multiplicative inverse of $(v - v')$ modulo p can be found using the Extended Euclidean algorithm in $\tilde{O}(\log p)$ time.

Equation (15.3) implies that an adversary capable of producing two valid openings for distinct values (v, v') can compute $g^{1/(\tau - z)}$. This directly contradicts the **q-Strong Diffie–Hellman (q-SDH)** assumption, which states that it is computationally infeasible to compute such a value given only the public parameters $(g, g^{\tau}, g^{\tau^2}, \ldots, g^{\tau^q})$.

Therefore, any adversary that can produce two distinct valid openings for a single commitment can be used to construct an algorithm that breaks the SDH assumption. Hence, the KZG polynomial commitment scheme is **binding** under the q-SDH assumption.

2.6 Security Properties

The KZG scheme satisfies the following:

- **Binding:** Without knowing α, it is computationally infeasible to open a commitment to two different polynomials. This relies on the **q-Strong Bilinear Diffie–Hellman (q-SBDH)** assumption.
- **Hiding:** The basic KZG scheme is not hiding, as $C = g_1^{f(\alpha)}$ leaks information if multiple commitments are correlated. A hiding version can be constructed by adding randomness as $C = g_1^{f(\alpha)} h^r$.
- **Soundness:** A dishonest prover cannot produce a valid proof for an incorrect evaluation, assuming the binding property and hardness of the underlying pairing assumptions.

2.7 Efficiency Analysis

- **Commitment size:** Constant (one element in G_1).
- **Proof size:** Constant (one element in G_1).
- **Verification time:** Requires two pairings and a few exponentiations in G_T, making it highly efficient.
- **Prover time:** Linear in the polynomial degree n due to exponentiations.

These properties make KZG ideal for large-scale verifiable computations where succinctness is crucial. Its constant proof and commitment sizes are a major advantage over earlier schemes that scaled linearly with polynomial degree.

2.8 Limitations

Despite its efficiency, the KZG scheme suffers from several drawbacks:

1. **Trusted Setup:** The setup phase requires generating powers of a secret α. If this secret is compromised or reused, the entire system's soundness fails.

2. **Non-Transparency:** KZG is not transparent; it depends on structured reference strings (SRS) rather than public randomness.

3. **Pairing Dependence:** The scheme requires bilinear pairings, which are computationally expensive and less efficient on some elliptic curve families.

4. **Lack of Post-Quantum Security:** Security is based on pairing assumptions that are not believed to be quantum-resistant.

Chapter 3

Extended KZG: Multilinear KZG Commitments

The univariate KZG polynomial commitment scheme provides succinct, constant-size proofs for evaluations of univariate polynomials. However, many cryptographic protocols require commitments to **multivariate** or **multilinear** polynomials. For example, in succinct non-interactive proofs and verifiable computation frameworks, computations are represented as constraint systems involving polynomials over multiple variables. The original KZG scheme cannot efficiently handle these cases without committing separately to many univariate polynomials, which increases communication cost and proof size.
To address this limitation, the KZG scheme can be extended to support **multilinear** or **multivariate** commitments. The extension preserves the succinctness properties of KZG (i.e., constant-size commitments and logarithmic-size proofs) while enabling evaluation proofs for polynomials in multiple variables. This section describes the mathematical formulation and analysis of the multilinear KZG scheme.

3.1 Mathematical Setup

The setup phase is similar to that of univariate KZG but requires parameters for multivariate evaluation. Let the commitment scheme support polynomials in m variables, each variable taking values in a finite field $\mathbb{F}_p$.
Let $f(x_1, x_2, \ldots, x_m)$ be a multilinear polynomial of degree at most one in each variable:

$$f(x_1, \ldots, x_m) = \sum_{S \subseteq [m]} c_S \prod_{i \in S} x_i,$$

where $c_S \in \mathbb{F}_p$ are the coefficients.

The trusted setup chooses a secret scalar $\alpha \in \mathbb{F}_p$ and publishes:

$$\text{PP} = (g_1, g_1^{\alpha}, g_1^{\alpha^2}, \ldots, g_1^{\alpha^{2^m-1}}, g_2, g_2^{\alpha}).$$

The powers of α correspond to the 2^m possible monomials of the multilinear polynomial.

3.2 Commitment Phase

Given $f(x_1, \ldots, x_m)$, the prover computes the commitment as:

$$C = g_1^{f(\alpha_1, \alpha_2, \ldots, \alpha_m)},$$

where $(\alpha_1, \ldots, \alpha_m)$ are structured powers derived from α according to the multilinear encoding:

$$\alpha_S = \prod_{i \in S} \alpha^{2^{i-1}}.$$

Thus, for each subset $S \subseteq [m]$, the term $c_S \alpha_S$ represents the coefficient of the corresponding monomial. The commitment can then be compactly represented as:

$$C = g_1^{\sum_{S \subseteq [m]} c_S \alpha_S}.$$

3.3 Evaluation and Proof Generation

To prove that $f(z_1, z_2, \ldots, z_m) = y$ for some point $(z_1, \ldots, z_m) \in \mathbb{F}_p^m$, the prover constructs a set of quotient polynomials. The idea is to reduce the m-variate evaluation problem into smaller univariate subproblems.
For each variable x_i, define a partial evaluation polynomial:

$$Q_i(x_1, \ldots, x_m) = \frac{f(x_1, \ldots, x_m) - f(x_1, \ldots, x_{i-1}, z_i, \ldots, z_m)}{x_i - z_i}.$$

The final proof is an aggregation of these m quotient commitments:

$$\pi = (g_1^{Q_1(\alpha)}, g_1^{Q_2(\alpha)}, \ldots, g_1^{Q_m(\alpha)}).$$

Each π_i acts as a sub-proof that ensures correct evaluation in the i^{th} variable dimension.

3.4 Verification

The verifier checks that all quotient relations hold simultaneously. For each $i \in [m]$, the following pairing equation must be satisfied:

$$e(C/g_1^{f(z_1,\ldots,z_m)}, g_2) = e(\pi_i, g_2^{\alpha_i - z_i}),$$

where α_i denotes the structured evaluation of α corresponding to variable x_i.
To reduce verification cost, recursive or batched verification techniques can be used to combine all m equations into a single pairing check:

$$e\left(C/g_1^{y}, g_2\right) \stackrel{?}{=} e\left(\prod_{i=1}^{m} \pi_i^{\lambda_i}, g_2^{\sum_{i=1}^{m} \lambda_i(\alpha_i - z_i)}\right),$$

where λ_i are random challenge scalars used for aggregation. This optimization reduces the verification time from $O(m)$ to $O(1)$ pairing checks.

Intuition

Intuitively, the multilinear KZG scheme extends the univariate commitment by encoding the multivariate polynomial into powers of α that represent combinations of variable indices. When committing, each monomial term of the polynomial corresponds to a unique exponent of α. During verification, each variable's evaluation is checked via an independent quotient relation, ensuring that the polynomial's value at the claimed point is consistent with the commitment.
The use of multilinear encoding allows maintaining a compact commitment and logarithmic proof size while supporting exponential combinations of variables.

3.5 Security Properties

The multilinear KZG scheme inherits its security from the univariate version, under the same **q-Strong Diffie–Hellman (q-SDH)** assumption. Formally:

- **Binding:** Without knowledge of α, it is infeasible to produce two valid openings for different polynomial evaluations.
- **Soundness:** Producing a false proof of evaluation implies the ability to compute $g^{1/(\alpha - z)}$, thereby violating the SDH assumption.
- **Hiding:** As in the base KZG, a perfectly hiding variant can be obtained by including a blinding term h^r in the commitment.

3.6 Efficiency Analysis

- **Commitment size:** Constant (one element in G_1).
- **Proof size:** $O(m)$ group elements (logarithmic when aggregation is used).
- **Verifier time:** Linear in m, or constant with batch pairing.
- **Prover time:** Grows linearly with 2^m due to the number of monomials in a multilinear polynomial.

While the proof remains succinct, the computation overhead increases exponentially with the number of variables, which limits scalability to moderate values of m.

3.7 Limitations

Despite being a significant extension, the multilinear KZG scheme still inherits certain drawbacks:

1. **Trusted Setup:** The scheme requires a structured reference string with powers of α, generated via a secure setup ceremony.
2. **Exponential Complexity:** The number of monomials in a multilinear polynomial grows as 2^m, increasing prover computation.
3. **Pairing Dependence:** The scheme relies on pairing-friendly elliptic curves, which are not quantum-secure.

Chapter 4

Problems with Trusted Setup

Many efficient polynomial commitment schemes, including KZG and its multilinear extensions, rely on a **trusted setup** phase to generate public parameters. This setup involves a secret scalar, typically denoted α, which is used to construct a structured reference string (SRS):

$$\text{SRS} = (g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^n}).$$

After setup, α must be permanently destroyed. The secrecy of this trapdoor parameter is critical to the scheme's soundness and binding properties. While the trusted setup enables succinct proofs and efficient verification, it introduces significant trust, transparency, and security challenges.

4.1 The Toxic Waste Problem

The secret value α generated during setup is often referred to as **toxic waste**. If an adversary learns this value, the entire system's integrity collapses. The adversary could:

- Forge valid proofs for arbitrary polynomial evaluations.
- Break the binding property by opening one commitment to multiple values.
- Generate commitments and proofs that verify without knowing the underlying polynomial.

Since it is impossible to publicly verify whether α has been destroyed, all users must rely on trust in the entity or participants that performed the setup. This reliance introduces a non-cryptographic assumption into otherwise mathematically sound protocols.

4.2 Limitations of Multi-Party Setup

To mitigate this single point of trust, many systems use a **multi-party computation (MPC)** setup, where several participants collectively generate the SRS. Each participant contributes secret randomness, and as long as one participant deletes their portion honestly, the parameters remain secure.
Although MPC improves trust distribution, it introduces several limitations:

- The setup process is complex, time-consuming, and requires global coordination.
- Any modification to the circuit size or polynomial degree demands a new ceremony.
- Verifying that all participants behaved correctly is impractical.

Thus, even multi-party ceremonies do not provide true transparency, and practical scalability remains an issue.

4.3 Motivation for Transparent Alternatives

The challenges associated with trusted setup—centralized trust, lack of verifiability, and inflexibility—motivate the design of **transparent** commitment schemes. In a transparent setup, all public parameters are derived deterministically from publicly known randomness, such as hash functions or standard cryptographic assumptions.
Transparent polynomial commitments eliminate the need for secret trapdoor values, ensuring that:

- Security is verifiable and independent of any trusted entity.
- Setup can be reproduced and audited by anyone.
- The system aligns with decentralization and long-term trust goals.

While transparent constructions such as Pedersen commitments, Bulletproofs, and Dory may have slightly larger proofs or higher computational costs, they provide stronger security and trust guarantees. The following chapters explore these schemes in detail, highlighting how transparency addresses the fundamental weaknesses of trusted setup–based systems.

Chapter 5

Pedersen Commitment Scheme

The **Pedersen commitment scheme**, introduced by Torben Pedersen in 1991, is one of the simplest and most elegant commitment constructions in modern cryptography. It provides a **perfectly hiding** and **computationally binding** commitment based solely on the **discrete logarithm assumption**.
Unlike the KZG scheme, Pedersen commitments require no trusted setup or bilinear pairings. They rely only on publicly known group parameters and standard hardness assumptions, making them a fully **transparent** cryptographic primitive. Pedersen commitments form the foundation for several advanced transparent protocols, including Bulletproofs and Dory, which extend its properties to polynomial and vector commitments.

5.1 Mathematical Setup

Let G be a cyclic group of prime order p in which the discrete logarithm problem (DLP) is hard. Let $g, h \in G$ be two generators such that $\log_g(h)$ is unknown to all parties. These generators can be derived deterministically from a public random seed to ensure transparency.
The public parameters are:

$$\text{PP} = (G, p, g, h).$$

5.2 Commitment Phase

To commit to a message $m \in \mathbb{Z}_p$, the committer selects a random blinding factor $r \in \mathbb{Z}_p$ and computes:

$$C = g^m h^r.$$

The value C serves as the commitment to m, while (m, r) constitutes the opening information.

5.3 Verification (Opening Phase)

To open the commitment, the committer reveals (m, r) to the verifier. The verifier checks the validity of the opening by recomputing:

$$C \stackrel{?}{=} g^m h^r.$$

If the equality holds, the verifier accepts the opening as valid. Otherwise, the verifier rejects it.

Intuition

The commitment hides the message m because the random exponent r completely masks its contribution in the group element C. Even if the verifier knows g, h, and C, the value of m remains indistinguishable since for every possible m, there exists a corresponding r that produces the same C. This property is what makes the scheme **perfectly hiding**. On the other hand, the commitment is **binding** because changing the committed message would require solving the discrete logarithm problem. If a committer could produce (m, r) and (m', r') such that both open to the same C, then:

$$g^m h^r = g^{m'} h^{r'} \implies g^{m-m'} = h^{r'-r}.$$

Taking logarithms relative to g, we get:

$$m - m' = (r' - r) \log_g(h),$$

which would reveal $\log_g(h)$, violating the discrete logarithm assumption. Hence, finding two valid openings is computationally infeasible.

5.4 Security Properties

- **Perfect Hiding:** The commitment $C = g^m h^r$ is uniformly distributed in G due to the random blinding factor r. Thus, C leaks no information about m.
- **Computational Binding:** It is computationally infeasible to find two pairs (m, r) and (m', r') such that $g^m h^r = g^{m'} h^{r'}$, unless the discrete logarithm $\log_g(h)$ is known.
- **Transparency:** The parameters (g, h) can be derived deterministically from public randomness, requiring no trusted setup. This eliminates the risk of toxic waste or secret trapdoors.

5.5 Homomorphic Property

Pedersen commitments are additively homomorphic. Given commitments $C_1 = g^{m_1}h^{r_1}$ and $C_2 = g^{m_2}h^{r_2}$, their product is:

$$C_1C_2 = g^{m_1+m_2}h^{r_1+r_2}.$$

Thus,

$$\text{Commit}(m_1 + m_2, r_1 + r_2) = \text{Commit}(m_1, r_1) \cdot \text{Commit}(m_2, r_2).$$

This property enables efficient operations on committed values without revealing them and forms the mathematical basis for many zero-knowledge and vector commitment schemes.

5.6 Efficiency Analysis

- **Commitment size:** One group element (constant size).
- **Proof size:** Same as opening size (m, r) — both elements in $\mathbb{Z}_p$.
- **Computation:** Requires two exponentiations for commitment and one equality check for verification.
- **Transparency:** Fully transparent — no secret parameters required.

5.7 Limitations

Although Pedersen commitments are elegant and efficient, they have limitations when applied to polynomial or vector settings:

1. They cannot directly prove polynomial evaluations or relations succinctly.
2. Proof size and verification time grow linearly with the number of committed values.
3. They provide no inherent mechanism for logarithmic-size proofs or aggregated openings.

These limitations motivate the development of more advanced transparent schemes such as **Bulletproofs** and **Dory**, which build upon Pedersen's core ideas to achieve succinctness and efficient verifiability.

Chapter 6

Bulletproofs

The **Bulletproofs** protocol provides a transparent polynomial commitment scheme in which the commitment size is constant, and the proof length in the evaluation phase grows only logarithmically with the number of coefficients of the committed polynomial. The verifier's runtime, however, remains linear in the number of coefficients. Compared to earlier schemes, Bulletproofs strictly improve the communication complexity of the proof—from linear to logarithmic—while maintaining similar computational efficiency for the prover and verifier. The construction builds on generalized Pedersen vector commitments and introduces an elegant recursive reduction technique based on **inner product arguments**.

6.1 Knowledge of a Vector Commitment Opening

To understand the main ideas behind Bulletproofs, we begin with a simplified protocol showing how a prover can demonstrate knowledge of a vector $u \in \mathbb{F}_p^n$ that opens a Pedersen vector commitment. Let G be a cyclic group of prime order p written additively, and let $\mathbf{g} = (g_1, g_2, \ldots, g_n)$ denote a public vector of generators. A generalized Pedersen commitment to $u = (u_1, \ldots, u_n)$ is:

$$\mathrm{Com}(u) = \sum_{i=1}^{n} u_i g_i = \langle u, \mathbf{g} \rangle.$$

The prover wishes to convince the verifier that it knows a vector u such that:

$$\langle u, \mathbf{g} \rangle = c_u,$$

without revealing u.

If $n = 1$, the problem reduces to proving knowledge of a discrete logarithm: the prover simply sends u such that $ug_1 = c_u$. For larger n, the protocol proceeds recursively in

$\log_2 n$ rounds, halving the vector length in each step until the base case $n = 1$ is reached.

6.2 Recursive Reduction of Vector Length

Suppose u and $\mathbf{g}$ are each split into two halves:

$$u = (u_L, u_R), \qquad \mathbf{g} = (\mathbf{g}_L, \mathbf{g}_R),$$

so that:

$$\langle u, \mathbf{g} \rangle = \langle u_L, \mathbf{g}_L \rangle + \langle u_R, \mathbf{g}_R \rangle.$$

The goal of each recursive step is to reduce the claim "the prover knows u such that $\langle u, \mathbf{g} \rangle = c_u$" to an equivalent claim about shorter vectors. To do this, the verifier chooses a random challenge $\alpha \in \mathbb{F}_p^*$ and defines:

$$u' = \alpha u_L + \alpha^{-1} u_R, \qquad \mathbf{g}' = \alpha^{-1} \mathbf{g}_L + \alpha \mathbf{g}_R.$$

The verifier can compute $\mathbf{g}'$ independently since it knows $\mathbf{g}_L$ and $\mathbf{g}_R$, but it does not know u'.
Ideally, one might hope that $\langle u', \mathbf{g}' \rangle = \langle u, \mathbf{g} \rangle$, but this equality does not hold directly because of *cross terms*. A direct computation gives:

$$\langle u', \mathbf{g}' \rangle = \langle u, \mathbf{g} \rangle + \alpha^2 \langle u_L, \mathbf{g}_R \rangle + \alpha^{-2} \langle u_R, \mathbf{g}_L \rangle.$$

The verifier cannot compute these cross terms, as they depend on the secret vector u. To resolve this, the prover sends two additional values:

$$v_L = \langle u_L, \mathbf{g}_R \rangle, \quad v_R = \langle u_R, \mathbf{g}_L \rangle,$$

before learning α. The verifier then defines a new commitment:

$$c'_u = c_u + \alpha^2 v_L + \alpha^{-2} v_R.$$

If v_L and v_R are correctly computed, this ensures that:

$$\langle u', \mathbf{g}' \rangle = c'_u.$$

The claim "the prover knows u such that $\langle u, \mathbf{g} \rangle = c_u$" is thus reduced to a similar claim involving u' and $\mathbf{g}'$, each of half the original length. This process repeats recursively until $n = 1$.

6.3 Protocol Description

Formally, the protocol proceeds as follows:

1. If $n = 1$, the prover sends u to the verifier, who checks $ug_1 = c_u$.
2. Otherwise, the prover computes and sends $v_L = \langle u_L, \mathbf{g}_R \rangle$ and $v_R = \langle u_R, \mathbf{g}_L \rangle$.
3. The verifier samples a random $\alpha \in \mathbb{F}_p^*$ and sends it to the prover.
4. Both parties compute:

$$u' = \alpha u_L + \alpha^{-1} u_R, \quad \mathbf{g}' = \alpha^{-1} \mathbf{g}_L + \alpha \mathbf{g}_R, \quad c'_u = c_u + \alpha^2 v_L + \alpha^{-2} v_R.$$

5. The protocol recurses on the new instance $(u', \mathbf{g}', c'_u)$ of length $n/2$.

The recursion continues for $\log_2 n$ rounds, with the prover sending exactly two group elements (v_L, v_R) in each round. The verifier performs corresponding checks using the updated commitments.

6.4 Efficiency Analysis

The Bulletproofs inner product argument achieves:

- **Proof size:** $2 \log_2 n$ group elements.
- **Prover computation:** $O(n)$ group exponentiations (or scalar multiplications in additive notation).
- **Verifier computation:** $O(n)$ total group operations, dominated by generator updates across rounds.

Thus, the protocol achieves logarithmic communication while maintaining linear verification complexity. This marks a substantial improvement in proof size compared to previous polynomial commitment schemes.

Properties

The scheme is a public-coin zero-knowledge argument of knowledge under the hardness of the discrete logarithm problem:

- **Knowledge Soundness:** A dishonest prover cannot produce a valid transcript without knowing u satisfying $\langle u, \mathbf{g} \rangle = c_u$.

- **Computational Binding:** Finding two valid openings to the same commitment would require solving the discrete logarithm problem.
- **Zero-Knowledge:** If a blinding factor (omitted here for simplicity) is used, the commitment hides u completely.

6.5 Discussion and Comparison

Bulletproofs offer constant-size commitments and logarithmic-size proofs, making them more communication-efficient than prior schemes with linear or square-root proof sizes. However, their verifier runtime is linear in the number of coefficients, unlike other protocols that achieve sublinear verification at the cost of larger proofs. The balance of succinct communication and reasonable verification complexity makes Bulletproofs a cornerstone of transparent polynomial commitment design.

Chapter 7

Dory Protocol

The **Dory protocol** is a transparent polynomial commitment scheme that achieves logarithmic communication and verification costs without relying on a trusted setup. Unlike pairing-based schemes such as KZG, Dory does not require any secret trapdoor parameters, and unlike Bulletproofs, it achieves both logarithmic proof size and logarithmic verifier time.
Dory builds on several algebraic building blocks including the AFGHO commitment and inner-pairing-product (IPP) commitments, combining them through recursive inner product arguments. It achieves transparency by replacing trusted setup assumptions with public pre-processing procedures that introduce no toxic waste and can be verified by any participant.

7.1 Vectors Commitments via Inner-Pairing-Products

Let $\mathbb{F}_p$ be a finite field of prime order p, and let (G_1, G_2, G_T) be cyclic groups of order p equipped with a bilinear pairing $e : G_1 \times G_2 \to G_T$. For a vector $w = (w_1, \ldots, w_n) \in G_1^n$ and a public vector $g = (g_1, \ldots, g_n) \in G_2^n$, the **inner-pairing-product commitment** (IPPCom) is defined as:

$$\text{IPPCom}(w) = \sum_{i=1}^{n} e(w_i, g_i) = \langle w, g \rangle.$$

This commitment binds a vector of group elements into a single element of G_T. The construction can be viewed as a Pedersen-style commitment extended to vectors of group elements rather than field elements. Under the Decisional Diffie–Hellman (DDH) assumption in G_1, IPPCom is **computationally binding**.
For vectors of field elements $v \in \mathbb{F}_p^n$, we can define $w(v) = (v_1 h, \ldots, v_n h)$ for some public

generator $h \in G_1$, and commit as:

$$\text{IPPCom}(v) = \sum_{i=1}^{n} e(v_i h, g_i) = e(h, \sum_{i=1}^{n} v_i g_i).$$

This forms the basis of Dory's polynomial commitments.

7.2 Motivation and Improvements over Bulletproofs

Bulletproofs achieve logarithmic proof size but require linear verification time because the verifier must update its commitment key in each round. Dory introduces a transparent **pre-processing phase** that allows these updates to be handled homomorphically, reducing verifier time to logarithmic.
The pre-processing consists of recursively generating commitments to halves of the generator vector $g = (g_1, \ldots, g_n)$. This produces a sequence of commitments $\Delta_L^{(i)}, \Delta_R^{(i)}$ under commitment keys $\Gamma^{(i)}$ for $i = 1, \ldots, \log n$, allowing the verifier to later combine commitments using public randomness rather than recomputing generators. Importantly, this pre-processing introduces no secret randomness and produces no trapdoor.

7.3 Knowledge-of-Opening Protocol

The core of Dory is a recursive proof that the prover knows a vector u opening an inner-pairing-product commitment $c_u = \text{IPPCom}(u) = \langle u, g \rangle$. The recursion proceeds similarly to Bulletproofs but leverages the pre-processed commitments to maintain logarithmic verifier cost.

Round 1: Initialization

The prover begins with $c_u^{(0)} = \langle u^{(0)}, g^{(0)} \rangle$. It sends two commitments v_L, v_R to cross terms $\langle u_L, g_R \rangle$ and $\langle u_R, g_L \rangle$. The verifier samples a random challenge $\alpha_1 \in \mathbb{F}_p^*$ and computes:

$$c_u^{(1)} = c_u^{(0)} + \alpha_1^2 v_L + \alpha_1^{-2} v_R.$$

While the verifier does not know $g^{(0)}$, it holds commitments $\Delta_L^{(1)} = \langle g_L, \Gamma^{(1)} \rangle$ and $\Delta_R^{(1)} = \langle g_R, \Gamma^{(1)} \rangle$ from pre-processing, which it combines homomorphically:

$$c_g^{(1)} = \alpha_1^{-1} \Delta_L^{(1)} + \alpha_1 \Delta_R^{(1)}.$$

The protocol thus reduces to proving knowledge of $u^{(1)}, g^{(1)}$ satisfying:

$$c_u^{(1)} = \langle u^{(1)}, g^{(1)} \rangle, \quad c_g^{(1)} = \langle g^{(1)}, \Gamma^{(1)} \rangle.$$

Subsequent Rounds

In each subsequent round i, the verifier and prover execute the same structure recursively:

- The prover sends commitments to left and right halves of the current vectors.
- The verifier samples a random challenge α_i.
- Both parties fold the vectors as:

$$u^{(i)} = \alpha_i u_L^{(i-1)} + \alpha_i^{-1} u_R^{(i-1)}, \quad g^{(i)} = \alpha_i^{-1} g_L^{(i-1)} + \alpha_i g_R^{(i-1)}.$$

- The verifier uses pre-processing commitments to compute commitments to the updated vectors homomorphically.

After $\log n$ recursive steps, the vectors have length 1, and the prover reveals the final scalars $u^{(\log n)}$ and $g^{(\log n)}$ for direct verification.

7.4 Extending to Polynomial Commitments

To commit to a polynomial $q(X) = \sum_{i=0}^{n-1} a_i X^i$, the prover commits to its coefficient vector $a = (a_0, a_1, \ldots, a_{n-1})$ as:

$$c_q = \text{IPPCom}(w(a)) = \sum_{i=1}^{n} e(a_i h, g_i).$$

To prove that $q(r) = v$ for some $r \in \mathbb{F}_p$, the prover and verifier execute the above recursive argument on two simultaneous claims:

$$c_q = \langle w(a), g \rangle, \quad \text{and} \quad \langle a, y \rangle = v,$$

where $y = (1, r, r^2, \ldots, r^{n-1})$. A tensor-structured computation of $y^{(\log n)}$ allows the verifier to perform all necessary checks in $O(\log n)$ time.

7.5 Efficiency Analysis

Dory achieves the following asymptotic costs:

- **Commitment size:** Constant (one group element).
- **Proof size:** $O(\log n)$ group elements.
- **Verifier time:** $O(\log n)$ scalar multiplications in G_T.

- **Prover time:** $O(n)$ operations, dominated by polynomial evaluation and commitment computation.
- **Pre-processing:** $O(\sqrt{n})$ time (in the optimized variant), fully transparent and reusable.

7.6 Security Properties

- **Completeness:** Honest provers always produce proofs that satisfy the verifier's checks.
- **Computational Binding:** Based on the DDH or SXDH assumptions, no adversary can open a commitment to two different values.
- **Zero-Knowledge (Optional):** A blinding factor can be included in commitments to hide the committed polynomial completely.
- **Transparency:** No trusted setup is required; all parameters are derived from public randomness.

7.7 Discussion and Comparison

Dory achieves a key milestone in the evolution of polynomial commitments: it matches the succinctness of Bulletproofs while improving verifier efficiency to logarithmic time. Its use of inner-pairing-product commitments and transparent preprocessing distinguishes it from earlier pairing-based or trusted-setup constructions. Although proofs are larger in practice due to constants introduced by pairing operations, Dory offers the strongest combination of transparency, succinctness, and soundness among current transparent schemes.

Chapter 8

Comparison of Polynomial Commitment Schemes

This chapter presents a comparative analysis of the polynomial commitment schemes discussed in this dissertation—**KZG**, **Multilinear KZG**, **Pedersen**, **Bulletproofs**, and **Dory**. Each protocol offers distinct trade-offs in proof size, verification efficiency, transparency, and underlying cryptographic assumptions. The goal of this section is to highlight the evolution of these schemes from trusted-setup, pairing-based constructions to fully transparent, pairing-free protocols.

8.1 Comparison Criteria

The comparison focuses on five key metrics that determine the practical and theoretical efficiency of commitment schemes:

- **Proof Size:** The asymptotic and qualitative measure of communication cost between the prover and verifier.
- **Prover Time:** The computational effort required to generate the proof.
- **Verification Time:** The computational effort required by the verifier to check correctness.
- **Trusted Setup:** Whether the scheme depends on a trusted setup or is fully transparent.
- **Assumptions:** The underlying hardness assumptions providing security (e.g., pairing-based or discrete logarithm).

8.2 Comparative Discussion

The **KZG scheme** achieves constant-size commitments and proofs with extremely fast verification, but it relies on a structured reference string generated via a trusted setup. Its **multilinear extension** generalizes to multivariate polynomials but incurs higher prover cost and linear verification complexity.

The **Pedersen commitment** is the simplest transparent construction, offering perfect hiding and computational binding under the discrete logarithm assumption, but with linear proof and verification costs. **Bulletproofs** improve communication efficiency by reducing the proof size to logarithmic in the number of coefficients through recursive inner product arguments. However, their verifier runtime remains linear.

Finally, the **Dory protocol** achieves both logarithmic proof size and logarithmic verification time through homomorphic pre-processing and pairing-based inner-product commitments. While its proofs are slightly larger in concrete size due to pairing overheads, it provides the strongest balance between efficiency, succinctness, and transparency.

8.3 Comparison Table

Scheme	Proof Size	Prover Time	Verification Time	Trusted Setup	Assumptions
KZG	Constant ($O(1)$)	$O(n)$	$O(1)$	Required	Pairing-based (q-SDH)
Multilinear KZG	Linear ($O(\ell)$)	$O(2^\ell)$	$O(\ell)$	Required	Pairing-based (q-SDH)
Pedersen	Linear ($O(n)$)	$O(n)$	$O(n)$	None	Discrete Logarithm (DL)
Bulletproofs	Logarithmic ($O(\log n)$)	$O(n)$	$O(n)$	None	Discrete Logarithm (DL)
Dory	Logarithmic ($O(\log n)$)	$O(n)$	$O(\log n)$	None	SXDH / DDH

Table 8.1: Comparison of polynomial commitment schemes in terms of asymptotic efficiency and security assumptions.

8.4 Analysis

The comparison illustrates a clear progression in design philosophy:

- **Succinctness:** KZG and Dory achieve constant or logarithmic proof sizes, providing near-optimal communication efficiency.
- **Transparency:** Pedersen, Bulletproofs, and Dory eliminate the need for trusted setup, enhancing auditability and decentralization.
- **Verifier Efficiency:** Dory achieves the most efficient transparent verification with $O(\log n)$ cost, improving over Bulletproofs' linear-time verification.

- **Security Assumptions:** Trusted-setup schemes rely on pairing-based assumptions (e.g., q-SDH), while transparent schemes rely on the more standard discrete logarithm or DDH assumptions.

Chapter 9

Conclusion

9.1 Summary

This dissertation presented a comprehensive survey and comparative study of **polynomial commitment schemes**, a fundamental primitive in modern cryptography that enables succinct and verifiable proofs of polynomial evaluations. Through the analysis of five major constructions—**KZG**, **Multilinear KZG**, **Pedersen**, **Bulletproofs**, and **Dory**—we explored the trade-offs between efficiency, transparency, and underlying cryptographic assumptions.

The **KZG** commitment scheme demonstrated the theoretical efficiency of pairing-based systems, achieving constant-size proofs and constant-time verification. However, its reliance on a **trusted setup** and potential vulnerability due to toxic waste limit its practical deployment in decentralized environments. The **Multilinear KZG** extension generalized the scheme to multivariate settings, offering greater expressiveness but at increased computational cost.

In contrast, **Pedersen commitments** introduced the concept of **transparent commitments**, eliminating the need for a trusted setup and relying solely on the discrete logarithm assumption. While they offer perfect hiding and simplicity, their linear proof and verification sizes make them unsuitable for large-scale systems requiring succinctness.

Building on Pedersen's foundation, **Bulletproofs** achieved logarithmic proof size through recursive inner product arguments. This marked a significant milestone in the design of transparent and succinct proof systems, though the verifier's time remained linear.

Finally, the **Dory protocol** refined this line of research, combining transparent setup with logarithmic verification costs. By introducing homomorphic preprocessing and inner-pairing-product commitments, Dory achieves near-optimal efficiency without sacrificing transparency.

9.2 Comparative Insights

The comparative analysis highlighted a clear evolution in polynomial commitment design:

- Early schemes like KZG optimized for succinctness but required trust.
- Later constructions, such as Pedersen and Bulletproofs, prioritized transparency and auditability.
- Dory successfully balanced both, achieving transparency and near-optimal verification efficiency.

This progression reflects a broader shift in cryptographic protocol design—moving away from trusted, specialized assumptions toward general-purpose, transparent, and publicly verifiable systems.

9.3 Future Work

Although existing schemes like Dory mark a significant advancement, several open directions remain:

- **Post-quantum commitments:** Current constructions rely on algebraic assumptions (e.g., discrete logarithm, pairings) that are not secure against quantum adversaries. Exploring lattice-based or hash-based alternatives is an important direction.
- **Aggregation and batching:** Further improvements in proof aggregation could reduce communication costs across multiple commitments.
- **Practical implementations:** Concrete optimizations, hardware acceleration, and integration into zero-knowledge systems such as zkSNARKs or zkSTARKs would enhance usability.
- **Unified frameworks:** Designing a generic commitment framework that balances succinctness, transparency, and post-quantum security remains an open challenge.

Closing Remarks

Polynomial commitment schemes continue to play a central role in verifiable computation, zero-knowledge proofs, and scalable cryptographic systems. The evolution from trusted, pairing-based designs to transparent and logarithmic-time protocols like Dory demonstrates the rapid progress in achieving both security and efficiency. As cryptography moves toward a future of decentralized and quantum-resistant applications, polynomial commitments will remain a foundational tool enabling trustless, efficient, and privacy-preserving computation.

References

1. J. Thaler, *Proofs, Arguments, and Zero-Knowledge*. Cambridge University Press, 2022. Available online at: `https://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.pdf`.

2. A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-Size Commitments to Polynomials and Their Applications," in *Advances in Cryptology – ASIACRYPT 2010*, Lecture Notes in Computer Science, vol. 6477, Springer, 2010, pp. 177–194. doi: 10.1007/978-3-642-17373-8_11.

3. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short Proofs for Confidential Transactions and More," in *2018 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2018, pp. 315–334. doi: 10.1109/SP.2018.00020.

4. B. Bünz, B. Drake, B. Fisch, and A. Gabizon, "Dory: Efficient, Transparent Arguments for Generalized Inner Products and Polynomial Commitments," in *Advances in Cryptology – CRYPTO 2021*, Lecture Notes in Computer Science, vol. 12825, Springer, 2021, pp. 693–722. doi: 10.1007/978-3-030-84259-8_24.

5. S. Zhang, et al., "Transparent Polynomial Commitments and Applications to Zero-Knowledge Proofs," *Cryptology ePrint Archive*, Report 2018/133, 2018. Available: `https://eprint.iacr.org/2018/133`.

6. D. Boneh, R. Gennaro, and S. Halevi, "Efficient Non-Interactive Proof Systems for Bilinear Groups," in *Advances in Cryptology – EUROCRYPT 2019*, Lecture Notes in Computer Science, vol. 11477, Springer, 2019, pp. 123–151.