
Politecnico di Torino

Politecnico di Torino
Corso di Laurea Magistrale in Ingegnieria Matematica

# *Post-Quantum Anonymous Credentials and ZK-STARKs*

Candidata:
Bianca Rampazzo

Relatore:
Antonio Josè di Scala

Relatori aziendali:
**LINKS Foundation**
Vesco Andrea Guido Antonio
Alessandro Pino

Anno Accademico 2024-25

# Contents

# Introduction

In the modern digital landscape, where individuals continuously interact with online services and platforms, the protection of user privacy and the integrity of authentication mechanisms have become critical concerns. As digital identity becomes increasingly central to accessing services, from financial systems to healthcare and social media, ensuring that users can prove who they are without unnecessarily revealing sensitive information is essential.

Traditional authentication mechanisms, such as passwords and centralized identity providers, present numerous risks: data breaches, single points of failure, and pervasive user tracking. In response to these challenges, the cryptographic community has developed **privacy-preserving authentication protocols**, including **anonymous credential systems**, which allow users to authenticate while maintaining control over what information is disclosed. Such systems enable selective disclosure of attributes (e.g., proving over-18 status without revealing a birthdate) and offer **unlinkability**, meaning that multiple authentications by the same user cannot be trivially correlated. These properties are particularly valuable in contexts such as digital cash, access control, and anonymous access to web resources.

However, the emergence of quantum computing poses new challenges. Many cryptographic primitives underlying current privacy-preserving systems, such as discrete logarithms and pairings, may become insecure in the presence of quantum adversaries. This has triggered a growing interest in **post-quantum cryptography**, and, in particular, in the development of post-quantum anonymous credential schemes and zero-knowledge proofs that remain secure even in a quantum world.

This thesis is situated within this context, exploring how advanced cryptographic constructions, especially **post-quantum zero-knowledge proofs** such as zk-STARKs, can be leveraged to build quantum-resistant privacy preserving authentication systems.

**Contents:**

1. Chapter I introduces the concept of anonymous credentials, beginning with traditional identity models and progressing to attribute-based credentials (ABCs). It describes the BBS+ signature scheme and briefly introduces zk-SNARKs as tools for enabling selective disclosure.

2. Chapter II introduces the theoretical background of zero-knowledge proofs, covering key concepts such as hash functions, complexity classes, Sigma-protocols, and the formal definitions of zero-knowledge and proof-of-knowledge.

3. Chapter III focuses on zk-SNARKs, presenting the algebraic structures behind them, such as arithmetic circuits, R1CS, and polynomial commitments, and detailing core protocols like the Sum-Check and GKR protocols. A dedicated section also introduces the FRI protocol as a low-degree testing mechanism.

4. Chapter IV turns to zk-STARKs, explaining the notions of computational integrity, IOPs (Interactive Oracle Proofs), and arithmetization models such as AIR. A worked example and full protocol description are provided to illustrate the zk-STARK construction.

5. Lastly, Chapter V presents a post-quantum Privacy Pass based on anonymous credentials and zk-STARKs. It includes a STARK-friendly signature variant (zkDilithium20), a credential construction, a rate-limited protocol, implementation details, and performance benchmarks.

Chapter I

# Anonymous credentials

Anonymous credentials were invented to allow users to obtain credentials and show some properties without revealing any additional information or allowing tracking. It's a way to show credentials without compromising privacy: for example, a user might want to convince a verifier that it's over some age, without revealing its date of birth.

Currently, known implementations of this technology include Idemix by IBM and UProve by Microsoft.

To appreciate the significance of anonymous credentials, it is essential to first understand how traditional credential systems work.

## I.1 Traditional credentials

A credential is a set of data that proves a user's identity or provides information about their rights or access level. Examples include passwords, ID cards, certificates, and digital tokens. Credentials are essential for a secure access control because they help verify that a user or system is who or what they claim to be (authentication) and confirm the user's permission to access certain resources (authorization).

A trusted entity, like a government or an organization, **issues** the credential directly tied to the user's identity. For example, a government might issue a driver's license with the user's name, photo, and date of birth. In digital contexts, a website might issue a password or a digital certificate linked to a user's account.

When accessing a service or system, the user presents the credential to **verify** their identity. For example, entering a username and password to log into an account, or showing an ID card to enter a restricted area. The credential is often verified by matching it against stored information (like a password hash in a database or a database of authorized ID card holders).

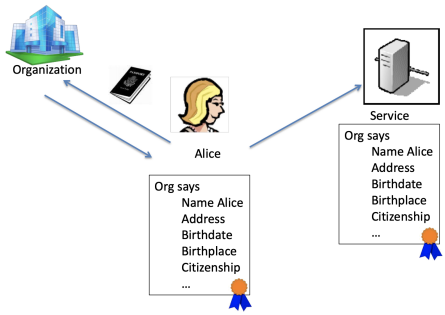



Figure I.1: Functioning of Traditional Credentials (Source: Melissa Chase)

With traditional credentials, the user is typically required to share their identifying information in full with the verifier; such as their full name, date of birth, or account details. This approach results in what is commonly referred to as **full disclosure of identity**, meaning that each time the credential is presented, the verifier learns the complete identity of the user. This process often leaves a **traceable digital footprint**, as verifiers can associate multiple interactions with the same individual. Consequently, traditional credentials can expose users to tracking across services and transactions, raising serious privacy concerns. In many cases, verifiers may store, share, or even monetize this personal information, further undermining user privacy.

## I.2 Attribute-Based Credentials (ABC)

The two primary properties that should aim to be achieved are:

- Selective disclosure: the holder can choose to disclose only parts of the credential, rather than the entire credential
- Unlinkability: the verifier would not be able to link previous holder's interactions with him

In some cases, even the above two properties might not be sufficient, for example when the holder wants to reveal part of the data: if a holder wants to prove he is over 18 or above 65 for availing a service, he doesn't need to show his date of birth; just being able to prove those conditions, called **predicates** is sufficient for the verifier. Similarly, an investor should be able to convince a verifier that the total value of his assets is greater than some amount without revealing the actual values of his assets.
One of the solutions to solve the above problem is called anonymous credentials, sometimes called **Attribute-Based Credentials (ABC)**. Instead of considering the credential data as arbitrary bytes which are then signed by the issuer, anonymous credentials add "structure" to the credential. Each claim in the credential is treated as an attribute and then these attributes are signed in a specific way by the issuer to create a signature. These attributes and the signature now form anonymous credentials.
Because of the "structure" in this credential, the holder can prove to a verifier, using zero-knowledge proof, that he has this credential from the issuer without revealing any of the attributes. To be precise, the holder can disclose these attributes exist and have been signed by an official issuer, without disclosing the attributes themselves or the signature.

A trusted authority **issues** an anonymous credential to the user, but instead of binding it directly to the user's real identity, it links the credential to specific attributes or entitlements. For instance, an anonymous credential might show that a user is over 18 or has a certain qualification without linking back to their actual name or ID. The credential is encrypted and secured cryptographically, making it tamper-proof and difficult to forge.

When presenting an anonymous credential, users can **selectively disclose** only the information that's necessary for the interaction. Cryptographic techniques, like Zero-Knowledge Proofs (ZKP), allow users to prove their credential's validity without revealing the credential itself.

The verifier can confirm that the credential is valid without knowing who the user is (**Verification Without Identification**). This process protects user anonymity by preventing linkages across different transactions, so the verifier cannot track users or create profiles based on repeated interactions (**unlinkability**). As a result, privacy is enhanced, since users' identities remain protected, allowing anonymous access to services. Additionally, regulatory compliance is improved, especially with privacy laws like GDPR, as less personal data is collected or shared.

### I.2.1 BBS+ signature

To achieve both selective disclosure and unlinkability, a suitable approach is to use the BBS+ signature scheme [CDL16]. One of its key properties is the ability to prove, in zero knowledge, that specific attributes shared across multiple credentials are equal, without disclosing the attributes themselves.

For instance, if both a driving license and a passport are issued with BBS+ signatures, it is possible to demonstrate that the Social Security Number (SSN) attribute in both credentials is identical, without revealing the SSN. This capability enables the secure linking of multiple credentials, which is particularly useful when it is necessary to prove that all credentials pertain to the same subject or identifier.

## Introduction to zk-SNARKs

As previously said, some common requirements with anonymous credentials are being able to prove conditions or predicates about the attributes, like proving that total income from a bank statement credential is less than the limit. Here, the total income might not be a single attribute but composed of several attributes with one attribute per income source. Therefore, the attributes need to be added together before comparing to the limit amount; some cases may need more complex operations on one or more attributes.

These capabilities are not possible with BBS+ alone, and here is where zk-SNARK comes into play, as it can be used to prove arbitrary conditions on the attributes. Just to give the reader an idea, a Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK) is a cryptographic proof that allows one party (the prover) to demonstrate to another party (the verifier) that they know a specific piece of information without revealing the information itself and without requiring any interaction between the prover and verifier after the initial proof setup. A more in-depth discussion of zk-SNARKs, including their structure, properties, and practical applications, will be presented in Chapter III.

Chapter II

# ZKP: Zero-Knowledge Proofs

Prior to presenting a formal definition of zero-knowledge proofs, it is necessary to revisit several foundational definitions and concepts from the field of cryptography, which will serve as the basis for the subsequent discussion.

**Definition II.0.1. Computational Security (Informal)**
A scheme achieves computational security if any "efficient adversary" succeeds in "breaking the scheme" with at most "negligible" probability.

To make this definition precise, we have to formalize the following concepts:

- "Efficient adversary", which means probabilistic polynomial-time (PPT) algorithm. An algorithm $\mathcal{A}$ is polynomial time if its complexity is $\mathcal{O}(n^c)$ for some constant $c > 0$ and linear time when the complexity is $\mathcal{O}(n)$, where $n$ is the number of bits of the input.
- "Breaking the scheme", which means succeeding at an experiment (or game).
- "Negligible", which means less than any negative power of a security parameter $\lambda$. Parameters of the cryptographic schemes are chosen in such a way that the best-known attack would break the scheme using at least $2^\lambda$ operations.

A function $\mu(\lambda)$ is negligible in $\lambda$ if, for every positive integer $c$, there exists a $\lambda_0$ such that, for each $\lambda > \lambda_0$, we get $\mu(\lambda) < \frac{1}{\lambda^c}$.

# II.1 Hash functions and commitments

**Definition II.1.1.** A **hash function** (with output length $l(\lambda)$) is a pair of PPT algorithms (Gen, H) satisfying the following:

- Gen is a probabilistic algorithm that takes as input a security parameter $\lambda$ in unary and outputs a key k.
- H takes as input a key k and a string $x \in \{0,1\}^*$and outputs a string $H_k(x) \in \{0,1\}^{l(\lambda)}$.

**Definition II.1.2.** Let (Gen, H) be a hash function. A **collision** for a given key $k \leftarrow Gen(1^\lambda)$ of H is a pair $x, x' \in \{0,1\}^*$ such that $x \neq x'$ and $H_k(x) = H_k(x')$.

The **invert experiment**, which we denote $Invert_{\mathcal{AH}}(1^\lambda)$ works as follows:

- Generate a random $x \xleftarrow{\$} \{0,1\}^*$
- Compute $y \longleftarrow H_k(x)$ for some $k \xleftarrow{\$} Gen(1^\lambda)$
- Give $k$ and $y$ to the adversary $\mathcal{A}$
- Let $\mathcal{A}$ output $x'$
- Return 1 if $H_k(x') = y$, 0 otherwise

**Definition II.1.3.** A hash function $\mathcal{H} = (Gen, H)$ is **one way** if the following two properties hold:

- **Easy to compute**: there exists a polynomial time algorithm to compute H.
- **Hard to invert**: for every PPT algorithm $\mathcal{A}$, there exists a negligible function $negl(\lambda)$ such that $Pr[Invert_{\mathcal{AH}}(1^\lambda) = 1] \leq negl(\lambda)$

Commitment schemes are now introduced as fundamental cryptographic primitive used to bind data while preserving its confidentiality.

**Definition II.1.4.** A **commitment scheme** on a message space $\mathcal{M}$ is a triple of PPT algorithms (PGen, Commit, Open) such that:

1. PGen($1^\lambda$) takes as input a security parameter $\lambda$ in unary and returns the public parameters pp used by the scheme;

2. Commit(pp,$m$) takes as input the public parameters pp and a message $m$ in $\mathcal{M}$. It returns the commitment com and the opening material $o$;

3. Open(pp,$m$,com,o) takes as input the public parameters pp, the message $m$, the commitment com and the opening material o. It returns **Accept** if com is the commitment of $m$ or **Reject** otherwise.

A commitment must satisfy two properties: hiding and binding.

Hiding means that com reveals nothing about $m$ and binding means that it is not possible to create a commitment com that can be opened in two different ways. These properties can be formally defined.

**Definition II.1.5.** Let $\Pi_{Com}$ = (PGen, Commit, Open) be a commitment scheme and let Hiding($\Pi_{Com}$) be the hiding game represented in Figure II.1. A commitment scheme $\Pi_{Com}$ is **computationally hiding** if for all PPT adversaries $\mathcal{A}$ there is a negligible function $negl(\lambda)$ such that

$$|Pr[\mathcal{A} \text{ wins Hiding}(\Pi_{Com})]| \leq \frac{1}{2} + negl(\lambda)$$

If, for every pair $m_0, m_1$, the commitments $com_0$ and $com_1$ have the same distribution, where $(com_i, o_i) = Commit(pp, m_i)$ for i=0,1, we say that the commitment is **perfectly hiding**.

**Definition II.1.6.** A commitment scheme $\Pi_{Com}$=(PGen, Commit, Open) is **computationally binding** if for all PPT adversaries $\mathcal{A}$ there is a negligible function $negl(\lambda)$ such that

$$\Pr\left[ (com, m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(\mathsf{pp})) \;\middle|\; \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{PGen}(1^\lambda), \\ m_0 \neq m_1, \\ \mathsf{Open}(m_0, \mathsf{com}, r_0) = \textbf{Accept} \\ \mathsf{Open}(m_1, \mathsf{com}, r_1) = \textbf{Accept} \end{array} \right] \leq \text{negl}(\lambda)$$

If for every adversary $\mathcal{A}$ it holds that $negl(\lambda) = 0$, the commitment scheme is said to be **perfectly binding**.

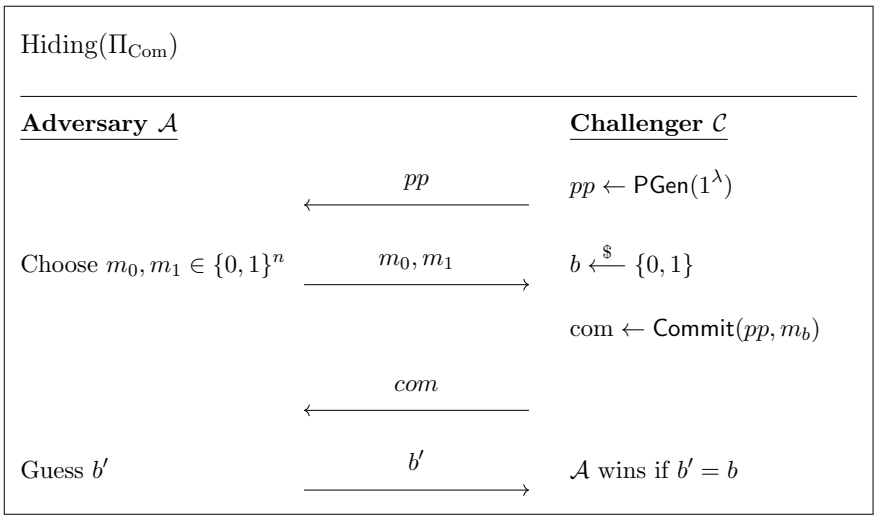



Figure II.1: Hiding game for commitment schemes

There are many different assumptions that can be used to build secure schemes: one well-known example is the **Discrete Logarithm Problem** assumption (DLP) [McC90]. Given a finite cyclic group $G$ of prime order $p$, generated by an element $g$, and any element $a \in G$, there exists an exponent $x \in \mathbb{F}_p$ such that $g^x = a$. However, computing $x$ from the known values of $g$ and $a$ is not computationally efficient when the order of $G$ is sufficiently large. The same assumption holds for elliptic curves (ECDLP): given an elliptic curve $E$ defined on a finite field $\mathbb{F}_q$ and given two points of the curve $G$ and $P$, it is unfeasible to find the value d such that $P = dG$.

One famous commitment based on the difficulty of the discrete logarithm is the **Pedersen commitment** [Ped92].

Let $G$ be a finite cyclic group of prime order $p$.

1. The public parameters $\mathsf{pp}$ of the scheme consist of two randomly chosen generators $g, h \in G$, such that $g = h^s$ for some exponent $s \in \mathbb{F}_p$ which remains computationally hidden and unknown to all parties.

2. Given a message $m$ to be committed, the committer samples a random

value $r \xleftarrow{\$} \mathbb{F}_p$, referred to as the *blinding factor*, and computes the commitment as

$$com \leftarrow g^m \cdot h^r.$$

The opening material is the couple $o = (m, r)$.

3. To open the commitment, the verifier receives the opening material $o$ and verifies its correctness by checking whether

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

**Theorem II.1.7.** *The Pedersen commitment is perfectly hiding and computationally binding, under the discrete logarithm assumption.*

This scheme also has an interesting property: it is **additively homomorphic**. In fact, it is easy to prove that

$$com(m, r) \cdot com(m', r') = com(m + m', r + r')$$

.

## II.2 Complexity classes

Let $\{0, 1\}^*$ denote the set of all finite binary strings.

**Definition II.2.1.** A **(formal) language** $\mathcal{L}$ is a subset of $\{0, 1\}^*$ or, in other words, a set of finite binary strings.

**Definition II.2.2.** Let $\mathcal{L}$ be a language. The **decision problem** for $\mathcal{L}$ is the problem that, given a binary string $x \in \{0, 1\}^*$, asks to determine if $x$ belongs or does not belong to $\mathcal{L}$.

In other words, a decision problem is a problem whose answer can be only YES or NO. However, most of the interesting problems are not decision problems, since the answer can be more complex (these are called **search problems**). For example, the factoring problem asks to find the prime numbers p and q after being given $N = pq$. Luckily, most of the time one can easily reduce a search problem to a finite number of decision problems. For instance, the factoring problem can be equivalently formulated as the decision problems:

1. Is the first (binary) digit of p equal to 1? (yes or no)
2. Is the second digit of p equal to 1? (yes or no) ... and from the answers one can easily compute p (and then $q = N/p$).

**Definition II.2.3.** A **complexity class** is a set of decision problems that can be solved by algorithms of a specific complexity. Interchangeably, a complexity class is a set of languages whose decision problems can be solved by algorithms of a specific complexity.

Some important complexity classes are the following.

- **P**, which is the class of all decision problems that can be solved in polynomial-time.
- **BPP** (Bounded-error Probabilistic Polynomial-time), which is the class of all decision problems that can be solved by PPT algorithms that are allowed to return a wrong answer with probability at most 1/3.
- **PSPACE**, which is the class of all decision problems that can be solved by an algorithm using a polynomial amount of memory.
- **NP** (Nondeterministic Polynomial-time), which is the class of all decision problems whose solutions can be verified in polynomial time.

**Definition II.2.4.** The **NP (Nondeterministic Polynomial-time)** complexity class is defined as follows. A language $\mathcal{L}$ belongs to **NP** if and only if there exists a polynomial $p$ and a polynomial-time algorithm $\mathcal{A}$ such that:

- For all $x \in \mathcal{L}$ there exists a **witness** (or **proof**) $w \in \{0,1\}^*$ such that $|w| \leq p(|x|)$ and $\mathcal{A}(x, w) = 1$.
- For all $x \notin \mathcal{L}$ and for all $w \in \{0,1\}^{p(|x|)}$ we have that $\mathcal{A}(x, w) = 0$.

Informally, a language $\mathcal{L}$ belongs to NP if for every true statement "$x \in \mathcal{L}$" there exists a "short" proof $w$ that can be used to efficiently verify the truthfulness of "$x \in \mathcal{L}$". The term nondeterministic comes from an equivalent definition of NP that employs nondeterministic Turing machines.

## II.3 Zero-Knowledge Proofs

**Definition II.3.1.** A k-round **interaction** of two functions $f, g : \{0,1\}^* \to \{0,1\}^*$ on input $x \in \{0,1\}^*$ is a sequence of strings $a_1, \ldots, a_k \in \{0,1\}^*$ defined as

$a_1 \coloneqq f(x)$
$a_2 \coloneqq g(x, a_1)$
$a_3 \coloneqq f(x, a_1, a_2)$
$\ldots$
$a_{2i+1} \coloneqq f(x, a_1, \ldots, a_{2i})$ for $2i < k$
$a_{2i+2} \coloneqq g(x, a_1, \ldots, a_{2i+1})$ for $2i + 1 < k$.

The final output of $f$ is denoted by $\text{out}_f\langle f, g\rangle(x)$.

**Definition II.3.2.** A language $\mathcal{L}$ is said to be in the complexity class **IP** if there exists a PPT algorithm $V$ (**verifier**) that can have a $k$ -round interaction with a function $P : \{0,1\}^* \to \{0,1\}^*$ (**prover**) such that the following two properties hold.

- **Completeness.** If $x \in \mathcal{L}$ then there exists $P$ such that

$$Pr[\text{out}_V\langle V, P\rangle(x) = 1] \geq 2/3$$

.

- **Soundness.** If $x \notin \mathcal{L}$ then for every $\widetilde{P}$ we have that

$$Pr[\text{out}_V\langle V, \widetilde{P}\rangle(x) = 1] \leq 1/3$$

.

A pair of $(V, P)$ satisfying completeness, is called an **interactive proof system** for $\mathcal{L}$.

Roughly speaking $\mathcal{L} \in IP$ if, for every $x \in \mathcal{L}$, there exists an interactive proof that, with overwhelming probability, convinces the verifier that $x \in \mathcal{L}$; while if $x \notin \mathcal{L}$, such a proof is impossible.

The class **IP** does not change if its definition is changed by:

- allowing the prover $P$ to be probabilistic;
- replacing the probability 2/3 with 1 (this is a nontrivial theorem);
- replacing the probability 1/3 with any fixed constant $c < 1$ (this follows by repeating the interactive proof several times). In such a context, c is called **soundness error** or **cheating probability**.

**Definition II.3.3.** Let $\mathcal{L}$ be a language in **NP**. A pair (P, V) of interactive PPT algorithms is a perfect (statistical, computational, respectively) **zero-knowledge proof (ZKP)** for $\mathcal{L}$ if the following three properties hold.

- **Completeness.** For every $x \in \mathcal{L}$ and for every witness $w$ of this fact, we have that $\Pr[\text{out}_V\langle P(x,w), V(x)\rangle = 1] \geq 2/3$
- **Soundness.** For every $x \notin \mathcal{L}$ and for every (unbounded) probabilistic algorithm $\widetilde{P}$, we have that $\Pr[\text{out}_V\langle \widetilde{P}(x), V(x)\rangle = 1] \leq 1/3$
- **Zero-Knowledge.** For every PPT algorithm $V^*$, there exists an expected PPT algorithm S (**simulator**) such that for every $x \in \mathcal{L}$ and for every witness $w$ of this fact, the random variables $\text{out}_{V^*}\langle P(x,w), V^*(x)\rangle$ and $S(x)$ are identical (statistically indistinguishable, computationally indistinguishable, respectively)

The complexity classes of languages admitting perfect, statistical, and computational ZKPs are called **PZK**, **SZK** and **CZK** respectively.

The completeness and soundness properties of ZKPs are analogous to that of **IP** and the same considerations apply (2/3 can be replaced by 1, and 1/3 can be made arbitrarily small). The zero-knowledge property says that the verifier cannot learn anything new from the interaction, even if he employs a different strategy $V^*$. Indeed, he could have obtained the same information by directly executing the simulator $S$ on the publicly known input $x$. Occasionally, the zero-knowledge property might be too difficult to handle, since it takes into account every possible strategy of the verifier. Therefore, it is relaxed to the following weaker property, which assumes that the verifier applies a fixed (honest) strategy.

- **Honest-Verifier Zero-Knowledge.** There exists an expected PPT algorithm $S$ (**simulator**) such that for every $x \in \mathcal{L}$ and for every witness $w$ of this fact, the random variables $\text{out}_V\langle P(x,w), V(x)\rangle$ and $S(x)$ are identical (statistically indistinguishable, computationally indistinguishable, resp.)

**Definition II.3.4.** Let $\mathcal{L}$ be a language in **NP**. A pair $(P, V)$ of interactive PPT algorithms is a **proof of knowledge (PK)** for $\mathcal{L}$ if the following property holds.

- **Knowledge soundness** (or **knowledge extractability**). There exists a constant $k > 0$ (**knowledge error**) and an expected PPT algorithm E (**extractor**) such that for every interactive function $\widetilde{P}$ and every $x \in \{0,1\}^*$ the following condition holds:
  "If $p := \Pr[\text{out}_V\langle \widetilde{P}, V\rangle] > k$ then,on input $x$ and access to the oracle $\widetilde{P}(x)$, the extractor $E$ returns a witness $w$ for $x \in \mathcal{L}$ within a number of steps bounded by $1/(p-k)$ times a fixed polynomial of $|x|$."

The definition of proof of knowledge is very involved. However, roughly speaking, it says that given any algorithm $\widetilde{P}$ that convinces the verifier that $x \in \mathcal{L}$, it is possible to build another algorithm $E$ that produces a witness $w$ for the fact that $x \in \mathcal{L}$.

The original definition of knowledge soundness given by Bellare and Goldreich [BG93] was restricted to $x \in \mathcal{L}$. However, it does no harm to extend the definition to any $x \in \{0,1\}^*$, as it is common today [Cou17]. Then it can be proved that knowledge soundness implies soundness. At this point, zero-knowledge proofs of knowledge can be defined as zero-knowledge proofs (completeness, soundness, zero-knowledge) which are also proofs of knowledge (knowledge soundness).

**Definition II.3.5.** Let $\mathcal{L}$ be a language in NP. A pair $(P, V)$ of interactive PPT algorithms is a perfect (statistical, computational, resp.) **zero-knowledge proof of knowledge (ZKPoK)** for $\mathcal{L}$ if:

- **Completeness.** For every $x \in \mathcal{L}$ and for every witness w of this fact, we have $\Pr[\text{out}_V\langle P(x,w), V(x)\rangle = 1] \geq 2/3$.

- **Knowledge soundness.** There exists a constant $k > 0$ (**knowledge error**) and an expected PPT algorithm $E$ (**extractor**) such that for every interactive function $\widetilde{P}$ and every $x \in \mathcal{L}$ the following condition holds:
  If $p \coloneqq \Pr[\text{out}_V\langle \widetilde{P}, V\rangle] > k$ then, on input $x$ and access to the oracle $\widetilde{P}(x)$, the extractor $E$ returns a witness $w$ for $x \in \mathcal{L}$ within a number of steps bounded by $1/(p-k)$ times a fixed polynomial of $|x|$.

- **Zero-Knowledge.** For every PPT algorithm $V^*$, there exists an expected PPT algorithm $S$ (**simulator**) such that for every $x \in \mathcal{L}$ and for every witness $w$ of this fact, the random variables

$$\text{out}_{V^*}\langle P(x,w), V^*(x)\rangle > k$$

  and $S(x)$ are identical (statistically indistinguishable, computationally indistinguishable, resp.)

The complexity classes of languages admitting perfect, statistical, computational ZKPoKs are called **PZKPoK**, **SZKPoK**, **CZKPoK**, respectively.

## II.4 Sigma-Protocols

Before diving into the technical construction of zk-STARKs, it is useful to briefly recall the concept of *Sigma-protocols* (or $\Sigma$-protocols). These are three-move interactive proof systems that have historically played a central role in the development of zero-knowledge proofs and proofs of knowledge.

A Sigma-protocol consists of a prover and a verifier engaging in a simple interaction: the prover sends a commitment, the verifier responds with a challenge, and the prover replies with a response. Despite their simplicity, Sigma-protocols satisfy important properties such as completeness, special soundness, and special honest-verifier zero-knowledge. These properties make them foundational tools in the design of cryptographic protocols, especially in scenarios where proving knowledge of a secret without revealing it is essential.

While zk-STARKs differ significantly in structure, being non-interactive, scalable, and post-quantum secure, the conceptual goals remain similar.

Presenting Sigma-protocols helps build intuition about the nature of zero-knowledge and how these goals are achieved in more advanced proof systems.

A Σ-protocol is a particular honest-verifier ZKPoK with a 3-pass structure.

**Definition II.4.1.** Let $\mathcal{L}$ be a language in NP. A **Σ-protocol** is a 3-pass interactive proof between a prover and a verifier having the following structure:

1. The prover's input is $x \in L$ and a witness $w$ of that fact.
2. The verifier's input is x.
3. The prover sends to the verifier a **commitment**.
4. The verifier sends to the prover a **response** and sends it to the verifier.
5. The verifier checks if the response is correct, according to the challenge and the commitment. In such a case the verifier accepts, otherwise he rejects.

The prover consists of two PPT algorithms $P_1$, $P_2$ and the verifier consists of two PPT algorithms $V_1$,$V_2$. The following scheme summarizes the execution of the protocol of **transcript** (x, com, ch, rsp).

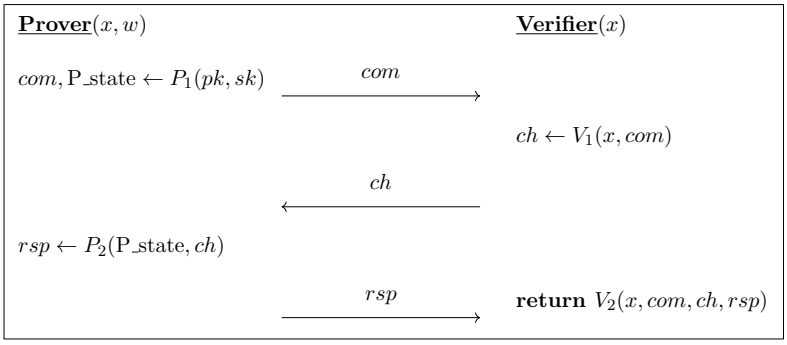



Figure II.2: Σ-protocol

Moreover, the following properties hold.

- **Completeness.** If the parties follow the protocol for $x \in \mathcal{L}$ the verifier always accepts.
- **Special Soundness.** There exists an expected PPT algorithm (**extractor**) that takes as input two transcripts ($x$, com, $\text{ch}_1$, $\text{rsp}_1$) and ($x$, com, $\text{ch}_2$, $\text{rsp}_2$), with $\text{ch}_1 \neq \text{ch}_2$, of an honest execution of the protocol, and returns as output a witness $w'$ of $x \in \mathcal{L}$. (Here $w$ and $w'$ may or may not be equal.)
- **Honest Verifier Zero Knowledge (HVZK).** There exists an expected PPT algorithm, called **simulator**, that takes as input $x$ and returns as output a random transcript ($x$, com, ch, rsp) having identical (statistically indistinguishable, computationally indistinguishable, resp.) probability distribution of transcripts of an honest execution of the protocol on input $(x, w)$.

The name "Σ-protocol" was used for the first time by Cramer [Cra97]. Usually, the verifier selects the challenge by simply picking a random element of $\mathcal{C}$. The notion of completeness is the more natural and it simply means that the protocol should work if every party is honest. Usually, proving completeness is straightforward. The notion of special soundness expresses the idea that if a cheater is able to answer to more challenges for the same commitment, then he is also able to compute the witness. In other words, cheating is as difficult as computing the witness. The property of honest verifier zero-knowledge says that a simulator, who does not know the witness, can still produce transcripts that are indistinguishable from a legitimate execution of the protocol. Therefore, the transcripts of a legitimate execution of the protocol contain no information (zero-knowledge) on the witness. As in ZKPs, the **soundness error** (or **cheating probability**) of a Σ-protocol is the probability that a verifier accepts the response of a cheater who does not know the witness.

The **Fiat-Shamir transform** [FS99] is a technique used to convert a Sigma protocol into a digital signature scheme. The basic idea is to replace the verifier challenge generation with a hash function (Fig. II.3).

The transform can be used only if the protocol is public coin, that is when the random generated by the verifier is also known by the prover in the interactive version of the protocol.

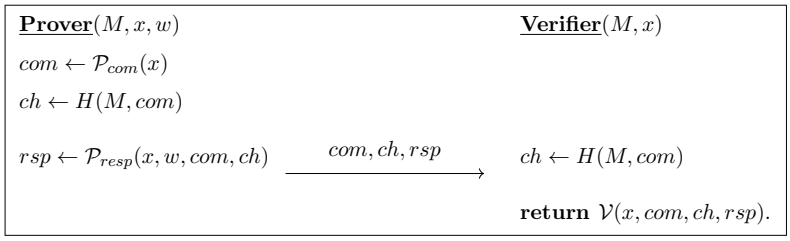



Figure II.3: Fiat-Shamir transform

**Theorem II.4.2.** *Any Sigma protocol that is transformed with the Fiat-Shamir transform is non-interactive in the Random Oracle Model.*

The **Random Oracle Model** (ROM) assumes the existence of a publicly accessible oracle that behaves as a black box, implementing a truly random function. Although such oracles cannot exist in practice, they are typically instantiated using cryptographic hash functions that are designed to approximate the properties of random functions. The ROM is particularly useful for conducting security proofs. The core idea is as follows: if an adversary $\mathcal{A}$ is able to break a cryptographic scheme that is proven secure in the ROM, then any practical instantiation of the random oracle, such as a cryptographic hash function, would be insufficient for ensuring security in that context.

While security proofs in the Random Oracle Model do not guarantee absolute security in practice, they provide a meaningful level of assurance based on the assumption that well-designed cryptographic hash functions closely emulate the behavior of a truly random oracle. For a broader discussion on the notion of oracles in cryptographic proof systems, including zero-knowledge proofs (ZKPs), interactive proofs (IPs), and interactive oracle proofs (IOPs), the reader is referred to the Appendix B.

## II.5 Identification Schemes

Identification schemes extend the ideas introduced by Sigma protocols. In these schemes, the prover's goal is to authenticate themselves (e.g., prove they know a secret or have certain knowledge) to a verifier, but without disclosing any private information. This concept is pivotal because it applies to numerous real-world cryptographic applications like authentication, secure communication, and privacy-preserving transactions. Understanding

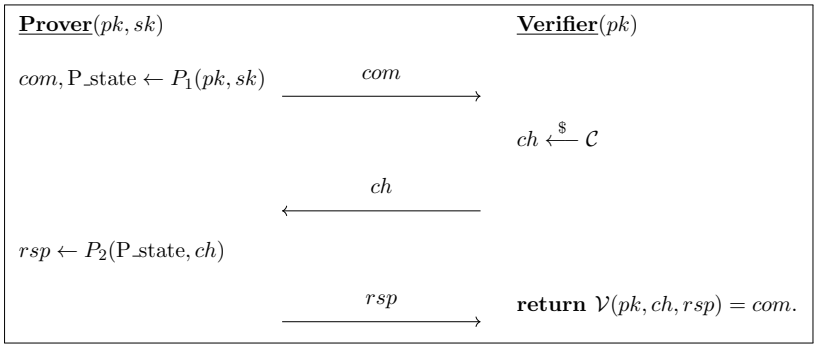
identification schemes provides the necessary groundwork for grasping more sophisticated zero-knowledge protocols, such as zk-STARKs. zk-STARKs (Zero-Knowledge Scalable Transparent Arguments of Knowledge) are a modern and highly scalable class of zero-knowledge proofs. While Sigma protocols and identification schemes operate on a small scale, zk-STARKs aim to scale up this idea while maintaining transparency and security without relying on a trusted setup.

By grasping the core principles of identification schemes, which often build on the structure of Sigma protocols, one can more easily understand the advantages and challenges associated with zk-STARKs. These include their scalability, transparency (without the need for a trusted setup), and their ability to prove knowledge of large datasets or complex computations while preserving privacy.

An identification scheme is an interactive method that allows a prover to prove his identity to a verifier. The prover has a public key, which is publicly known, and a secret key, which he keeps for himself. The prover identifies himself to the verifier by providing a ZKPoK of his secret key.

**Definition II.5.1.** A **(3-pass) identification scheme** consists of PPT algorithms Gen, $P_1$, $P_2$, $V$ such that:

- The **key-generation** algorithm Gen takes as input $1^\lambda$ and returns as output the public key pk and the secret key sk of the prover.
- The prover and the verifier runs $P_1$, $P_2$, $V$ as follows (here $\mathcal{C}$ is the space of challenges).



And, if everybody behaves honestly, the verifier returns 1 (**accept**).

It can be also introduced the **Communication cost**, that is, the amount of information (in bits) that the prover and the verifier have to exchange in a full identification.

**Lemma II.5.2.** *Assume that the verifier picks the challenge at random with uniform distribution. Then the communication cost is equal to*

$$R \cdot \left( |com| + \frac{1}{\#\mathcal{C}} \sum_{ch \in \mathcal{C}} (|ch| + |rsp_{ch}|) \right)$$

*where $R$ is the number of rounds and $rsp_{ch}$ is the response to the challenge $ch$.*

**Argument systems** are typically obtained combining two different tools:

1. An information-theoretically secure protocol, such as an IP, or a probabilistically checkable proof (PCP);
2. Cryptography, which is used to restrict the prover, thus obtaining an argument system.

Cryptography can also provide the argument system with other important properties, such as zero-knowledge, succinctness and non-interactivity. An argument system that satisfies all these properties is called zk-SNARK (zero knowledge Succinct Non interactive ARgument of Knowledge).

Chapter III

# SNARKs

## III.1 Preliminaries

Interactive proofs usually exploit the following basic property of polynomials, known as **Schwartz-Zippel Lemma** [Sch79; Zip79].

**Lemma III.1.1.** *Let $\mathbb{F}$ be a field (finite or infinite), and let $g \in \mathbb{F}[X_1, \ldots, X_m]$ be a nonzero formal polynomial in $m$ variables, of total degree at most $d$. Let $S \subseteq \mathbb{F}$ be a finite subset. If $x = (x_1, \ldots, x_m)$ is chosen uniformly at random from $S^m$, then*

$$\Pr_{x \leftarrow S^m}[g(x) = 0] \leq \frac{d}{|S|}.$$

*Note*: In a finite field, a nonzero formal polynomial may induce a function that vanishes at every point of the domain. For example, over $\mathbb{F}_p$, the polynomial $X^p - X$ is formally nonzero but vanishes on every $x \in \mathbb{F}_p$. The lemma applies to the polynomial as a formal object, not to the induced function. Therefore, it is essential to maintain this distinction.

An easy implication of this Lemma is that, for any pair of distinct $m$-variate polynomials $p$ and $q$ of total degree at most $d$ over $\mathbb{F}$, $p(x) = q(x)$ for at most $\frac{d}{|\mathbb{F}|}$ fraction of inputs. Suppose that a vector $\mathbf{a} = [a_1, \ldots, a_n]$ is given. One natural way to associate this vector with a polynomial is to treat

the elements of $\mathbf{a}$ as the coefficients of the polynomial. In this case:

$$p_{\mathbf{a}}(X) = \sum_{i=1}^{n} a_i X^{i-1}$$

Alternatively, the vector $\mathbf{a}$ can be interpreted as a set of evaluations of a polynomial at a canonical set of inputs, typically $\{1, \ldots, n\}$. In this perspective, the unique (univariate) polynomial that matches these evaluation values can be recovered using *Lagrange interpolation*, a well-known method for reconstructing a polynomial from point-value pairs.

**Lemma III.1.2.** *Let $p$ be a prime number larger than $n$. For any vector $a = [a_1, ..., a_n] \in \mathbb{F}_p^n$, there is a unique polynomial $q_a$ of degree at most $n-1$ such that*

$$q_a(i) = a_{i+1}, \quad i \in \{0, ..., n-1\}.$$

This polynomial is given by

$$q_a(X) = \sum_{i=0}^{n-1} a_{i+1} \delta_i(X),$$

where

$$\delta_i(X) = \prod_{k=0,\ldots,n-1:k\neq i} \frac{X-k}{i-k}.$$

$\delta_i(X)$ is usually referred to as the $i$-th Lagrange basis polynomial. The most efficient way to evaluate $q_a$ in a random point requires $O(n)$ time.

### III.1.1 Multilinear Extensions (MLE)

In the context of STARKs and other proof systems, many computations are defined over discrete structures, such as vectors or tables indexed by elements of $\{0,1\}^n$. To work with these structures algebraically, it is often convenient to represent them as polynomials defined over a finite field.

A *multilinear extension* is a technique that allows us to extend a function defined over $\{0,1\}^n$ to the entire field $\mathbb{F}^n$, by constructing a unique polynomial that is multilinear in each variable and agrees with the original function on all Boolean inputs.

Consider a multivariate function $f$, defined over the domain $\{0,1\}^v$. If $v = \log_2(n)$, the domains $\{0,1\}^v$ and $\{1, ..., n\}$ have the same cardinality.

**Definition III.1.3.** Let $\mathbb{F}$ be any field, and let $f : \{0,1\}^v \to \mathbb{F}$ be any function. A $v$-variate polynomial $g$ over $\mathbb{F}$ is an **extension** of $f$ if $g(x) = f(x)$ for all $x \in \{0,1\}^v$.

Any function $f : \{0,1\}^v \to \mathbb{F}$ has a *multilinear polynomial extension*, that is it has degree at most 1 in each variable. This means that the total degree is at most $v$, which is logarithmic in the domain size $2^v$.

**Lemma III.1.4.** *Any function $f : \{0,1\}^v \to \mathbb{F}$ has a unique multilinear extension (MLE) over $F$.*

The unique MLE is denoted by $\tilde{f}$.

**Lemma III.1.5.** *Let $f : \{0,1\}^v \to \mathbb{F}$ be any function. Then the following $\tilde{f} : \mathbb{F}^v \to \mathbb{F}$ is the unique MLE of $f$:*

$$\tilde{f}(x_1, ..., x_v) = \sum_{w \in \{0,1\}^v} f(w) \cdot \prod_{i=1}^{v} (x_i w_i + (1 - x_i)(1 - w_i)),$$

*where $w = (w_1, ..., w_v) \in \{0,1\}^v$.*

Suppose that $n = 2^v$. Then, the most efficient way for evaluating $\tilde{f}$ in a random point $r \in \mathbb{F}^v$ requires $O(n)$ time and $O(n)$ space [VSBW13].

**Example III.1.1.** Let $p = 11$, and consider the function $f : \{0,1\}^2 \to \mathbb{F}_p$ given by $f(0,0) = 3, f(0,1) = 4, f(1,0) = 1, f(1,1) = 2$. Write out an explicit expression for the MLE $\tilde{f}$ of $f$.

In this example, $v = 2$. The expression can be computed as follows:

$$\tilde{f}(x_1, x_2) = \sum_{w \in \{0,1\}^2} f(w) \cdot \prod_{i=1}^{2} (x_i w_i + (1 - x_i)(1 - w_i)).$$

So it holds:

$$\begin{aligned}
\tilde{f}(x_1, x_2) &= f(0,0)(1-x_1)(1-x_2) + f(0,1)(1-x_1)x_2 + \\
&\quad + f(1,0)x_1(1-x_2) + f(1,1)x_1x_2 \\
&= 3(1-x_1)(1-x_2) + 4(1-x_1)x_2 + 1 \cdot x_1(1-x_2) + 2x_1x_2 \\
&= 3(1-x_1-x_2+x_1x_2) + 4(x_2-x_1x_2) + (x_1-x_1x_2) + 2x_1x_2 \\
&= 3 - 3x_1 - 3x_2 + 3x_1x_2 + 4x_2 - 4x_1x_2 + x_1 - x_1x_2 + 2x_1x_2 \\
&= (3) + (-3x_1 + x_1) + (-3x_2 + 4x_2) + \\
&\quad + (3x_1x_2 - 4x_1x_2 - x_1x_2 + 2x_1x_2) \\
&= 3 - 2x_1 + x_2 + (3 - 4 - 1 + 2)x_1x_2 \\
&= 3 - 2x_1 + x_2 + 0 \cdot x_1x_2 \\
&= 3 - 2x_1 + x_2.
\end{aligned}$$

### III.1.2 Arithmetic circuits

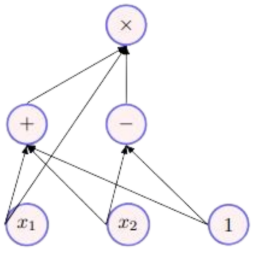
**Definition III.1.6.** Fix a field $\mathbb{F}_p$ with $p > 2$. An **arithmetic circuit** $C : \mathbb{F}_p^n \to \mathbb{F}_p$ is a directed acyclic graph (DAG) where inputs are $\{1, \ldots, n\}$ and internal nodes are $\{+, -, \times\}$.

An arithmetic circuit always defines an $n$-variate polynomial, and the number of gates (the size) of the circuit $\mathcal{C}$ is denoted with $|C|$.

**Example III.1.2.** For example, the circuit related to the polynomial

$$f(x_1, x_2) = x_1(x_1 + x_2 + 1)(x_2 - 1)$$

is the following (it has $|\mathcal{C}| = 3$ gates):



Differently from the addition or multiplication gate, in the subtract gate the ordering actually matters. Arithmetic circuits avoid subtraction gates entirely by rewriting expressions. For instance, suppose to compute $x - y$. It's possible to rewrite it using two gates:

1. A first one (a multiplication gate) to compute $(-1) \cdot y$;
2. A second one (an addition gate) to compute $x + (-y)$.

The arithmetic circuit for a given polynomial is not unique.

**Definition III.1.7.** An arithmetic circuit is **unstructured** if it has arbitrary wires. A circuit is **structured** if it is divided into layers, and layer $n$ receives data from layer $n - 1$ and sends data to layer $n + 1$.

Two classic problems when dealing with arithmetic circuits are the **circuit evaluation problem** and the **circuit satisfiability problem** ($\mathsf{circuit} - \mathsf{SAT}$). In the former, the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ agree on some arithmetic circuit $\mathcal{C}$, and the goal is to compute the value(s) of the output gate(s) of $\mathcal{C}$, given the input vector $x$. In the latter, given a circuit $\mathcal{C}$ that takes two inputs, a public one $x$, and a private one $w$ (called the witness and known only by the prover P), the prover wants to convince the verifier $\mathcal{V}$ that indeed there exists a witness $w$ such that $\mathcal{C}(x, w) = y$, where $y$ is a public output. Implementing an argument system typically involves two main phases:

- **Front-end**
  The computation described by a general-purpose program is transformed into an equivalent representation, such as an arithmetic circuit or an instance of arithmetic circuit satisfiability.
- **Back-end**
  A cryptographic proof system (interactive or non-interactive) is then applied to verify that the circuit has been correctly evaluated, without requiring re-execution of the computation.

Importantly, any arbitrary program can be compiled into an arithmetic circuit. Furthermore, verifying the correctness of a proof (i.e., solving the circuit satisfiability problem) typically requires significantly less computational effort than performing the original computation itself.

### III.1.3 Rank-1 Constraint Systems (R1CS)

A representation closely related to the arithmetic circuit satisfiability problem is the *Rank-1 Constraint System* (R1CS). This formalism is widely used in the design of zero-knowledge proof systems due to its algebraic structure and compatibility with cryptographic protocols.

An R1CS instance over a finite field $\mathbb{F}_q$ is defined by three matrices $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{F}_q^{m \times n}$. A vector $\mathbf{z} = (z_0, z_1, \ldots, z_{n-1}) \in \mathbb{F}_q^n$, with $z_0 = 1$ (often referred to as the *constant term*), is said to satisfy the system if the following component-wise equation holds:

$$(\mathbf{A} \cdot z) \circ (\mathbf{B} \cdot z) = \mathbf{C} \cdot z,$$

where $\circ$ denotes the Hadamard (element-wise) product.

Every instance of the circuit-SAT problem can be translated into an equivalent R1CS instance. The matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ encode the constraints corresponding to each gate of the arithmetic circuit, and the number of nonzero entries in any row is bounded by the *fan-in* of the corresponding gate (typically equal to 2 in most practical constructions).

Similarly, the *fan-out* of a circuit refers to the number of gates that receive the output of a given gate as input. While R1CS primarily models the fan-in behavior, careful circuit design can manage fan-out effectively through intermediate variables and duplication of constraints.

This construction implies that the matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are inherently *sparse*, as each constraint involves only a limited number of variables.

Let $|x|$ denote the length of the public input, and $|w|$ the length of the witness. Define $N = |\mathcal{C}| + |x| + |w|$, where $|\mathcal{C}|$ is the number of gates in the arithmetic circuit. The vector $z \in \mathbb{F}_q^n$, which includes all circuit wires, the public input, and the witness, is then of length $n = N + 1$, accounting for the leading constant term $z_0 = 1$.

Each row of the matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ corresponds to a single constraint in the R1CS instance. The total number of constraints $m$ is determined by the structure of the circuit and the nature of its inputs and outputs:

- One constraint is added for each element of the public input $x$;
- No constraint is added for entries of the witness $w$;
- One constraint is added for each internal (non-output) gate of the circuit;

- Two constraints are added for each output gate, in order to ensure correct propagation and output validity.

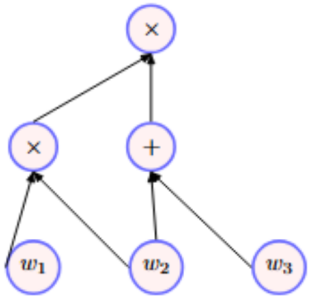
As a result, the total number of constraints is given by:

$$m = N - |w| + |y|,$$

where $|y|$ denotes the number of output wires of the circuit.

It is important to note that each constraint in an R1CS instance is always expressed as a single multiplication between two linear combinations of variables (mirroring the behavior of gates in an arithmetic circuit).

**Example III.1.3.** Let us consider the polynomial function $f(w_1, w_2, w_3) = (w_1 \cdot w_2)(w_2 + w_3)$ where the variables $w_1, w_2, w_3 \in \mathbb{F}_q$ represent private witness inputs, and the output $y \in \mathbb{F}_q$ is public. The polynomial $f(w_1, w_2, w_3)$ can be represented by the following arithmetic circuit:




This circuit has:

- No public inputs, i.e., $|x| = 0$;
- Three witness elements, i.e., $|w| = 3$;
- Three internal gates (two multiplications and one addition);
- One public output, i.e., $|y| = 1$.

To encode the computation, we introduce intermediate variables corresponding to the outputs of internal gates:

$$z_4 = z_1 \cdot z_2$$
$$z_5 = z_2 + z_3$$
$$z_6 = z_4 \cdot z_5$$
$$y = z_6$$

These operations translate into the following R1CS constraints:

$$(z_1 \cdot z_2) = z_4$$
$$(1 \cdot (z_2 + z_3)) = z_5$$
$$(z_4 \cdot z_5) = z_6$$
$$(1 \cdot z_6) = y$$

Each equation can be expressed in the R1CS form:
$(\mathbf{A} \cdot z) \circ (\mathbf{B} \cdot z) = \mathbf{C} \cdot z$, where $\mathbf{z} = (1, w_1, w_2, w_3, z_4, z_5, z_6, y)$ includes the constant term and all intermediate variables.

The previous constraints are encoded in the matrices as follows:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix},$$

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Each row in these matrices corresponds to one constraint in the form:

$$(\mathbf{A} \cdot z) \circ (\mathbf{B} \cdot z) = \mathbf{C} \cdot z$$

Thus, finding a valid witness $w = (w_1, w_2, w_3)$ such that the circuit computes $y = f(w)$ is equivalent to finding a vector $\mathbf{z} \in \mathbb{F}_q^8$ that satisfies all constraints defined by the R1CS matrices.

This example illustrates how even a simple polynomial can be represented using a sequence of R1CS constraints, enabling it to be used in zero-knowledge proof systems such as SNARKs or STARKs.

## III.2 Definition

**Definition III.2.1.** A **Succinct Non-interactive ARgument of Knowledge** (SNARK) is a triple of algorithms $(\mathsf{S}, \mathsf{P}, \mathsf{V})$ defined as follows:

- $(\mathsf{pp}, \mathsf{vp}) \leftarrow \mathsf{S}(\mathcal{C}, \lambda, r)$: a *setup algorithm* that, given a circuit $\mathcal{C}$, a security parameter $\lambda$, and some randomness $r$, generates a pair of public parameters: $\mathsf{pp}$ (prover parameters) and $\mathsf{vp}$ (verifier parameters).
- $\pi \leftarrow \mathsf{P}(\mathsf{pp}, x, w)$: a *proving algorithm* that, given the prover's parameters $\mathsf{pp}$, a public input $x$, and a witness $w$, outputs a proof $\pi$. The proof must be of *sublinear size* with respect to the size of the witness.
- $b \leftarrow \mathsf{V}(\mathsf{vp}, x, \pi)$: a *verification algorithm* that, given the verifier's parameters $\mathsf{vp}$, the public input $x$, and a proof $\pi$, outputs a bit $b \in \{0, 1\}$, where 1 indicates acceptance. The verifier must run in *sublinear time* with respect to the size of the circuit $\mathcal{C}$.

If the SNARK additionally satisfies the property of zero-knowledge (i.e., the proof $\pi$ reveals no information about the witness $w$) then the scheme is referred to as a **zero-knowledge SNARK** (zk-SNARK).

**Definition III.2.2.** An argument system for circuit satisfiability is said to be **succinct** if it satisfies the following properties:

- **Proof length:** The total communication (i.e., the proof size $\ell_\pi$) is sublinear in the length of the witness $|w|$, such as $\ell_\pi = \mathcal{O}(\log |w|)$. If

$\ell_\pi = \mathcal{O}(\log |\mathcal{C}|)$, where $|\mathcal{C}|$ is the size of the circuit, the argument is said to be *strongly succinct*.

- **Verification time:** The time required by the verifier to check the proof is sublinear in the size of the circuit, for example $\text{time}(\mathsf{V}) = \mathcal{O}(|x| + \log |\mathcal{C}|)$, where $|x|$ is the size of the public input.

Some definitions of succinctness only require the first condition (i.e., sublinear proof size), without imposing constraints on the verifier's runtime.

Before formally presenting the security properties of zk-SNARKs, an informal definition of knowledge soundness is given to capture the extractor-based security intuition behind these proof systems.

**Definition III.2.3.** A non-interactive argument system for circuit satisfiability is said to be **adaptively knowledge sound** for a circuit $\mathcal{C}$ if the following holds:

For any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ that outputs a proof $\pi$ and public input $x$ such that the verifier accepts with non-negligible probability $\kappa$, there exists a PPT extractor $\mathcal{E}$ (with black-box or non-black-box access to $\mathcal{A}$) that is able to extract a witness $w$ such that:

$$\Pr[\mathcal{C}(x, w) = y] = \kappa - \varepsilon,$$

for some negligible function $\varepsilon(\lambda)$, where $\lambda$ is the security parameter.

In this context, *adaptivity* means that the adversary is allowed to choose the statement $x$ after observing the public parameters generated by the setup algorithm.

Knowledge soundness implies the standard notion of soundness. However, when the prover is computationally unbounded, the stronger property of knowledge soundness may not hold. In such a case, the argument system is referred to as a **SNARG** (Succinct Non-interactive ARgument), which guarantees only *soundness* rather than *knowledge soundness*.

## III.3 Setup

When constructing a SNARK system, three main types of setup procedures can be distinguished:

- **Trusted setup per circuit:** In this approach, the setup phase generates random parameters specific to a particular circuit $\mathcal{C}$. It is crucial that the prover does not learn these secret random values; otherwise, they could forge proofs for false statements. This setup typically relies on a *trusted ceremony* to ensure the secrecy and integrity of the parameters.
- **Trusted but universal (updatable) setup:** The setup algorithm $S$ is divided into two sub-algorithms:

$$\begin{cases} \text{Generate a universal reference string:} & gp \leftarrow S_{\text{init}}(\lambda, r), \\ \text{Derive circuit-specific parameters:} & (pp, vp) \leftarrow S_{\text{index}}(gp, C), \end{cases}$$

  where $\lambda$ is the security parameter and $r$ denotes randomness. In this setting, the random values generated in $S_{\text{init}}$ are independent of the circuit $C$, allowing the setup to be reused across different circuits. Moreover, the universal parameters $gp$ can be updated to improve security without restarting the entire setup.
- **Transparent setup:** the setup algorithm does not rely on any secret randomness. Instead, it uses only publicly verifiable procedures, such as cryptographic hash functions or publicly known constants, to generate the parameters. This approach removes the need for trust in the setup phase but often comes with increased computational costs.

There are two main paradigms for building SNARKs:

- **Functional commitment schemes combined with Interactive Oracle Proofs (IOPs):** this approach employs functional commitment schemes such as polynomial commitments, multilinear commitments, vector commitments, or inner product commitments, paired with compatible IOPs. The resulting constructions yield SNARKs that are applicable to general arithmetic circuits.

- **Pairing-based cryptography combined with linear Probabilistically Checkable Proofs (PCPs):** in this approach, SNARKs are constructed using bilinear pairings and are based on representations such as Quadratic Arithmetic Programs (QAPs). This method also supports general circuits and is notable for its succinctness and efficient verification.

## III.4 The Sum-Check protocol

The first interactive proof protocol described is the *Sum-Check protocol* [LFKN99]. Suppose there is a $v$-variate polynomial $g$ defined over a finite field $\mathbb{F}_q$. The prover $\mathcal{P}$ wishes to convince the verifier $\mathcal{V}$ that the following sum

$$\sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_v \in \{0,1\}} g(b_1, b_2, \ldots, b_v)$$

is equal to some claimed value $H$.

For instance, if $g$ is a bivariate polynomial, the goal is to verify that

$$H = g(0,0) + g(1,0) + g(0,1) + g(1,1).$$

The Sum-Check protocol provides an efficient interactive method for the verifier to check the correctness of $H$ without evaluating $g$ exhaustively at all points.

Clearly, the verifier $\mathcal{V}$ could compute the sum directly; however, this would require $2^v$ evaluations of $g$, which is computationally infeasible for large $v$. Instead, the Sum-Check protocol enables $\mathcal{V}$ to verify the claim in a number of steps linear in $v$, with polynomial-time computations relative to the number of variables.

For simplicity, assume both $\mathcal{V}$ and the prover $\mathcal{P}$ have oracle access to $g$, meaning that $\mathcal{V}$ can evaluate $g(r_1, \ldots, r_v)$ for any vector $(r_1, \ldots, r_v) \in \mathbb{F}_q^v$ with a single query.

Denote by $\deg_i(g)$ the degree of $g$ with respect to the variable $X_i$.

The protocol proceeds in $v$ rounds as follows:

**Initial step:** Before the interaction begins, the prover $\mathcal{P}$ sends a value $C_1$ to the verifier $\mathcal{V}$, which is claimed to be equal to the sum $H$.

**Round 1:**

1. $\mathcal{P}$ sends to $V$ a univariate polynomial $g_1(X_1)$, which is claimed to satisfy
$$g_1(X_1) = \sum_{(b_2,\ldots,b_v)\in\{0,1\}^{v-1}} g(X_1, b_2, \ldots, b_v).$$
2. $\mathcal{V}$ checks that
$$C_1 \stackrel{?}{=} g_1(0) + g_1(1).$$
3. $\mathcal{V}$ selects a random element $r_1 \in \mathbb{F}_q$ and sends it to $\mathcal{P}$.

**Round $i$, for $1 < i < v$:**

1. $\mathcal{P}$ sends to $\mathcal{V}$ a univariate polynomial $g_i(X_i)$, claimed to satisfy
$$g_i(X_i) = \sum_{(b_{i+1},\ldots,b_v)\in\{0,1\}^{v-i}} g(r_1, \ldots, r_{i-1}, X_i, b_{i+1}, \ldots, b_v).$$
2. $\mathcal{V}$ verifies that
$$g_{i-1}(r_{i-1}) \stackrel{?}{=} g_i(0) + g_i(1).$$
3. $\mathcal{V}$ selects a random element $r_i \in \mathbb{F}_q$ and sends it to $\mathcal{P}$.

**Round $v$:**

1. $\mathcal{P}$ sends to $V$ a univariate polynomial $g_v(X_v)$, claimed to satisfy
$$g_v(X_v) = g(r_1, \ldots, r_{v-1}, X_v).$$
2. $\mathcal{V}$ verifies that
$$g_{v-1}(r_{v-1}) \stackrel{?}{=} g_v(0) + g_v(1).$$
3. $\mathcal{V}$ selects a random element $r_v \in \mathbb{F}_q$.
4. $\mathcal{V}$ queries the oracle to compute $g(r_1, \ldots, r_v)$.
5. $\mathcal{V}$ verifies that
$$g_v(r_v) \stackrel{?}{=} g(r_1, \ldots, r_v).$$
6. If all checks pass, $\mathcal{V}$ accepts; otherwise, $\mathcal{V}$ rejects.

**Theorem III.4.1.** *Let $g$ be a $v$-variate polynomial over a finite field $\mathbb{F}_q$, with degree at most $d$ in each variable. For any specified value $H \in \mathbb{F}_q$, define the language*

$$L = \left\{ g : H = \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_v \in \{0,1\}} g(b_1, \ldots, b_v) \right\}.$$

*The Sum-Check protocol is an interactive proof system for $L$ with perfect completeness and soundness error at most $\frac{vd}{q}$.*

*Proof.* Perfect completeness follows directly from the construction of the protocol: if the prover is honest and $H$ is indeed the sum of evaluations of $g$ over $\{0,1\}^v$, then the verifier accepts with probability 1.

To prove soundness, suppose that the prover is dishonest and tries to convince the verifier that $H$ equals the claimed sum when in fact

$$H \neq \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_v \in \{0,1\}} g(b_1, \ldots, b_v).$$

Then there must exist at least one round $i$ where the prover sends a polynomial $g_i(X_i)$ that does not match

$$s_i(X_i) = \sum_{(b_{i+1}, \ldots, b_v) \in \{0,1\}^{v-i}} g(r_1, \ldots, r_{i-1}, X_i, b_{i+1}, \ldots, b_v),$$

but the verifier's check passes, i.e.,

$$g_i(r_i) = s_i(r_i).$$

By hypothesis, both $g_i$ and $s_i$ are polynomials of degree at most $d$. By the Schwartz–Zippel lemma (III.1.1), the probability that two distinct polynomials of degree at most $d$ agree on a random point $r_i \in \mathbb{F}_q$ is at most $\frac{d}{q}$.

Applying a union bound over the $v$ rounds of the protocol, the total soundness error is at most $\frac{vd}{q}$. □

## Costs of the protocol

Suppose $\deg_i(g) \leq d$ for each $i$, and the protocol consists of $v$ rounds.

**Communication complexity.** The communication consists of $v$ univariate polynomials $g_i$, each of degree at most $d$, plus $v-1$ random field elements. Therefore, the total communication cost is

$$\mathcal{O}(vd).$$

**Prover time complexity.** The prover must compute $v$ polynomials. The $i$-th polynomial is evaluated $2^{v-i}$ times and has at most $d+1$ coefficients. Hence, the total prover running time is

$$\mathcal{O}\big(d \cdot 2^v \cdot T\big),$$

where $T$ is the time required to query the random oracle.

**Verifier time complexity.** The verifier's running time is proportional to the total communication plus the time to query the random oracle, resulting in

$$\mathcal{O}(vd + T).$$

**Remark** It is important to observe that the messages sent by the verifier to the prover consist solely of random elements drawn from the finite field $\mathbb{F}_q$, and are therefore completely independent of the polynomial $g$.

Indeed, the verifier requires only two pieces of information about the polynomial $g$ to carry out its role in the protocol:

- an upper bound on the degree of $g$ with respect to each of its $v$ variables, and
- the capability to evaluate $g$ at a randomly chosen point $\mathbf{r} \xleftarrow{\$} \mathbb{F}_q^v$.

Consequently, the verifier can execute the Sum-Check protocol even without explicit knowledge of the polynomial $g$ to which the protocol is applied.

## Applications

The Sum-Check protocol finds application in various fundamental problems and protocols in theoretical computer science and cryptography, including but not limited to:

1. The proof that #SAT $\in \mathsf{IP}$.
2. Counting the number of triangles in a graph.
3. Constructing highly optimized interactive proof protocols for matrix multiplication.
4. The Goldwasser-Kalai-Rothblum (GKR) protocol.

While the Sum-Check protocol is not directly employed in the construction of zk-STARKs, several of its underlying algebraic ideas — such as low-degree polynomial verification and arithmetization over structured domains — echo throughout the design of STARK components such as FRI and algebraic IOPs. These conceptual parallels will be explored in later chapters to better understand the broader landscape of scalable proof systems.

## A useful subroutine

A simplified version of the Sum-Check protocol has been described, where the verifier $\mathcal{V}$ can evaluate the polynomial with the help of an oracle. In general, such an oracle is not required.

A common scenario involves the evaluation of a multilinear extension $\widetilde{W}$ over $\mathbb{F}_q$ with $v$ variables at just two points, say $b, c \in \mathbb{F}_q^v$.

The following interactive protocol reduces the problem of evaluating $\widetilde{W}(b)$ and $\widetilde{W}(c)$ to evaluate $\widetilde{W}(r)$ at a single point $r \in \mathbb{F}_q^v$.

Let $l(t)$ be the parametric line defined such that $l(0) = b$ and $l(1) = c$.

1. The prover $\mathcal{P}$ sends to the verifier $\mathcal{V}$ a univariate polynomial $q(t)$ of degree at most $v$, defined as
$$q(t) = \widetilde{W}(l(t)).$$
2. The verifier $\mathcal{V}$ interprets $q(0)$ and $q(1)$ as $\widetilde{W}(b)$ and $\widetilde{W}(c)$, respectively. Then, $\mathcal{V}$ selects a random $r^* \in \mathbb{F}_q$ and sets
$$r = l(r^*).$$

3. Finally, the verifier $\mathcal{V}$ regards $q(r^*)$ as the evaluation $\widetilde{W}(r)$.

**Example III.4.1.** Let $v = 2$, $b = (2, 4)$ and $c = (3, 2)$. Consider the multilinear extension $\widetilde{W}$ defined as

$$\widetilde{W}(x_1, x_2) = 3x_1x_2 + 2x_2.$$

The parametric line $l(t)$ connecting $b$ and $c$ is given by

$$l(t) = (t + 2,\ 4 - 2t).$$

The prover computes the univariate polynomial

$$q(t) = \widetilde{W}(l(t)) = 3(t + 2)(4 - 2t) + 2(4 - 2t) = -6t^2 - 4t + 32.$$

The verifier interprets

$$q(0) = \widetilde{W}(b) = 32, \quad q(1) = \widetilde{W}(c) = 22.$$

Finally, for a uniformly random $r \in \mathbb{F}_q$, the point

$$l(r) = (r + 2,\ 4 - 2r)$$

lies on the line $l$, and the verifier interprets $q(r) = \widetilde{W}(l(r))$.

**Theorem III.4.2.** *Let $\widetilde{W}$ be a multilinear extension (MLE) over $\mathbb{F}_q$ with $v$ variables. The protocol described above achieves perfect completeness and has soundness error at most $\frac{v}{q}$.*

*Proof.* Perfect completeness follows directly from the correctness of the protocol: if the prover $\mathcal{P}$ follows the protocol honestly and sends $q(t) = \widetilde{W}(l(t))$, then all of the verifier's $\mathcal{V}$ checks will succeed.

As for soundness, assume that the prover sends a polynomial $q(t)$ such that $q(t) \neq \widetilde{W}(l(t))$, attempting to cheat. Since both $q(t)$ and $\widetilde{W}(l(t))$ are univariate polynomials of degree at most $v$, they can agree on at most $v$ points in $\mathbb{F}_q$ unless they are equal.

Therefore, by the Schwartz-Zippel Lemma (III.1.1), the probability that the verifier's random check at a randomly chosen point $r^* \in \mathbb{F}_q$ passes despite $q(t) \neq \widetilde{W}(l(t))$ is at most $\frac{v}{q}$. □

## III.5 The GKR Protocol

A notable limitation of the Sum-Check protocol is that, in its general form, the verifier $\mathcal{V}$ runs in polynomial time. A natural goal is to design a general-purpose interactive proof system in which the verifier runs in *linear time*, leveraging interaction with the prover to achieve sublinear verification. At the same time, the prover $\mathcal{P}$ is expected to run in polynomial time.

Goldwasser, Kalai, and Rothblum proposed such a protocol in their article [GKR15]. Their interactive proof system achieves these objectives and is best understood in the context of the *arithmetic circuit evaluation problem*. The protocol is general-purpose in the sense that any problem in the complexity class $\mathcal{P}$ can be efficiently reduced to an instance of arithmetic circuit evaluation. This makes the protocol particularly powerful and applicable in a wide range of computational settings.

**Protocol setting.** Let $\mathcal{C}$ be a structured arithmetic circuit over a finite field $\mathbb{F}_q$, with fan-in 2 and depth $d$. The circuit is assumed to be log-space uniform.

**Definition III.5.1.** A circuit $\mathcal{C}$ is log-space uniform if there exists a deterministic Turing machine that, on input a gate label $g$, computes all relevant information about gate $g$ (e.g., gate type, input labels) using $\mathcal{O}(\log |g|)$ space.

Let:

- $|\mathcal{C}|$ be the total number of gates in the circuit;
- $d$ be the depth of the circuit, where layer $d$ contains the inputs and layer 0 the outputs;
- $S_i = 2^{k_i}$ be the number of gates in layer $i$;
- $(x_1, \ldots, x_n) \in \mathbb{F}_q^n$ be the input vector.

The protocol proceeds in $d$ rounds, one for each layer, where the goal of round $i$ is to reduce a claim about the values of the gates in layer $i$ to a claim about the values in layer $i + 1$.

To describe the protocol formally, we define the following functions:

- $W_i : \{0,1\}^{k_i} \to \mathbb{F}_q$, mapping the binary label of a gate in layer $i$ to its value;

- $\mathtt{in}_{1,i}, \mathtt{in}_{2,i} : \{0,1\}^{k_i} \to \{0,1\}^{k_{i+1}}$, returning the labels of the input gates of a given gate;
- $\mathtt{add}_i : \{0,1\}^{k_i+2k_{i+1}} \to \{0,1\}$, which returns 1 if the tuple corresponds to an addition gate with correct input labels;
- $\mathtt{mult}_i : \{0,1\}^{k_i+2k_{i+1}} \to \{0,1\}$, defined analogously for multiplication gates.

Let $\widetilde{W}_i$, $\widetilde{\mathtt{add}}_i$, and $\widetilde{\mathtt{mult}}_i$ be the multilinear extensions of the above functions.

**Protocol: Round 0.**

1. Let $D : \{0,1\}^{k_0} \to \mathbb{F}_q$ be a function that maps each output gate label to the claimed output value. The prover $\mathcal{P}$ sends $D$ to the verifier $\mathcal{V}$.
2. $\mathcal{V}$ samples a random $r_0 \leftarrow \mathbb{F}_q^{k_0}$, and computes $m_0 \leftarrow D(r_0)$. The goal is now to verify that $\widetilde{W}_0(r_0) = m_0$.

To achieve this, the protocol uses the following identity:

**Lemma III.5.2.** *For every $z \in \{0,1\}^{k_i}$,*

$$\widetilde{W}_i(z) = \sum_{b,c \in \{0,1\}^{k_{i+1}}} [\widetilde{\mathtt{add}}_i(z,b,c) \cdot (\widetilde{W}_{i+1}(b) + \widetilde{W}_{i+1}(c)) + \\ + \widetilde{\mathtt{mult}}_i(z,b,c) \cdot \widetilde{W}_{i+1}(b) \cdot \widetilde{W}_{i+1}(c)].$$

Let $f_i^{(r_i)}(b,c)$ denote the polynomial:

$$f_i^{(r_i)}(b,c) := \widetilde{\mathtt{add}}_i(r_i,b,c) \cdot (\widetilde{W}_{i+1}(b) + \widetilde{W}_{i+1}(c)) + \widetilde{\mathtt{mult}}_i(r_i,b,c) \cdot \widetilde{W}_{i+1}(b) \cdot \widetilde{W}_{i+1}(c).$$

**Protocol: Round $i$, $0 \le i < d$.**

1. $\mathcal{P}$ claims that $m_i = \sum_{b,c \in \{0,1\}^{k_{i+1}}} f_i^{(r_i)}(b,c)$;
2. $\mathcal{P}$ and $\mathcal{V}$ invoke the Sum-Check protocol on $f_i^{(r_i)}$;
3. At the end of the Sum-Check protocol, $\mathcal{V}$ must evaluate $f_i^{(r_i)}$ at a random point $(b^*, c^*) \in \mathbb{F}_q^{k_{i+1}} \times \mathbb{F}_q^{k_{i+1}}$;

4. To do so, define a line $\ell(t)$ such that $\ell(0) = b^*$ and $\ell(1) = c^*$. The prover sends the univariate polynomial $q(t) := \widetilde{W}_{i+1}(\ell(t))$;
5. $\mathcal{V}$ picks $r^* \leftarrow \mathbb{F}_q$, computes $r_{i+1} := \ell(r^*)$, and sets $m_{i+1} := q(r^*)$, which is claimed to equal $\widetilde{W}_{i+1}(r_{i+1})$.

**Final step.** At the last layer $d$, $\mathcal{V}$ directly evaluates $\widetilde{W}_d(r_d)$ from the known input vector and checks whether it equals $m_d$.

## Costs of the protocol

Suppose the circuit $\mathcal{C}$ has depth $d$, size $|\mathcal{C}|$, and is log-space uniform.

**Communication complexity.** The protocol proceeds in $d$ rounds, each invoking a Sum-Check protocol on a polynomial of arity $2k_{i+1}$, where $S_{i+1} = 2^{k_{i+1}}$ is the number of gates at layer $i+1$. Each round incurs a communication cost of $\mathcal{O}(k_{i+1})$, and hence the total communication is

$$\mathcal{O}(d \log_2 |\mathcal{C}|).$$

**Prover time complexity.** The prover $\mathcal{P}$ must simulate the Sum-Check protocol and evaluate intermediate multilinear extensions of functions like $\widetilde{W}_i$, $\widetilde{\mathsf{add}}_i$, and $\widetilde{\mathsf{mult}}_i$. In the original version of the protocol, the total prover time is

$$\mathcal{O}(|\mathcal{C}|^3),$$

though later refinements reduce this to quasilinear time in $|\mathcal{C}|$.

**Verifier time complexity.** The verifier $\mathcal{V}$ only needs to read the input and evaluate or access random positions in the circuit using log-space uniformity. Thus, its running time is

$$\mathcal{O}(n + d \log_2 |\mathcal{C}|),$$

where $n$ is the input length.
Crucially, $\mathcal{V}$ does not need to read the full description of the circuit $\mathcal{C}$.

## Soundness of the protocol

The GKR protocol achieves *perfect completeness* and has soundness error

$$\mathcal{O}\left(\frac{d\log_2|\mathcal{C}|}{q}\right).$$

**Proof sketch.** Suppose that $\mathcal{P}$ sends a false claim for the output of the circuit. In order to convince the verifier $\mathcal{V}$, it must happen that at some round $j$, the prover sends a polynomial $g_j \neq s_j$, where $s_j$ is the correct polynomial, and yet

$$g_j(r_j) = s_j(r_j),$$

for some randomly chosen $r_j \in \mathbb{F}_q$.

There are two possibilities to consider:

1. **During a Sum-check round:** The Sum-Check protocol is applied to polynomials of degree $\mathcal{O}(1)$, so by the Schwartz-Zippel Lemma (III.1.1), the probability that $g_j(r_j) = s_j(r_j)$ while $g_j \neq s_j$ is at most $\mathcal{O}\left(\frac{1}{q}\right)$.

2. **During a subroutine check (line evaluation):** Here, the verifier checks equality of polynomials of degree at most $\mathcal{O}(\log_2|\mathcal{C}|)$. Again, by the Schwartz-Zippel Lemma, the probability that two distinct polynomials of such degree coincide on a random point is at most $\mathcal{O}\left(\frac{\log_2|\mathcal{C}|}{q}\right)$.

Since the protocol consists of $d$ iterations, and each iteration contains a constant number of such checks, applying the union bound yields the overall soundness error

$$\mathcal{O}\left(\frac{d\log_2|\mathcal{C}|}{q}\right).$$

## The GKR protocol: an example

Consider the following arithmetic circuit defined over $\mathbb{F}_5$ (Fig.III.1). The goal is to use the GKR protocol to verify whether the output of the circuit is the vector $(4, 2)$.

At the beginning of the protocol, the prover $\mathcal{P}$ sends to the verifier $\mathcal{V}$ a function $D : \{0,1\} \to \mathbb{F}_5$ that is claimed to be equal to $\widetilde{W}_0$, the MLE of the output layer of the circuit. In particular, $D(0) = 4$ and $D(1) = 2$.

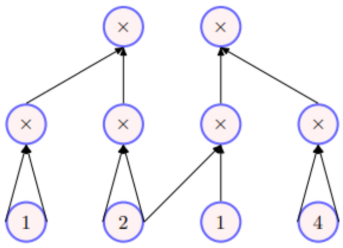



Figure III.1: arithmetic circuit defined over $\mathbb{F}_5$

**MLE of the output layer.** The verifier computes the multilinear extension:

$$\widetilde{D}(x) = D(0)(1-x) + D(1)x = -2x + 4 = 3x + 4.$$

Then, $\mathcal{V}$ chooses a random value $r_0 \leftarrow \mathbb{F}_5$, say $r_0 = 4$, and computes $m_0 = \widetilde{D}(r_0) = 1$. The rest of the protocol will check whether $\widetilde{W_0}(r_0) = m_0$.

### Round 0

The required multilinear extension is:

$$\widetilde{\text{mult}}_0(x_1, \ldots, x_5) = (1-x_1)(1-x_2)(1-x_3)(1-x_4)x_5 + x_1x_2(1-x_3)x_4x_5.$$

The prover claims that

$$\widetilde{W_1}(x_1, x_2) = -4x_1x_2 + x_1 + 3x_2 + 1.$$

Then, the Sum-Check protocol is applied to verify the equality:

$$m_0 = \sum_{b,c \in \{0,1\}^2} \widetilde{\text{mult}}_0(r_0, b, c) \cdot \widetilde{W_1}(b) \cdot \widetilde{W_1}(c).$$

In this example, the verifier is convinced because:

$$1 = (1-r_0)\left(\widetilde{W_1}(0,0) \cdot \widetilde{W_1}(0,1)\right) + r_0\left(\widetilde{W_1}(1,0) \cdot \widetilde{W_1}(1,1)\right) =$$

$$= -3(1 \cdot 4) + 4(2 \cdot 1) = -4 = 1 \mod 5.$$

However, $\mathcal{V}$ cannot compute $\widetilde{W_1}$ values directly. To complete the check, $\mathcal{V}$ selects a random point $(b^*, c^*) \leftarrow \mathbb{F}_5^2 \times \mathbb{F}_5^2$, e.g., $((1,2),(2,3))$, and defines the line:

$$\ell(t) = (1+t, 2+t).$$

$\mathcal{P}$ sends the univariate polynomial $q(t) = \widetilde{W_1}(\ell(t))$, which in this case is:

$$q(t) = t^2 + 2t.$$

$\mathcal{V}$ now completes the Sum-check using $q(0)$ and $q(1)$, then chooses randomly $r^* \leftarrow \mathbb{F}_5$, say $r^* = 2$, and sets:

$$r_1 = \ell(r^*) = (3,4), \quad m_1 = q(r^*) = 3.$$

**Round 1**

$\mathcal{V}$ now verifies that $m_1 = \widetilde{W_1}(r_1)$. For this, the next round of the GKR protocol is applied.

The relevant multilinear extension is:

$$\widetilde{\text{mult}_1}(x_1, \ldots, x_6) = (1-x_1)(1-x_2)(1-x_3)(1-x_4)(1-x_5)(1-x_6) +$$
$$+(1-x_1)x_2(1-x_3)x_4(1-x_5)x_6 + x_1(1-x_2)(1-x_3)x_4x_5(1-x_6) +$$
$$+x_1x_2x_3x_4x_5x_6.$$

The prover claims:

$$\widetilde{W_2}(x_1, x_2) = 2x_1x_2 + x_2 + 1.$$

The verifier checks:

$$m_1 = \sum_{b,c \in \{0,1\}^2} \widetilde{\text{mult}_1}(r_1, b, c) \left( \widetilde{W_2}(b) \cdot \widetilde{W_2}(c) \right).$$

Indeed:

$$m_1 = (1-r_1(0))(1-r_1(1)) \cdot \widetilde{W_2}(0,0)^2 + (1-r_1(0))r_1(1) \cdot \widetilde{W_2}(0,1)^2 +$$
$$+r_1(0)(1-r_1(1)) \cdot \widetilde{W_2}(1,0)\widetilde{W_2}(0,1) + r_1(0)r_1(1) \cdot \widetilde{W_2}(1,1)^2 = 3 \mod 5.$$

Then, $\mathcal{V}$ samples a random point $(b^*, c^*) \leftarrow \mathbb{F}_5^2 \times \mathbb{F}_5^2$, say $((3,1),(4,0))$, and defines:

$$\ell_1(t) = (3+t, 4-t).$$

$\mathcal{P}$ sends the polynomial $q_1(t) = \widetilde{W_2}(\ell_1(t)) = 3t^2 + t + 4$. $\mathcal{V}$ evaluates $q_1(0)$ and $q_1(1)$, samples $r^* = 3$, and sets:

$$r_2 = \ell_1(r^*) = (1,1), \quad m_2 = q_1(r^*) = 4.$$

**Final check**

Finally, the verifier checks:

$$m_2 = \widetilde{W}_2(r_2) = 2 \cdot 1 \cdot 1 + 1 + 1 = 4.$$

The check passes, and the verifier is convinced that the output of the circuit is indeed $(4, 2)$.

## III.6 Polynomial commitments

A fundamental problem in the study of arithmetic circuits is the *circuit satisfiability problem*. Given an arithmetic circuit $\mathcal{C}$ that takes as input a public value $x \in \mathbb{F}^n$ and a private witness $w \in \mathbb{F}^m$ (known only to the prover $\mathcal{P}$), the goal of the prover is to convince the verifier $\mathcal{V}$ that there exists a witness $w$ such that

$$\mathcal{C}(x, w) = y,$$

where $y \in \mathbb{F}^\ell$ is a publicly known output.

By combining the GKR protocol with a *polynomial commitment scheme*, one can construct efficient interactive arguments for this satisfiability relation [ZGKPP17]. In these constructions, the prover commits to certain low-degree polynomials that encode the structure and evaluation of the circuit, and then uses the GKR protocol to verify correct evaluation layer by layer.

These arguments are particularly powerful when they satisfy the property of *knowledge soundness*. This means that, if the prover successfully convinces the verifier, then there exists an efficient extractor that can recover a valid witness $w$ such that $C(x, w) = y$. In other words, the prover does not merely convince the verifier of the existence of such a witness—it must actually *know* one.

### The Naive Approach

A naive solution to the circuit satisfiability problem would have the prover $\mathcal{P}$ send the full witness $\pi = w$ to the verifier $\mathcal{V}$. Given a circuit $\mathcal{C}$, public input $x$, and public output $y$, the verifier would then directly check whether $\mathcal{C}(x, w) = y$. However, this approach suffers from several significant drawbacks:

- **Lack of succinctness:** the prover must send the entire witness $w$, which could be large;
- **High verification cost:** the verifier may require a large amount of time to evaluate $\mathcal{C}(x, w)$;
- **No zero-knowledge:** revealing the witness $w$ leaks private information.

These limitations motivate the need for *commitment schemes*. In the context of the GKR protocol, this need becomes even more evident: the verifier $\mathcal{V}$ does not require access to the full witness. Instead, during the protocol, the verifier only needs to verify a single value $\widetilde{w}(r)$, where $\widetilde{w}$ denotes the multilinear extension of the witness and $r \in \mathbb{F}^n$ is a randomly chosen point.

To maintain succinctness and privacy, the prover $\mathcal{P}$ can commit to $\widetilde{w}$ at the beginning of the protocol using a *polynomial commitment scheme*. The parties then proceed with the GKR protocol as usual. Only at the end of the protocol does the verifier need to verify that $\widetilde{w}(r)$ is consistent with the committed polynomial. This is done by having the prover open the commitment at the point $r$, producing a value and a short proof.

**Definition III.6.1.** A **polynomial commitment scheme** over a field $\mathbb{F}$ is a tuple of four probabilistic polynomial-time algorithms

$$(\mathsf{Setup}, \mathsf{Commit}, \mathsf{CreateWitness}, \mathsf{VerifyEval})$$

satisfying the following properties:

1. $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, t)$: takes a security parameter $\lambda \in \mathbb{N}$ (in unary) and a degree bound $t$, and returns a commitment key $\mathsf{ck}$ usable for polynomials of degree at most $t$;
2. $(\mathsf{com}, \mathsf{o}) \leftarrow \mathsf{Commit}(\mathsf{ck}, f(X))$: takes as input a polynomial $f(X) \in \mathbb{F}[X]$ and returns a commitment $\mathsf{com}$ and opening material $\mathsf{o}$;
3. $(v, \pi_z) \leftarrow \mathsf{CreateWitness}(\mathsf{ck}, f(X), z, \mathsf{o})$: takes a point $z \in \mathbb{F}$, the opening material $\mathsf{o}$, and the polynomial $f$, returning the evaluation $v = f(z)$ and a proof $\pi_z$;

4. $\{0,1\} \leftarrow \mathsf{VerifyEval}(\mathsf{ck}, \mathsf{com}, z, v, \pi_z)$: verifies that $v = f(z)$ for the committed polynomial. This algorithm should run in sublinear time with respect to the degree $t$.

As usual, a polynomial commitment must satisfy three fundamental properties: *correctness*, *hiding*, and *binding*. However, in many applications, a stronger notion of binding is desirable. This stronger property is known as *extractability*.

Extractability strengthens the binding property by ensuring that the prover $\mathcal{P}$ is not only committed to a single value at a single point, but actually to a unique low-degree polynomial $p \in \mathbb{F}[X]$ that explains all of its answers to evaluation queries. In particular, if $\mathcal{P}$ is able to provide multiple correct answers to different evaluation points, then extractability ensures the existence of an efficient algorithm that can reconstruct a polynomial consistent with all of these answers.

Informal definitions of extractability and security are given below.

**Definition III.6.2.** A polynomial commitment scheme is *extractable* if, for any probabilistic polynomial-time adversary that outputs a commitment $\mathsf{com}$, there exists an efficient extractor $\mathcal{E}$ that outputs a polynomial $p(X) \in \mathbb{F}[X]$ such that all evaluation proofs produced by the adversary are consistent with $p$.

**Definition III.6.3.** A polynomial commitment scheme is *secure* if it satisfies the following properties:

- **Correctness:** if $\mathsf{com}$ is a commitment to $f(X)$, then

$$\mathsf{VerifyEval}(\mathsf{ck}, \mathsf{com}, z, f(z), \pi_z) = 1$$

  for all $z \in \mathbb{F}$;

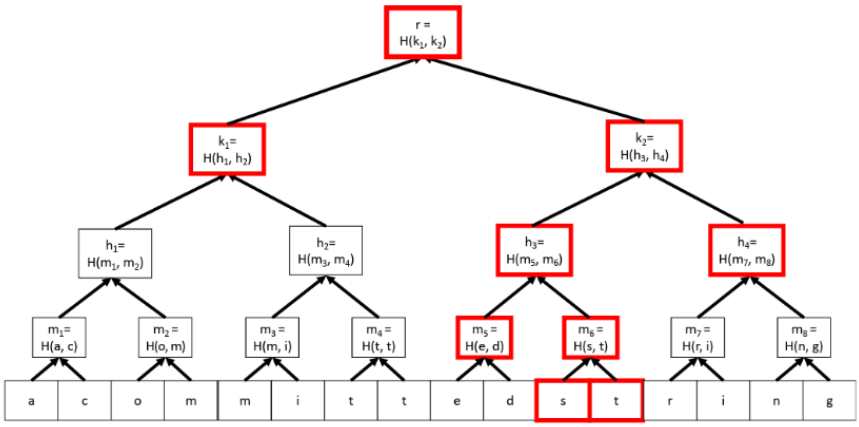
- **Evaluation Binding:** it is infeasible for an adversary to produce two valid openings $(v, \pi_z) \neq (v', \pi'_z)$ at the same point $z$;

- **Hiding:** given access to at most $t$ evaluations of $f$, an adversary cannot predict $f(z)$ for a new point $z \notin \{z_1, \ldots, z_t\}$.

**Historical Note.** An instructive (but inefficient) commitment scheme was proposed by Kalai in 2017 (during a workshop talk). This construction is a *relaxed polynomial commitment scheme* that binds the prover only to a function that is *close* to a low-degree polynomial. The scheme is based on Merkle trees and low-degree testing. The starting idea is to construct a commitment by evaluating the polynomial at various points, placing these evaluations as leaves of a Merkle tree, and then using the root as the commitment.

**Note.** A Merkle tree is a binary tree of depth $\log_2(k)$, where the $k$ inputs are placed at the leaves, and the value of each internal node is computed as the hash of the concatenation of its two children's values. The algorithm stores only the final hash, called the *root* of the tree. Let $\mathsf{MT}_k$ denote the algorithm that takes $k$ inputs and outputs the root of the corresponding Merkle tree. If $(\mathsf{Gen}, \mathsf{H})$ is a collision-resistant hash function family, then $(\mathsf{Gen}, \mathsf{MT}_k)$ is also collision-resistant.

To prove that a value is indeed one of the leaves of the tree, the committer sends the hash of its sibling node along with $\log_2(k) - 1$ other hashes, one for each level of the Merkle tree. The verifier can then recompute the root hash using these values. Hence, the communication complexity of this proof is $\mathcal{O}(\log_2(k))$.



### III.6.1 LDE: Low Degree Testing

To reveal $w(r)$, the prover $\mathcal{P}$ exploits the properties of the Merkle tree and simply sends $w(r)$ along with all the hashes required for path authentication.

**Issue** There is no guarantee that the committed string contains all evaluations of some multilinear polynomial — it may be an arbitrary function.

**Solution** A low degree test is employed to ensure that the committed string is close to the evaluations of a low-degree polynomial. This is achieved by inspecting only a small number of entries in the string.

Suppose a receiver is given oracle access to a large string $s$, which is claimed to contain all evaluations of an $m$-variate function over a finite field $\mathbb{F}_q$ (note that the total number of possible inputs is $q^m$).

A low degree test inspects a small fraction of $s$ to verify that these entries are consistent with a low-degree polynomial; in other words, it guarantees that $s$ and the low-degree polynomial are close in Hamming distance.

An important low-degree test is the *point-versus-line test* [Aro03]. The idea is to select a line $\ell \in \mathbb{F}_q^m$ and check whether $s$ restricted to $\ell$ is consistent with a univariate polynomial of degree at most $m$. This procedure is the same subroutine used in the GKR protocol.

It has been shown that if the test passes with probability $\alpha$, then there exists a low-degree polynomial that agrees with $s$ on approximately an $\alpha$ fraction of points.

Let $s$ be the string containing all $q^v$ evaluations of a $v$-variate multilinear extension (MLE) $w_e$, defined over a finite field $\mathbb{F}_q$.

The string $s$ is committed using a Merkle tree, after which the low-degree test is applied. The verifier chooses a random line $\ell \in \mathbb{F}_q^v$, and the prover must reveal the values at all points along the line. Finally, the verifier checks if these values are consistent with a univariate polynomial of degree at most $v$.

The committed MLE $w_e$ is computed by $\mathcal{P}$ starting from the knowledge of the witness $w$ used in the GKR protocol.

**Theorem III.6.4.** *The relaxed polynomial commitment scheme based on Merkle trees and low-degree tests is extractable.*

## Protocol Costs

The protocol consists of the GKR protocol combined with Merkle tree commitments and low-degree testing.

**Rounds.** The number of rounds is

$$\mathcal{O}(d \log |C|),$$

which matches the number of rounds required by the original GKR protocol.

**Communication complexity.** The communication consists of the messages exchanged in the GKR protocol, which amount to $\mathcal{O}(d \log |C|)$, plus the additional data required to send the Merkle commitment and perform the low-degree test. The latter dominates the overall cost, resulting in a total communication complexity of

$$\mathcal{O}(q \log^2 n),$$

where $n$ is the length of the witness and $q = \mathcal{O}(n)$ is the size of the field.

**Prover time complexity.** The prover's time is dominated by the cost of committing to the multilinear extension $w_e$ via a Merkle tree. This requires super-polynomial time in $n$, specifically

$$\mathcal{O}\left(n^{\log n}\right),$$

where $n = 2^v$ and $v$ is the number of variables in the MLE. As a consequence, this commitment scheme is not yet efficient for practical purposes, in contrast to SNARK systems which typically achieve near-linear prover time.

**Verifier time complexity.** The verifier's running time is

$$\mathcal{O}(n + d \log |C|),$$

which includes the time to verify the GKR proof and to open the Merkle-based polynomial commitment.

**Remark.** This polynomial commitment scheme can be replaced with any other more efficient polynomial commitment protocol.

# III.7 FRI

**Recap.** Looking at the scheme proposed by Kalai, a key issue arises: there is no guarantee that the committed string actually corresponds to the evaluations of a multilinear (or low-degree) polynomial—it may in fact represent an arbitrary function. To address this problem, a low-degree test is employed to ensure that the committed string is close to the evaluation table of a low-degree polynomial. Remarkably, this can be verified by inspecting only a small number of entries in the string. It has been proven that if the test is passed with probability $\varepsilon$, then there exists a low-degree polynomial that agrees with the committed string on approximately an $\varepsilon$ fraction of the evaluation points.

The resulting protocol constitutes a **polynomial commitment scheme**; however, it is not considered practical due to two main limitations.

1. The number of leaves involved is equal to $q$, the cardinality of the underlying finite field, resulting in a commitment time of $\mathcal{O}(q)$. Ideally, the computational complexity should scale with the degree bound $d$ of the committed polynomial, rather than the size of the field.

2. In the absence of a **proximity test**, the verifier lacks any guarantee regarding the degree of the committed polynomial, and cannot even be certain that the commitment corresponds to a polynomial at all.

**FRI (Fast Reed-Solomon IOP of Proximity)** [BBHR18a] is designed to address both of these issues.

## FRI: Addressing the First Issue

To address the first limitation of the basic polynomial commitment scheme (namely, the inefficiency due to committing over the entire field $\mathbb{F}_q$), FRI introduces a more efficient strategy. The idea is to Merkle-commit to the evaluations $f(x)$ of a polynomial $f$ only over a carefully selected subset $\Omega \subseteq \mathbb{F}_q$, rather than over the full field.

This subset $\Omega$ is chosen such that its size is $n = \rho^{-1} \cdot d$, where $d$ is a bound on the degree of the polynomial and $0 < \rho < 1$. In practice, $\rho$ is often selected to be close to $\frac{1}{2}$. The quantity $\rho^{-1}$ is commonly referred to as the *FRI blowup factor*, while $\rho$ represents the rate of the Reed–Solomon code employed in the protocol.

The choice of $\rho$ introduces a fundamental trade-off in FRI between prover efficiency and verifier cost. Specifically, a larger blowup factor $\rho^{-1}$ results in increased computation for the prover but leads to reduced verification time and lower query complexity. Conversely, a smaller blowup factor benefits the prover at the cost of higher verifier effort.

For efficiency purposes, the evaluation domain $\Omega$ is chosen as the set of all $n$-th roots of unity in $\mathbb{F}_q$, where $n = \rho^{-1} \cdot d$ is a power of two. This choice provides algebraic structure and enables fast computation via the Fast Fourier Transform (FFT).

The set

$$\Omega = \{x \in \mathbb{F}_q | x^n = 1 \ mod \ q\} = \{x, x^2, \ldots, x^n\}$$

has the following properties:

- $\Omega$ is a multiplicative subgroup of $\mathbb{F}_q$: if $x, y \in \Omega$, then $x \cdot y \in \Omega$.
- Since $n$ is even, for every $x \in \Omega$, it holds that $x^2$ is an $\frac{n}{2}$-th root of unity.
- Again, since $n$ is even, if x is a $n$-th root of unity ,$-x$ is also in $\Omega$.
- The set $\Omega$ has size exactly $n$ if and only if $n$ divides $q - 1$.

A commonly used field in practical implementations of FRI is the so-called *Goldilocks field*, with cardinality $q = 2^{64} - 2^{32} + 1$. This field admits a rich subgroup structure and supports efficient FFT-based operations.

The commitment phase in FRI operates over this domain and requires $\mathcal{O}(\rho^{-1} \cdot d)$ time, which is quasi-linear in the degree bound $d$.

## FRI: Addressing the Second Issue

Regarding the second issue, the verifier must be assured that the committed vector corresponds exactly to the evaluations of some polynomial of degree at most $d$ over the domain $\Omega$.

Previous approaches, such as the point-versus-line test (III.6.1), attempted to verify low-degree properties by inspecting only a few entries of the vector. However, these methods have proven to be impractical in realistic settings.

In contrast, the FRI low-degree test is an **interactive** protocol composed of two distinct phases. The first is the *folding phase* (also known as the

commitment phase), which proceeds in $\log_2(d)$ rounds. The second is the *query phase*, which consists of a single round of interaction.

### III.7.1 Folding phase

During the folding phase, the prover iteratively folds the committed vector by halving its length at each step. The verifier $\mathcal{V}$ samples a random element $r \in \mathbb{F}_q$ and sends it to the prover $\mathcal{P}$, who then uses this value to combine pairs of entries in the vector in a randomized linear manner. This folding operation ensures that the resulting vector corresponds to a polynomial of degree at most half the degree of the original polynomial.

The folding process is repeated until the polynomial degree is reduced to zero. Nonetheless, the length of the folded vector remains $\rho^{-1}$.

More formally, given a polynomial $f(X)$, it can be decomposed into its even and odd parts as follows:

$$f(X) = f_e(X^2) + Xf_o(X^2).$$

The folding step aims to combine these two parts into a single polynomial of lower degree by leveraging the random scalar $r$.

So, the verifier selects a random field element $r \in \mathbb{F}_q$ and sends it to the prover. The folding polynomial is defined as

$$f_{\text{fold}}(z) = f_e(z) + rf_o(z), \tag{III.1}$$

where $f_e$ and $f_o$ are the even and odd parts of $f$, respectively. This polynomial clearly has degree at most half that of $f$, since both $f_e$ and $f_o$ have degree at most $\frac{\deg(f)}{2}$.

Let $x$ and $-x$ be $n$-th roots of unity, and define $z = x^2$. Then, by interpolation, the folding polynomial satisfies

$$f_{\text{fold}}(z) = \frac{r+x}{2x}f(x) + \frac{r-x}{-2x}f(-x). \tag{III.2}$$

This identity holds because: by definition, $f(x) = f_e(z) + xf_o(z)$. And, observe that, if $r = x$, then $f_{\text{fold}}(z) = f(x)$, and if $r = -x$, then $f_{\text{fold}}(z) = f(-x)$. Equation (9) has been specifically defined to ensure the desired behavior when the verifier's random challenge $r$ coincides with either $x$ or $-x$. In particular, it satisfies

$$f_{\text{fold}}(z) = f(x) \quad \text{if } r = x, \quad \text{and} \quad f_{\text{fold}}(z) = f(-x) \quad \text{if } r = -x.$$

Since Equation III.2 is a linear function in $r$, and two degree-one polynomials that agree on at least two distinct values must be identical, it follows that Equation III.2 represents the unique valid folding expression for all values of $r \in \mathbb{F}_q$.

An essential property used in this construction is that the map $x \mapsto x^2$ is 2-to-1 over the evaluation domain $\Omega$, as long as $|\Omega| = n$ is a power of two. This is what guarantees that, at each round of folding, the size of the domain is halved.

This property also motivates the specific choice of $\Omega$. In fact, more naive domains such as $\{0, 1, \ldots, n-1\}$ do not satisfy the necessary algebraic closure under squaring and therefore cannot support the FRI folding process correctly.

### III.7.2 Query Phase

The query phase is performed after the completion of the folding phase. During the folding process, a dishonest prover may attempt to cheat by providing incorrect folded vectors. The purpose of the query phase is to allow the verifier to detect such inconsistencies with high probability.

To do so, the verifier samples approximately $\frac{\lambda}{log_2(\rho^{-1})}$ entries of each folded vector, where $\lambda$ is the desired security parameter, and checks whether the value is consistent with the corresponding linear combination of two entries from the previous layer of the folding process.

### Costs

There are $\log_2(d)$ rounds of folding (for a degree bound $d$) and for each round, the verifier performs $\frac{\lambda}{log_2(\rho^{-1})}$ queries. For every query, a Merkle authentication path is provided, whose length is $\log_2 d$, due to the depth of the Merkle tree. As a result, the total proof length and the verifier's runtime are both $\mathcal{O}(\log_2^2(d))$.

Finally, the interaction in the query phase can be removed using the Fiat–Shamir heuristic, yielding a fully non-interactive version of the protocol in the random oracle model.

**Example III.7.1.** Let $\mathbb{F}_{19}$ denote the underlying finite field.

Choose $\theta = 2 \in \mathbb{F}_{19}$, which has multiplicative order 9. The multiplicative subgroup generated by $\theta$ is

$$\Omega = \langle \theta \rangle = \{1,\ 2,\ 4,\ 8,\ 16,\ 13,\ 7,\ 14,\ 9\}.$$

Let the polynomial to be committed be

$$f(x) = 2x^2 + 4x + 3.$$

We evaluate $f$ over all elements of the domain $\Omega$:

$$\begin{aligned}
&f(1) = 2(1)^2 + 4(1) + 3 = 9,\\
&f(2) = 2(4) + 8 + 3 = 19 \equiv 0 \pmod{19},\\
&f(4) = 2(16) + 16 + 3 = 51 \equiv 13 \pmod{19},\\
&f(8) = 2(64) + 32 + 3 = 163 \equiv 11 \pmod{19},\\
&f(16) = 2(256) + 64 + 3 = 579 \equiv 10 \pmod{19},\\
&f(13) = 2(169) + 52 + 3 = 393 \equiv 13 \pmod{19},\\
&f(7) = 2(49) + 28 + 3 = 129 \equiv 15 \pmod{19},\\
&f(14) = 2(196) + 56 + 3 = 451 \equiv 14 \pmod{19},\\
&f(9) = 2(81) + 36 + 3 = 201 \equiv 11 \pmod{19}.
\end{aligned}$$

Thus, the evaluation vector is given by

$$f(\Omega) = [9,\ 0,\ 13,\ 11,\ 10,\ 13,\ 15,\ 14,\ 11].$$

**Commitment Phase** A Merkle tree is constructed over the vector of evaluations. Each leaf of the tree corresponds to a field element $x \in \Omega$ and its evaluation $f(x)$. These values are hashed and recursively combined to form the Merkle root, which serves as the commitment to the polynomial.

**Folding Phase** Assume that the verifier selects a random challenge $r = 6 \in \mathbb{F}_{19}$. The prover computes a folded polynomial over the domain

$\Omega^{(1)} := \{x^2 \mid x \in \Omega\}$, which has size $|\Omega|/2$ due to the 2-to-1 property of the squaring map on $\Omega$.

The pairing for folding proceeds as:

$$(1, 16), \quad (2, 13), \quad (4, 7), \quad (8, 14).$$

The folding rule is given by:

$$f_{\text{fold}}(z) = \frac{r + x}{2x} f(x) + \frac{r - x}{-2x} f(-x), \tag{III.3}$$

where $z = x^2$ and $-x$ denotes the multiplicative inverse of $x$ within the subgroup.

Let us compute an example with $x = 1$ and $-x = 16$. Given $f(1) = 9$ and $f(16) = 10$:

$$f_{\text{fold}}(1) = \frac{6 + 1}{2 \cdot 1} \cdot 9 + \frac{6 - 1}{-2 \cdot 1} \cdot 10 = \frac{7}{2} \cdot 9 + \frac{5}{-2} \cdot 10.$$

In $\mathbb{F}_{19}$, the inverse of 2 is 10, so:

$$\frac{7}{2} \equiv 7 \cdot 10 = 70 \equiv 13 \pmod{19}, \quad \frac{-5}{2} \equiv -5 \cdot 10 = -50 \equiv 7 \pmod{19}.$$

Thus:

$$f_{\text{fold}}(1) = 13 \cdot 9 + 7 \cdot 10 = 117 + 70 = 187 \equiv 16 \pmod{19}.$$

**Verifier Check** The verifier selects index $i = 0$, corresponding to $x = 1$, and requests the following from the prover:

- The values $f(1)$ and $f(16)$;
- Their respective Merkle authentication paths;
- The claimed value $f_{\text{fold}}(1) = 16$.

The verifier computes the expected folded value using Equation (III.3) and checks for consistency. Since the computed value matches and the Merkle paths validate the correctness of the values at $x$ and $-x$, the round is accepted.

**Conclusion** This procedure is repeated for multiple rounds of folding until the polynomial is reduced to a constant. In the final query phase, the verifier checks a small number of randomly selected positions across all layers to ensure consistency with the folding process. This achieves a sound and efficient proximity test for low-degree polynomials.

### III.7.3 FRI Polynomial commitment

Although FRI significantly improves the efficiency of proximity testing, it introduces two important limitations when employed as the basis for a polynomial commitment scheme:

- The prover commits to the evaluations of a polynomial $f$ only over a restricted domain $\Omega \subset \mathbb{F}_q$. Consequently, if the verifier queries an evaluation point $r \in \mathbb{F}_q \setminus \Omega$, the prover lacks a valid authentication path to provide a convincing answer. This limits the commitment's applicability to arbitrary points in the field.
- The verifier is only guaranteed that the committed function is *close* to a low-degree polynomial with respect to the relative Hamming distance. However, this proximity guarantee does not necessarily imply that the function is itself of low degree.

To address both limitations introduced by FRI, one can employ a standard algebraic technique used in many proof systems. Specifically, to prove that a committed function $f$ evaluates to a claimed value $v$ at an arbitrary point $r \in \mathbb{F}_q$, it suffices to show the existence of a polynomial $w(X)$ of degree at most $d$ such that

$$f(X) - v = w(X) \cdot (X - r).$$

This identity implies that the polynomial $f - v$ vanishes at $X = r$, i.e., $f(r) = v$. This algebraic condition can serve as the basis for constructing a succinct proof of correct evaluation at arbitrary points, and is often layered on top of protocols such as GKR (III.5) or FRI to enable general-purpose polynomial commitments.

To confirm that $f(r) = v$, the verifier $\mathcal{V}$ applies FRI's fold and query procedure to the function

$$g(X) := \frac{f(X) - v}{X - r}$$

using degree bound $d - 1$.

If $\mathcal{V}$ queries this function at a point in the domain $\Omega$, the evaluation can be obtained with one query to $f$ at the same point (also see IOPs).

It can be proved that to pass verifier's checks in this polynomial commitment with noticeable probability, $v$ must be equal to $h(r)$, where $h$ is the degree-$d$ polynomial that is closest to $f$ (in Hamming distance).

For a discussion of potential issues with applying the Fiat-Shamir transform to FRI, see Appendix A.4.

**Costs** The costs of the FRI polynomial commitment are the following:

| | |
|---|---|
| Setup phase | $\mathcal{O}(1)$ (transparent setup) |
| Prover time | $\mathcal{O}(d \log_2 d)$ |
| Commitment size | $\mathcal{O}(1)$ (Merkle root) |
| Proof size | $\mathcal{O}(\log_2^2 d)$ |
| Verification time | $\mathcal{O}(\log_2^2 d)$ |

Table III.1: Asymptotic costs of the FRI polynomial commitment scheme.

Chapter IV

# Zk-STARKs

Zero-Knowledge Scalable Transparent Arguments of Knowledge (zk-STARKs) are designed to address some of the critical limitations of zk-SNARKs, particularly the reliance on a trusted setup. zk-STARKs employ transparent setup processes based on publicly verifiable randomness, eliminating the risk inherent in trusted setup ceremonies. Furthermore, zk-STARKs are constructed using cryptographic hash functions rather than elliptic curve pairings, rendering them post-quantum secure, as quantum computers threaten the assumptions underlying elliptic-curve-based cryptography. The importance of zk-STARKs lies in their transparency, scalability, and long-term security.

The first formal definition of these concepts was introduced in the paper entitled "Scalable, transparent, and post-quantum secure computational integrity" [BBHR18b].

## IV.1 Computational Integrity and the DPM Problem

A motivating example of the problem addressed by [BBHR18b] is the DNA Profile Match (DPM) scenario. Suppose a national police force (P), which maintains a confidential forensic DNA profile database (D), claims that the DNA profile (p) of a controversial presidential candidate does not appear in D. How can cryptographic protocols be used to convince the skeptical public of this claim, without revealing any information about D or p, without relying

on any trusted third party, and within reasonable computational effort?

Figure IV.1: (P)

Figure IV.2: (p)

Figure IV.3: (D)

This example illustrates a broader issue: when a party P performs a computation C on a dataset D, there is always a risk that P might misreport the result $C(D)$, thereby undermining computational integrity (CI). Verifying CI requires ensuring that P has correctly performed and reported the computation without external oversight.

If the dataset D is public, then any verifier V can recompute C on D and compare the result with the claimed output. However, this verification strategy is inefficient: the verifier's runtime ($T_V$) is linear in the computation time ($T_C$) and requires access to the entire dataset D. This is similar to checking a restaurant bill or validating the Bitcoin blockchain.

To address this inefficiency, cryptographic hash functions can be used to compute a succinct, immutable fingerprint $cm_t$ of the dataset D at a given time t. This fingerprint can be recorded on a public blockchain for future reference. However, when D contains private data, such public verification becomes infeasible, as it risks revealing sensitive information.

Traditionally, one would resort to a trusted third party (e.g., an auditor) to verify computations over private data. However, this solution suffers from scalability limitations and introduces a central point of failure. To overcome these challenges, zero-knowledge (ZK) proof systems offer a compelling alternative.

As discussed in Chapter II, zero-knowledge proof system for a computation C consists of a prover P and a verifier V. The prover aims to convince the verifier that a certain computation was correctly executed, without revealing any additional information. A secure ZK system satisfies two main properties:

- **Completeness:** An honest prover can convince the verifier of a true

statement.

- **Soundness:** A dishonest prover cannot convince the verifier of a false statement, except with negligible probability.

Early ZK systems with scalable verification were based on **Probabilistically Checkable Proofs (PCPs)**. The PCP theorem guarantees the following:

1. The prover's runtime is polynomial in the computation: $T_P = T_C^{O(1)}$
2. The verifier's runtime is polylogarithmic in the computation: $T_V = \log^{O(1)} T_C$

ZK systems based on PCPs (ZK-PCPs) exhibit the following key features:

1. **Post-quantum security:** These systems rely on collision-resistant hash functions for interactive protocols and on the Random Oracle Model (ROM) for non-interactive ones.
2. **Proof/Argument of Knowledge (POK/ARK):** These systems enable the prover to demonstrate knowledge of private data without revealing it, and to prove the correctness of a computation.
3. **Transparency:** The verifier's randomness is publicly known, removing the need for a trusted setup.

## IV.2 Interactive Oracle Proofs (IOPs)

To improve prover scalability without sacrificing the essential properties of ZK-PCPs (i.e., transparency, universality, confidentiality, post-quantum security, argument of knowledge, and scalable verification), the **Interactive Oracle Proof (IOP)** model was introduced. IOPs generalize both PCPs and Interactive Proofs (IPs): like IPs, they involve multiple rounds of interaction, but like PCPs, the verifier is given **query access** to each message sent by the prover.

More formally, IOPs allow the prover to write messages to an oracle in each round, and the verifier can inspect any symbol of these messages by

querying it directly. This enables **sublinear-time verification** relative to the total proof length.

Ben-Sasson, Chiesa, and Spooner [BCS16] showed that any IOP can be converted into a non-interactive argument in the Random Oracle Model using the Fiat-Shamir heuristic. This transformation allows interactive protocols to be used in non-interactive settings, a key step in constructing zk-SNARKs and zk-STARKs.

## Polynomial IOPs

An important subclass of IOPs is the **Polynomial IOP (PIOP)**, in which certain messages from the prover are represented as low-degree polynomials. Let $h_i$ denote one such polynomial over a finite field, of degree at most $d_i$. The verifier can query the evaluation of $h_i$ at any point $r$ of its choice.

To turn PIOPs into practical SNARKs or STARKs, these polynomials are replaced with **polynomial commitment schemes**. Examples of such schemes include Ligero [AHIV22] and FRI [BBHR18a]. The general strategy is as follows:

- Start with a PIOP for a relation such as R1CS satisfiability.
- Replace polynomial messages with commitments.
- Use Fiat-Shamir to make the protocol non-interactive.

The resulting system remains an IOP at its core, but is now scalable, transparent, and non-interactive.

# IV.3 Arithmetization Models: AIR and CCS

To enable the use of algebraic proof systems such as STARKs, computations must be converted into algebraic form. Two key models for this purpose are **Algebraic Intermediate Representation (AIR)** and **Customizable Constraint Systems (CCS)**.

## AIR (Algebraic Intermediate Representation)

AIR represents computations using algebraic constraints over bounded-degree polynomials. It is structured as follows:

- **Finite field:** Computations are performed over a finite field $\mathbb{F}_q$ with prime $q$ carefully chosen for efficient arithmetic. For example, $q = 2^{23} - 2^{13} + 1$.
- **Execution trace:** A matrix $T$ of size $n \times m$ where $n$ is the number of computation steps and $m$ is the number of algebraic registers.
- **Transition constraints:** Algebraic relations that span consecutive rows (or with a fixed offset) in $T$.
- **Boundary constraints:** Conditions that fix initial or final values in $T$ to encode the inputs and outputs.
- **Probabilistic verification:** Constraints are checked at a single evaluation point $\xi \in \mathbb{F}_q$, randomly chosen by the verifier.
  By the Schwartz–Zippel lemma (III.1.1), a dishonest proof fails with high probability.
- **Randomized AIR with preprocessing (RAP):** The prover sends the execution trace, receives the verifier's challenge $\xi$, and returns an auxiliary trace. Fiat–Shamir renders this non-interactive.

## CCS (Customizable Constraint Systems)

CCS generalizes multiple arithmetization models (R1CS, AIR, Plonkish circuits) and is structured as follows:

- **Witness polynomials:** These encode the state of the computation using low-degree polynomials.
- **Selectors:** Define where and how constraints apply, similar to gates in R1CS.
- **Constraint composition:** Local constraints are aggregated into a global constraint polynomial.

- **Vanishing checks:** The global constraint polynomial must vanish over a predefined domain.

CCS serves as a unifying framework to capture most practical constraint systems used in modern ZK protocols.

## IV.4 ZK-STIKs and zk-STARKs

If an IOP-based proof system is:

- **Scalable:** Prover runtime is bounded by

$$T_P = T_C \cdot \log^{O(1)} T_C$$

- **Transparent:** No trusted setup
- **Zero-Knowledge:** Protects witness confidentiality

then it qualifies as a **ZK-STIK** (Scalable Transparent IOP of Knowledge).

By making such a system non-interactive using Fiat-Shamir, one obtains a **zk-STARK**: a **Scalable Transparent IOP Argument of Knowledge**.

zk-STARKs thus represent a culmination of decades of theoretical development in probabilistic proof systems, yielding a practical, secure, and efficient solution to verifying computations over both public and private data.

## IV.5 Example: Fibonacci square sequence

The basic principles of the STARK protocol are described by building a proof of a simple statement (knowledge of Fibonacci square sequence element) as an example.

### Preliminaries

Let $\mathbb{F}$ be a prime-order field of size $N$. Let $\mathbb{F}^\times$ denote the multiplicative group of $N-1$ elements, and let $\omega$ be any fixed primitive element in $\mathbb{F}^\times$. The field size is set to $N = 3 \cdot 2^{30} + 1$, and a multiplicative subgroup of order $2^k$ is required. For instance, $2251 + 17 \cdot 2^{192} + 1$ may also be suitable.

## Merkle trees

A Merkle tree is a cryptographic construction that allows committing to a large dataset using a single hash value. Each leaf node contains the hash of a data block, while each internal node holds a hash derived from its child nodes. To reveal any leaf's data, its value must be presented along with the Merkle path (Merkle proof).

## STARK Protocol

**Fibonacci Square Sequence:** $a_i = a_{i-1}^2 + a_{i-2}^2$ The following statement is to be proven:

> A field element $x$ exists such that the 1023rd element of the Fibonacci square sequence starting with 1 and $x$ is 2338775057.

Private input: $x = 3141592$.

### IV.5.1 Trace Polynomial

The *trace* is defined as a sequence of elements that serves as the foundation of the proof. This sequence comprises both private and public values and is required to satisfy a set of specific constraints. In the illustrative example under consideration, the trace is taken to be the sequence $a$ consisting of the first 1023 elements of the Fibonacci square sequence.

This trace is assumed to represent the evaluation of an unknown trace polynomial of degree $|a| - 1$, in accordance with the Unisolvence Theorem (III.1.2). The term *domain* refers to a sequence of elements from the field $\mathbb{F}$ at which polynomials are evaluated. In order to interpolate the trace polynomial, a multiplicative subgroup of 1024 elements from $\mathbb{F}^\times$ is chosen as the domain:

$$G = \{g^i \mid g = \omega^{3 \cdot 2^{20}},\ i \in [0, 1024)\}.$$

Using Lagrange interpolation over the 1023 points $\{(g^i, a_i)\}_{i=0}^{1022}$, a trace polynomial $f \in \mathbb{F}[x]$ is constructed.

It is worth noting that, in practice, more efficient algorithms for interpolation may be employed, such as the Fast Fourier Transform (FFT).

### IV.5.2 Polynomial Commitment

To commit to the trace polynomial, an evaluation domain significantly larger than the degree of the polynomial is employed. Specifically, the evaluation domain is selected to be a multiplicative subgroup of 8192 elements in $\mathbb{F}^\times$:

$$H = \{h^i \mid h = \omega^{3 \cdot 2^{17}},\ i \in [0, 8192)\}.$$

The evaluation set is then defined as:

$$E = \{\omega \cdot h_i \mid h_i \in H\}.$$

A Merkle tree is constructed over the evaluations $\{f(e_i)\}_{e_i \in E}$, and the root of this Merkle tree is referred to as the *trace polynomial commitment*. This same commitment approach is utilized for additional polynomials throughout the remainder of the protocol.

**Constraints** The initial statement implies the following constraints:

1. $a_0 = 1$
2. $a_{1022} = 2338775057$
3. $a_{i+2} = a_{i+1}^2 + a_i^2 \mod N$

In order to verify that the committed trace polynomial satisfies all imposed constraints, it is sufficient to verify that it vanishes at specific points. These points are determined by the selected interpolation pairs $(g^i, a_i)$, where $g$ is a generator of the multiplicative subgroup of the field $\mathbb{F}$ of order 1024. The constraints under consideration are the following:

1. The initial value $a_0$ is equal to 1, which is equivalent to stating that the polynomial $f(x) - 1$ has a root at $x = g^0 = 1$.
2. The final value $a_{1022}$ is equal to 2338775057, which corresponds to the condition that $f(x) - 2338775057$ has a root at $x = g^{1022}$.
3. For every index $i$ such that $0 \leq i \leq 1020$, the recurrence relation

   $$a_{i+2} = a_{2i+1} + a_{2i}$$

   must hold. This can be translated into the condition that the polynomial

   $$f(g^2 x) - f(gx)^2 - f(x)^2$$

   vanishes at all points $x = g^i$ for $i \in \{0, \ldots, 1020\}$.

To formalize these conditions, the following constraint polynomials are defined:

$$p_0(x) = \frac{f(x) - 1}{x - 1},$$
$$p_1(x) = \frac{f(x) - 2338775057}{x - g^{1022}},$$
$$p_2(x) = \frac{f(g^2x) - f(gx)^2 - f(x)^2}{\prod_{i=0}^{1020}(x - g^i)}.$$

The polynomials $p_0(x)$ and $p_1(x)$ are clearly well-defined, since their numerators vanish at the corresponding denominator roots by construction. However, the form of $p_2(x)$ is inconvenient to manipulate due to the high-degree denominator.

To simplify $p_2(x)$, one can exploit the following identity over the multiplicative subgroup $G = \{g^i \mid 0 \leq i < 1024\}$:

$$x^{1024} - 1 = \prod_{g \in G}(x - g),$$

which holds since both sides are polynomials whose roots are precisely the elements of $G$.

Consequently, the denominator of $p_2(x)$ can be replaced by a divisor of $x^{1024} - 1$ that omits the points where the constraint is not required to hold, namely $g^{1021}, g^{1022}, g^{1023}$. Thus, the final form of the third constraint polynomial is given by:

$$p_2(x) = \frac{(f(g^2x) - f(gx)^2 - f(x)^2)(x - g^{1021})(x - g^{1022})(x - g^{1023})}{x^{1024} - 1}.$$

This formulation satisfies an essential property: the verifier, who receives only evaluations of the trace polynomial at points $x$, $gx$, and $g^2x$, can reconstruct the constraint polynomials $p_i(x)$ without any additional witness information. This guarantees that the constraints are checkable solely from the provided evaluations and the publicly known structure of the domain.

### IV.5.3 Composition Polynomial

In order to aggregate all the individual constraint polynomials into a single expression, a standard approach consists of computing a linear combination

using random coefficients provided by the verifier. This technique ensures that, with high probability, the satisfaction of the combined constraint implies the satisfaction of all individual constraints, while maintaining soundness and zero-knowledge properties.

Specifically, after the prover commits to the trace polynomial, the verifier samples and sends three random field elements $\alpha_0, \alpha_1, \alpha_2 \in \mathbb{F}$, which serve as challenge scalars. The prover then constructs the *composition polynomial* by linearly combining the previously defined constraint polynomials $p_0(x)$, $p_1(x)$, and $p_2(x)$ as follows:

$$CP(x) = \alpha_0 \cdot p_0(x) + \alpha_1 \cdot p_1(x) + \alpha_2 \cdot p_2(x).$$

This composition polynomial encapsulates all the constraints in a single expression and is used in the remainder of the proof system to enforce the correctness of the computation trace.

To allow efficient verification, the prover evaluates $CP(x)$ over the evaluation domain (typically a multiplicative subgroup of the field) and commits to the resulting codeword by constructing a Merkle tree over the evaluations. This commitment enables subsequent queries and consistency checks during the proof, while maintaining succinctness and integrity guarantees.

### IV.5.4 Fast Reed-Solomon IOP of Proximity (FRI)

The overall goal is to verify that the committed polynomial $CP(x)$ satisfies all the imposed constraints by checking its evaluation at a random point chosen by the verifier from the evaluation domain. However, a potential issue arises when a malicious prover constructs a polynomial of large degree that admits many roots in the field, thus undermining security. For instance, even a field of size $2^{61}$ is not sufficiently secure when verifying the polynomial at only one point. Therefore, it is necessary to ensure that the degree of the committed polynomial lies within an acceptable upper bound, which depends on the size of the computation trace.

The final stage of the STARK protocol is the *Fast Reed-Solomon Interactive Oracle Proof of Proximity* (FRI). FRI is an interactive protocol between a prover and a verifier that establishes that a given evaluation corresponds to a polynomial of low degree, where low degree means at most a $\rho$-fraction of the size of the evaluation domain.

The key idea behind the FRI protocol is to iteratively reduce the degree of a polynomial $g_0(x)$ of degree $n = 2^t$ by half, repeating this process until the

polynomial becomes constant. Consider the polynomial $g_0(x)$ with degree $n$ and the initial evaluation domain $E_0 = E$. The polynomial $g_0(x)$ can be decomposed by grouping its even and odd coefficients into two separate polynomials:

$$g_0^e(x^2) = \sum_{i=0}^{\frac{n}{2}} a_{2i} \cdot x^{2i}, \quad g_0^o(x^2) = \sum_{i=0}^{\frac{n}{2}} a_{2i+1} \cdot x^{2i}.$$

Then, the next-layer FRI polynomial is defined as

$$g_1(x^2) = g_{0e}(x^2) + \beta \cdot g_{0o}(x^2),$$

where $\beta$ is a challenge randomly chosen by the verifier. The prover commits to $g_1(x^2)$ over the next-layer evaluation domain $E_1$ and repeats this procedure until $g_i(x^{2^i})$ becomes a constant polynomial.

Concretely, starting from the evaluation domain

$$E_0 = \{w \cdot h^i \mid i \in [0, 8192)\},$$

the next-layer domain is defined as

$$E_1 = \{(w \cdot h^i)^2 \mid i \in [0, 4096)\}.$$

This structure allows expressing $g_0^e(x^2)$ and $g_0^o(x^2)$ using evaluations of $g_0$ as follows:

$$g_0^e(x^2) = \frac{g_0(x) + g_0(-x)}{2}, \quad g_0^o(x^2) = \frac{g_0(x) - g_0(-x)}{2x},$$

which gives

$$g_1(x^2) = g_0^e(x^2) + \beta \cdot g_0^o(x^2).$$

It is important to note that for the above expressions to be valid, the domain $E$ must be closed under negation, i.e., $-x \in E$ for every $x \in E$ (in order to achieve that $g_i(-x)$ can be committed together with $g_i(-x)$). This property holds because the evaluation domain is chosen as a multiplicative coset of size $2^k$ for some integer $k$. The proof of this fact (see Appendix A.1) applies similarly to each subsequent domain $E_i$.

### IV.5.5 Protocol Definition

The verification of the entire computation is performed through the following interactive protocol between the prover and the verifier.

---

**Protocol:** Zero-Knowledge Scalable Transparent Argument of Knowledge (**ZK-STARK**)

---

**Input:**

- *Private:* $a_1 = 3141592$ (secret element of the Fibonacci square sequence).
- *Public:* $a_0 = 1$, $a_{1022} = 2338775057$ (fixed elements of the Fibonacci square sequence) and $\mathbb{F}$ (finite field over which computations are performed).

**Protocol:**

1. The prover interpolates the trace polynomial $f(x)$ and submits its commitment to the verifier.
2. The verifier samples random challenges $\alpha_0, \alpha_1, \alpha_2 \in \mathbb{F}$ and sends them to the prover.
3. The prover constructs the composition polynomial $CP(x)$ as a linear combination of the constraint polynomials using the challenges $\alpha_i$, then commits to $CP(x)$ and sends the commitment to the verifier.
4. The verifier selects a random index $i \in [0, 8192 - 16)$, computes $c = w \cdot h^i$, and sends $c$ to the prover.
5. The prover responds with the evaluations $f(c)$, $f(gc)$, $f(g^2c)$, $CP(c)$, $CP(-c)$ along with the corresponding Merkle proofs.

   *Note: To claim $gx$ where $x = w \cdot h^i \in E$ (evaluation domain), the prover uses power shifting by indices $i+8$ and $i+16$ to obtain $g^2x$ (see Appendix A.1).*
6. The verifier verifies the correctness of the Merkle proofs and checks the validity of $CP(c)$ by evaluating the constraint polynomials $p_0(c)$, $p_1(c)$, and $p_2(c)$.

7. The prover and verifier engage in the FRI protocol to verify low-degree proximity of $g_0(x) = CP(x)$. At each FRI iteration $i$, the prover commits to the polynomial $g_i(x)$, and the verifier selects a challenge $\beta$. The prover responds with evaluations $g_i(c)$, $g_i(-c)$, which the verifier uses to compute $g_{i+1}(c)$. This process repeats until $g_i(x)$, for $i \in [1, 12)$, reduces to a constant polynomial.

   *Note: The range $i \in [1, 12)$ corresponds to the maximum degree of the composition polynomial $CP(x)$, which in this example satisfies*

$$\deg CP(x) \leq 11$$

Chapter V

# Post-Quantum Privacy Pass via Post-Quantum Anonymous Credentials

## V.1 Introduction

Privacy Pass is a protocol that issues unlinkable tokens to clients, enabling anonymous authentication (Fig.V.1). Classical implementations rely on elliptic-curve cryptography and blind signatures, which are rendered insecure by quantum adversaries.

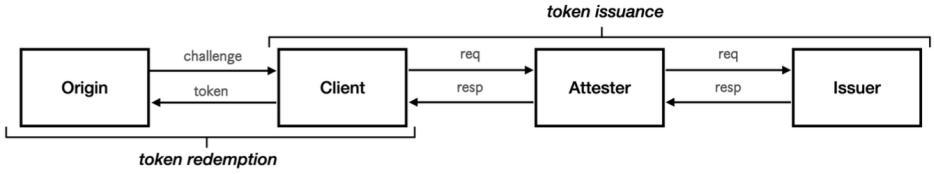



Figure V.1: Privacy Pass token issuance and redemption interaction overview

This chapter presents a post-quantum redesign of Privacy Pass based on lattice-based anonymous credentials and zero-knowledge proofs constructed with zk-STARKs. After describing the post-quantum threat model, the chapter introduces the underlying proof system, a STARK-friendly variant of Dilithium, the construction of anonymous credentials, and a protocol extension for rate-limiting. Experimental results confirm that the proposed

approach remains practical even when proof sizes range from 85–175 kB.

*Note: It is important to highlight that the protocol presented in this chapter does not implement a zk-STARK in the strict sense as defined by Ben-Sasson in [BBHR18b]. Instead, the construction deviates from the formal notion of zk-STARKs and does not achieve zero-knowledge.*

## V.2 Cryptographic Setting and Motivation

Quantum adversaries capable of executing Shor's and Grover's algorithms undermine the security of schemes based on discrete logarithms and RSA. A post-quantum variant of Privacy Pass must ensure the following properties:

- **Post-quantum security**: All primitives must be secure against quantum attacks.
- **Unlinkability**: Multiple authentications from the same client must remain uncorrelated.
- **Selective disclosure**: Only the predicates required by the verifier should be revealed.
- **Transparency**: Proof systems must avoid trusted setups.

These objectives are fulfilled using lattice-based digital signatures, hiding commitments, and transparent zk-STARKs.

## V.3 zk-STARK Framework

A zk-STARK is a scalable, transparent argument of knowledge that proves statements represented as AIR (Arithmetic Intermediate Representation) constraints over a finite field $\mathbb{F}_q$. An execution trace is modeled as a matrix $\mathbf{T} \in \mathbb{F}_q^{m \times n}$, where rows denote time steps and columns denote registers. Boundary constraints fix initial values, and transition constraints (of bounded degree) express state evolution. Soundness is guaranteed probabilistically via the Schwartz–Zippel lemma (III.1.1).

To support deeper traces and avoid costly non-native arithmetic, computations are embedded in $\mathbb{F}_q$ with prime $q = 2^{23} - 2^{13} + 1$ or $q = 2^{23} - 2^{20} + 1$, and extended to $\mathbb{F}_{q^6}$ for proof evaluation.

### Randomized AIR with Preprocessing

Verification of polynomial identities such as $f(X)g(X) = h(X)$ is reduced to checking the identity at a random verifier-supplied point $\xi$. This is implemented in the STARK framework via a Randomized AIR with Preprocessing (RAP): the prover sends the execution trace, receives the challenge $\xi$, and responds with an auxiliary trace. The Fiat–Shamir heuristic (II.3) renders the protocol non-interactive.

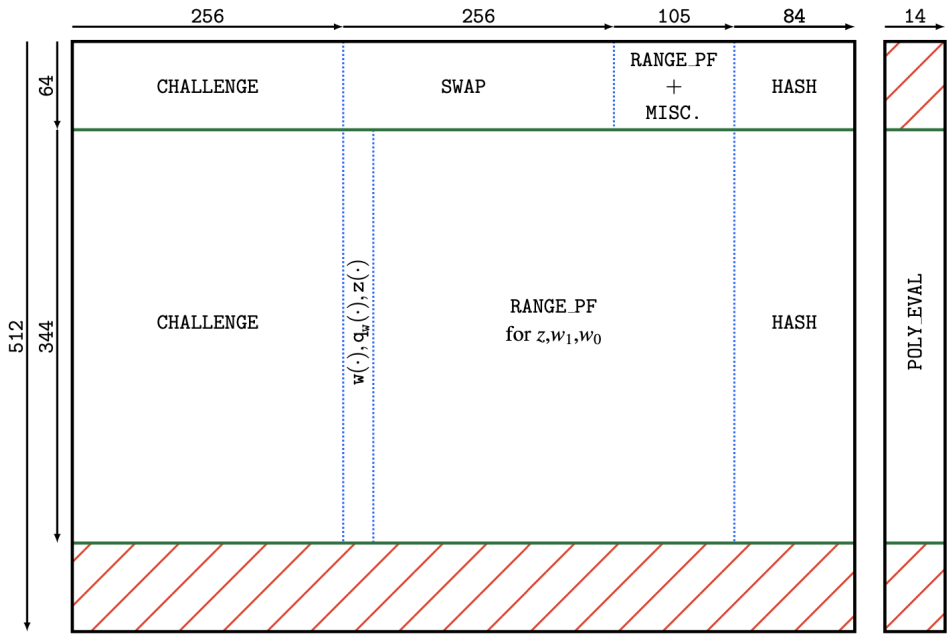



Figure V.2: Execution trace for the signature verification RAP. Sizes of components are roughly to scale.

## V.4 zkDilithium: STARK-Friendly Signature Scheme

Dilithium (see Appendix A.2) is modified to permit efficient verification within zk-STARKs. The adapted scheme, referred to as zkDilithium, in-

cludes the following changes:

1. The SHA-3 hash function is replaced with Poseidon, an algebraic hash suitable for STARKs.
2. The public key includes the full $t = As_1 + s_2$ vector, avoiding the use of public key compression and eliminating the need for hints.
3. The `SampleInBall` procedure, which generates a sparse challenge polynomial with $\tau$ coefficients in $\{-1, 0, +1\}$, is reformulated to avoid rejection sampling. Instead, the procedure selects swap indices and signs using field elements and performs a Fisher–Yates shuffle. A negligible rejection probability ($< 0.06\%$) is preserved.
4. For applications requiring larger trace depth, a variant denoted zk20Dilithium is introduced, using $q = 2^{23} - 2^{20} + 1$ and adjusted bounds ($\gamma_2 = 2^{16}$).

### Verification Constraints

Let $z$ be the Dilithium response and $w = Az - ct$. The value $w$ is decomposed as $w = 2\gamma_2 w_1 + w_0$, where $w_0 \in (-\gamma_2, \gamma_2]$. STARK constraints enforce the validity of this decomposition:

$$w_0 + w_1 2\gamma_2 - w = 0, \tag{V.1}$$

$$(1 - w_2) w_1 w_0 = 0, \qquad w_2(w_2 - 1) = 0, \tag{V.2}$$

$$\|z\|_\infty \leq \gamma_1 - \beta. \tag{V.3}$$

Here, $w_2$ is a binary flag indicating whether $w = -\gamma_2$, a special case required for correctness.

## V.5 Anonymous Credential Construction

The credential system is constructed following the Camenisch–Lysyanskaya paradigm:

1. The client commits to a vector of attributes $\mathbf{a} = (a_1, \ldots, a_k)$, obtaining a hiding commitment $C$.

2. The issuer signs $C$ using zkDilithium via a blind signature protocol.
3. The client presents $(C, \sigma)$ and a zero-knowledge proof showing that the committed attributes satisfy a predicate $P(\mathbf{a})$, without revealing the attributes themselves.

The STARK proof certifies:

- knowledge of a valid signature on a hidden commitment,
- correct opening of $C$ to $\mathbf{a}$,
- satisfaction of the predicate $P$.

The resulting proof has logarithmic complexity relative to the circuit size. Empirical evaluations indicate proof sizes between 85 and 175 kB.

## V.6 Rate-Limited Privacy Pass

The credential system enables token-based rate limiting without an attester. Each credential embeds counters representing usage in different time windows:

$$(\mathsf{nonce},\ t_{\text{issue}},\ c_{5\text{m}},\ c_{1\text{h}},\ c_{1\text{d}})$$

When redeeming a token, the client generates a new commitment $C'$ with updated counters and provides a zero-knowledge proof asserting:

1. knowledge of a valid signature on a previous credential,
2. correct computation of updated counters,
3. compliance with a rate-limiting policy (e.g., no more than 300 uses per 5 minutes).

Double-spending is prevented by revealing only the previous nonce. The origin signs the new credential upon successful verification.

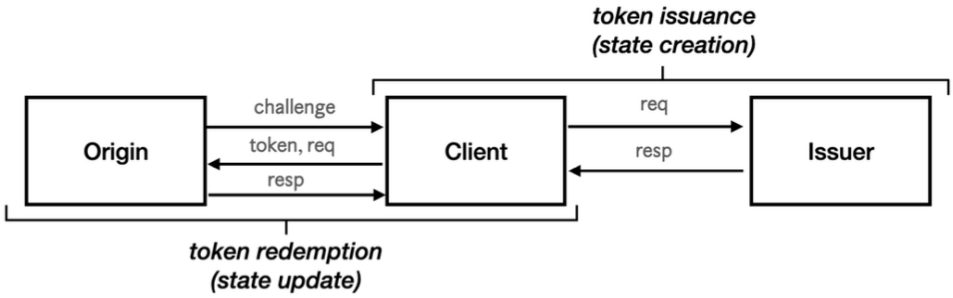



Figure V.3: Protocol flow for rate-limited Privacy Pass with anonymous credentials.

| Variant | Proof size (kB) | Prover time (s) | Verifier time (ms) |
|---|---|---|---|
| Size-optimized | 85.6 | 4.8 | 19.8 |
| Balanced | 112.3 | 0.66 | 22.0 |
| Time-optimized | 173.3 | 0.30 | 31.4 |

## V.7 Implementation and Performance

The proposed protocol is implemented in Rust using the Winterfell library. All benchmarks target 115-bit conjectured security.

A measurement study involving over 750,000 client sessions demonstrated that over 90% of users experienced an upload latency below 1 second for 100 kB proofs, assuming median upstream bandwidths above 2.5 Mbit/s.

## V.8 Comparison with Prior Work

Recent proposals for lattice-based blind signatures report signature sizes ranging from 22 to 100 kB, often accompanied by large transcripts and long proving times. The credential scheme presented here offers comparable sizes and significantly greater flexibility, including support for arbitrary predicates, efficient verification, and full transparency.

Chapter VI

# Conclusion

In the paper Post-Quantum Privacy Pass via Post-Quantum Anonymous Credentials, the authors claim to construct a privacy-preserving authentication protocol based on "post-quantum zk-STARKs." However, a closer examination reveals that this usage of the term "zk-STARK" diverges significantly from the formal, technical definition established in the literature on zero-knowledge proofs. The Formal Definition of zk-STARK Traditionally, a zk-STARK (Zero-Knowledge Scalable Transparent Argument of Knowledge) is defined by a set of specific structural and cryptographic components:

- AIR (Algebraic Intermediate Representation): Encodes a computation as a set of polynomial constraints.
- LDE (Low-Degree Extension): Extends the computation trace to a larger evaluation domain to enable error detection.
- FRI (Fast Reed–Solomon IOP): A low-degree test protocol used to prove that a function is close to a low-degree polynomial.
- Merkle Commitments over hash functions for transparent commitments to the trace.

These ingredients collectively enable highly scalable, post-quantum secure and transparent proofs of arbitrary computations. STARKs are particularly suited for applications like verifiable computation, zk-rollups, and general-purpose zero-knowledge machines.

Despite the reference to zk-STARKs, the construction proposed in the Privacy Pass paper does not define an AIR. Does not apply a low-degree extension or use the FRI protocol. Does not rely on polynomial commitments or Merkle trees in the standard STARK fashion. Is instead tailored to a very specific ZK proof of knowledge of a Dilithium signature, implemented via hand-crafted circuit constraints. Moreover, the authors acknowledge that they were unable to implement the ZK component fully due to challenges in encoding Dilithium's structure within existing proof systems.

The authors seem to use the term "zk-STARK" to refer more generally to a zero-knowledge proof system that is transparent, scalable in principle, and post-quantum secure, regardless of whether it follows the STARK pipeline. This looser definition reflects a shift from structural fidelity to property-based reasoning: rather than using zk-STARK as a label for a specific cryptographic protocol, it becomes shorthand for any ZK system that is non-interactive, transparent (i.e. no trusted setup), and post-quantum safe. While such an interpretation may be justified from a high-level perspective, it raises important questions about terminology consistency and rigor, especially in academic or security-critical contexts.

# Implications

This discrepancy highlights a broader issue in the cryptographic community: the growing divergence between formal definitions and practical usage of protocol labels such as "zk-SNARK," "zk-STARK," or even "ZK proof." As these systems become more modular and application-specific, terms are increasingly repurposed to describe their goals or security properties, rather than their mathematical architecture. From an academic standpoint, this creates a potential for confusion. For example: A reader might assume that the system uses algebraic trace-based STARKs with polynomial constraints, when it does not. The absence of AIR/LDE/FRI means the system cannot inherit the same composability and scalability guarantees as traditional STARK-based designs. It would be more precise for the authors to describe their system as a transparent post-quantum ZK proof, or as a non-interactive ZK argument based on hash functions, rather than using the zk-STARK label in an overloaded way.

Appendix A

# Technical Appendices

## A.1 Structural Properties of the STARK Evaluation Domain

### Proof of existence of additive inverse element in E

In general, this proof works for any field of order $N$ where it is possible to select a multiplicative subgroup of order $2^k$. In our example, $N = 3 \cdot 2^{30} + 1$ and $k = 13$.

Any element $x \in E$ can be written as $x = w \cdot h^i = w \cdot w^{i \cdot 3 \cdot 2^{17}}$ by construction. Without loss of generality, we can simplify this to $x = h^i$. Then,

$$x = h^i = w^{3 \cdot 2^{17} \cdot i} \quad \text{and} \quad -x = h^j = w^{3 \cdot 2^{17} \cdot j},$$

where

$$j = i + \frac{|E|}{2} \pmod{|E|}.$$

We will explain why this holds below. For simplicity, assume $i < j$. Then:

$$x + (-x) \equiv w^{3 \cdot 2^{17} \cdot i} \left(1 + w^{3 \cdot 2^{17} \cdot \frac{|E|}{2}}\right) \equiv 0 \pmod{N}.$$

Since $|E| = 4096 = 2^{12}$, we have:

$$1 + w^{3 \cdot 2^{17} \cdot 2^{12}} \equiv 0 \pmod{N}.$$

Note that:

$$w^{3\cdot 2^{29}} \equiv N - 1 \pmod{N},$$

and therefore

$$w^{3\cdot 2^{30}} \equiv (N-1)^2 \equiv 1 \pmod{N}.$$

The equation $w^{3\cdot 2^{30}} \equiv 1 \pmod{N}$ follows from the order property of the primitive element $w$ in the multiplicative group $\mathbb{F}^{\times}$.

### Proof of shifted element index in E

Any element $x \in E$ is equal to

$$x = w \cdot h^i = w \cdot w^{i\cdot 3\cdot 2^{17}}$$

by construction.

Then, while $g = w^{3\cdot 2^{20}}$, to claim $gx$ and $g^2x$ from $E$ we take $w \cdot h^{i+8}$ and $w \cdot h^{i+16}$:

$$gx = w^{3\cdot 2^{20}} \cdot w \cdot w^{i\cdot 3\cdot 2^{17}} = w \cdot w^{3\cdot 2^{17}(i+2^3)} = w \cdot h^{i+8}$$

$$g^2x = w^{3\cdot 2^{21}} \cdot w \cdot w^{i\cdot 3\cdot 2^{17}} = w \cdot w^{3\cdot 2^{17}(i+2^4)} = w \cdot h^{i+16}$$

This approach requires $i$ to be less than $|E| - 16$, so the verifier selects $i$ from the range $[0; 8192 - 16)$.

## A.2 The Dilithium Scheme

CRYSTALS-Dilithium is a lattice-based digital signature scheme built on the hardness of two fundamental problems in module lattices: the *Module Learning With Errors (MLWE)* and the *Module Short Integer Solution (MSIS)* problems. It was selected as the primary signature algorithm by NIST for standardization in the post-quantum era.

Let $R := \mathbb{Z}_q[X]/(X^n + 1)$ be a cyclotomic ring with $n$ a power of two (typically $n = 256, 512, 1024$) and modulus $q$ a prime (e.g., $q = 8380417$). The scheme operates over modules $R^k$ and $R^\ell$ for small integers $k, \ell$.

**Key Generation.**

- Sample uniformly random matrix $\mathbf{A} \in R^{l \times k}$;
- Sample secret vectors $\mathbf{s}_1 \in R^k$, $\mathbf{s}_2 \in R^\ell$ from a discrete centered distribution over small-norm polynomials;
- Compute $\mathbf{t} := \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 \in R^\ell$;
- Apply a rounding function `Power2Round` to decompose $\mathbf{t}$ into $(\mathbf{t}_1, \mathbf{t}_0)$.

The public key is $(\mathbf{A}, \mathbf{t}_1)$ and the secret key is $(\mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)$.

**Signing.** To sign a message $m$:

1. Sample nonce vector $\mathbf{y} \in R^k$ with small norm;
2. Compute $\mathbf{w} := \mathbf{A}\mathbf{y}$;
3. Extract the high bits of $\mathbf{w}$ to obtain $\mathbf{w}_1$;
4. Compute challenge $c := H(\mu, \mathbf{w}_1)$, where $\mu$ is a message-dependent hash;
5. Compute response $\mathbf{z} := \mathbf{y} + c \cdot \mathbf{s}_1$;
6. Reject and restart if $\mathbf{z}$ or auxiliary bounds are violated (rejection sampling).

The signature consists of $(\mathbf{z}, c, \mathbf{h})$ where $\mathbf{h}$ is a sparse hint vector enabling reconstruction of $\mathbf{w}_1$ during verification.

**Verification.** The verifier:

- Recomputes $\mathbf{w}' := \mathbf{A}\mathbf{z} - c \cdot \mathbf{t}_1$;
- Uses the hint $\mathbf{h}$ to derive $\mathbf{w}'_1$;
- Accepts if $c = H(\mu, \mathbf{w}'_1)$ and $\|\mathbf{z}\|$ is bounded.

**Security.** The Dilithium signature scheme has been proven to be existentially unforgeable under chosen-message attacks (EUF-CMA), assuming the hardness of MLWE and MSIS. Rejection sampling ensures leakage resilience, and deterministic hashing guarantees uniqueness.

## Modifications Introduced by zkDilithium20

To enable efficient integration into zero-knowledge proof systems (such as zk-STARKs), the zkDilithium20 variant introduces several critical adjustments. These changes aim to make Dilithium more compatible with algebraic circuit representations (e.g., R1CS or AIR) and to avoid operations that are difficult to arithmetize.

**1. Rounding Function Replacement.** The standard `Power2Round` and `Decompose` functions are bit-oriented and complex to model algebraically. zkDilithium20 replaces these with field-level rounding and decomposition that can be expressed via:

$$\mathbf{t} = \mathbf{t}_1 \cdot 2^D + \mathbf{t}_0,$$

where both components lie in $R^k$ and have coefficient bounds suited to field arithmetic. This avoids branching and modular reduction logic in the circuit.

**2. Hash Function Substitution.** SHAKE-256 is not STARK-friendly due to its bit-level permutation. zkDilithium20 replaces it with arithmetizable cryptographic hash functions like *Poseidon* or *Rescue*, which offer efficient representation over finite fields: $c = H_{\text{Poseidon}}(m, \mathbf{w}_1)$, where $H$ is designed to be efficiently verifiable inside low-degree polynomial constraint systems.

**3. Prime Field Alignment.** To avoid costly modular reduction inside proof systems, the modulus $q$ used in the ring $\mathbb{Z}_q[X]/(X^n + 1)$ is chosen such that $q < p$, where $p$ is the characteristic of the STARK field $\mathbb{F}_p$. This ensures native field compatibility.

**4. Signature Encoding for ZK Circuits.** Instead of using compressed or byte-oriented formats, signatures in zkDilithium20 are encoded directly as tuples over $\mathbb{F}_p$, enabling zero-knowledge verification circuits to avoid serialization logic and branching conditions.

**5. Optimized Constraints and Commitment Model.** The internal commitments to $\mathbf{w}_1$ and $\mathbf{z}$ are structured to minimize constraint complexity by using field-packing and deterministic encodings, enabling succinct non-interactive zero-knowledge proofs of signature validity.

*Note: zkDilithium20 is an experimental adaptation of Dilithium designed for zero-knowledge proof compatibility. While it retains the general structure and is inspired by the same MLWE/MSIS assumptions, its security has not been formally analyzed at the same level as CRYSTALS-Dilithium, and caution is advised when using it in critical applications.*

## A.3 Poseidon: A STARK-Friendly Hash Function

**Poseidon** is a cryptographic hash function designed specifically for efficient implementation inside zk-STARK and other algebraic proof systems. Key properties include:

- **Permutation-based construction:** Poseidon operates as a fixed number of rounds of a permutation applied to vectors over $\mathbb{F}_q$.
- **Native field operations:** It uses only additions, multiplications, and low-degree exponentiations (e.g., cube or fifth power), avoiding bitwise operations such as XOR or shifts that are costly in algebraic proof systems.
- **Algebraic S-boxes:** Nonlinear layers are implemented using polynomial maps (e.g., $x \mapsto x^5$) rather than lookup tables, which makes the function transparent and efficiently representable in arithmetic circuits.
- **Security:** Despite its algebraic simplicity, Poseidon is designed to resist standard cryptanalytic attacks (collision, preimage, differential) and achieves strong security margins for parameters matched to the underlying field.
- **Parameter flexibility:** The number of rounds, state size, and S-box exponent are chosen depending on the field $\mathbb{F}_q$ and desired security level.

Because of these features, Poseidon is well-suited as a drop-in replacement for SHA-family hashes in zk-STARK contexts, reducing proof size and verification complexity.

## A.4 Additional Notes on FRI

One issue with FRI is that the Fiat-Shamir transform should not be applied naively to render the folding phase non-interactive. The reason lies in the general principle that, when the Fiat-Shamir transform is used to make a multi-round interactive protocol non-interactive, a certain amount of security is typically lost at each round unless the protocol satisfies a stronger property known as *round-by-round soundness*.

This property ensures that soundness is preserved even when the verifier's challenges are generated in advance via a hash function, as in the Fiat-Shamir paradigm. While round-by-round soundness has been established for some protocols, such as the sum-check protocol III.4, no such guarantee has been proven for FRI.

Nevertheless, despite the absence of a formal round-by-round soundness proof, FRI is widely used in non-interactive settings in practical SNARK constructions and production systems.

Appendix B

# The Dual Use of the Term "Oracle"

In the context of this thesis, the term *oracle* refers to a theoretical abstraction commonly used in computational complexity and cryptographic proof systems. However, in the domain of blockchain technologies and smart contracts, the word "oracle" acquires a distinctly different meaning. Although both usages are technically sound within their respective domains, the semantic divergence can lead to confusion, especially when integrating zero-knowledge proof systems into blockchain-based environments.

This appendix aims to clearly distinguish between the two definitions, highlighting the conceptual and practical differences, and grounding the discussion with reference to the online course *CryptoZombies* [Loo24], a widely used educational resource for learning smart contract development on Ethereum.

## Oracle in Cryptographic Proof Systems (ZKPs, IOPs, ROM)

In cryptographic literature, particularly in zero-knowledge proofs (ZKPs), interactive proofs (IPs), and interactive oracle proofs (IOPs), an *oracle* is an **idealized mathematical abstraction** that allows an algorithm (usually the verifier) to access information in a non-standard way.

Examples include:

- In **Interactive Oracle Proofs**, the verifier has oracle (i.e., query)

access to the prover’s message and may inspect any part of it without reading the entire proof;

- In the **Random Oracle Model**, an oracle emulates a truly random function, typically instantiated by a cryptographic hash function in practice.

These oracles do **not exist physically**; they are conceptual tools used in security proofs and theoretical constructions to model limited knowledge, efficiency, or access complexity.

## Oracle in Blockchain and Smart Contracts

In contrast, within blockchain ecosystems (most notably Ethereum) an *oracle* is a **trusted external service** that delivers real-world data to smart contracts. Because smart contracts are designed to be deterministic and isolated from external sources for security reasons, they require oracles to access off-chain information.

This concept is introduced in the *CryptoZombies* course [Loo24], where oracles are explained as essential mechanisms for enabling smart contracts to interact with inputs such as:

- Cryptocurrency prices (e.g., ETH/USD);
- Weather conditions or external events;
- Secure randomness for games or lotteries.

Oracles in this context are implemented as **actual software services** (e.g., Chainlink) that must ensure integrity, availability, and often decentralization. They serve as a *bridge* between on-chain logic and off-chain data.

## Summary Comparison

| **Aspect** | **Cryptographic Proof Systems** | **Blockchain / Smart Contracts** |
|---|---|---|
| Definition | Abstract interface to access data or functions | External service that feeds real-world data to contracts |
| Example usage | IOPs, PCPs, ROM | Chainlink, Band Protocol, custom APIs |
| Physical existence | No | Yes (real-world software) |
| Purpose | Optimize verification and model limited access | Inject off-chain data into on-chain systems |
| Reference | Ben-Sasson et al. [BCS16], Fiat-Shamir, FRI | CryptoZombies [Loo24], Solidity Docs |

In advanced blockchain applications of zero-knowledge proofs, **both notions of oracle may coexist** at different levels of the protocol, thus, correctly distinguishing the context and role of the term is essential for building sound and secure cryptographic systems that interact with real-world environments.

# Bibliography


[AHIV22] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. *Ligero: Lightweight Sublinear Arguments Without a Trusted Setup*. Cryptology ePrint Archive, Paper 2022/1608. 2022. DOI: 10.1145/3133956. URL: https://eprint.iacr.org/2022/1608.

[Aro03] Sanjeev Arora. "Improved Low-Degree Testing and its Applications". In: *COMBINATORICA* 23 (July 2003), pp. 365–426. DOI: 10.1007/s00493-003-0025-0.

[BBHR18a] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. "Fast reed-solomon interactive oracle proofs of proximity". In: *45th international colloquium on automata, languages, and programming (icalp 2018)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. 2018, pp. 14–1.

[BBHR18b] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. *Scalable, transparent, and post-quantum secure computational integrity*. Cryptology ePrint Archive, Paper 2018/046. 2018. URL: https://eprint.iacr.org/2018/046.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. *Interactive Oracle Proofs*. Cryptology ePrint Archive, Paper 2016/116. 2016. URL: https://eprint.iacr.org/2016/116.

[BG93] Mihir Bellare and Oded Goldreich. "On Defining Proofs of Knowledge". In: *Advances in Cryptology — CRYPTO' 92*. Ed. by Ernest F. Brickell. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 390–420. ISBN: 978-3-540-48071-6.

[CDL16] Jan Camenisch, Manu Drijvers, and Anja Lehmann. "Anonymous Attestation Using the Strong Diffie Hellman Assumption Revisited". In: vol. 9824. Aug. 2016, pp. 1–20. ISBN: 978-3-319-45571-6. DOI: 10.1007/978-3-319-45572-3_1.

[Cou17] Geoffroy Couteau. "Zero-Knowledge Proofs for Secure Computation". PhD thesis. Nov. 2017.

[Cra97] Ronald Cramer. "Modular Design of Secure yet Practical Cryptographic Protocols". PhD thesis. Jan. 1997.

[FS99] Amos Fiat and Adi Shamir. "How To Prove Yourself: Practical Solutions to Identification and Signature Problems". In: vol. 263. Mar. 1999. ISBN: 978-3-540-18047-0. DOI: 10.1007/3-540-47721-7_12.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. "Delegating Computation: Interactive Proofs for Muggles". In: 62.4 (2015). ISSN: 0004-5411. DOI: 10.1145/2699436. URL: https://doi.org/10.1145/2699436.

[LFKN99] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. "Algebraic Methods for Interactive Proof Systems". In: *Journal of the ACM* 39 (Apr. 1999). DOI: 10.1145/146585.146605.

[Loo24] Loom Network. *CryptoZombies: Learn to Code Blockchain DApps by Building Simple Games*. https://cryptozombies.io/en/course/. Accessed July 2025. 2024.

[McC90] Kevin S. McCurley. "The Discrete Logarithm Problem". In: *Cryptology and Computational Number Theory*. Ed. by H. C. A.

van Tilborg. Vol. 42. American Mathematical Society, 1990, pp. 49–74. DOI: 10.1090/psapm/042/092d:11133.

[Ped92] Torben Pryds Pedersen. "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing". In: *Advances in Cryptology — CRYPTO '91*. Ed. by Joan Feigenbaum. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 129–140.

[PWFW23] Guru-Vamsi Policharla, Bas Westerbaan, Armando Faz-Hernández, and Christopher A Wood. *Post-Quantum Privacy Pass via Post-Quantum Anonymous Credentials*. Cryptology ePrint Archive, Paper 2023/414. 2023. URL: https://eprint.iacr.org/2023/414.

[Sch79] Jacob T. Schwartz. "Probabilistic algorithms for verification of polynomial identities". In: *Symbolic and Algebraic Computation*. Ed. by Edward W. Ng. Berlin, Heidelberg: Springer Berlin Heidelberg, 1979. ISBN: 978-3-540-35128-3.

[VSBW13] Victor Vu, Srinath Setty, Andrew J. Blumberg, and Michael Walfish. "A Hybrid Architecture for Interactive Verifiable Computation". In: *2013 IEEE Symposium on Security and Privacy*. 2013, pp. 223–237. DOI: 10.1109/SP.2013.48.

[ZGKPP17] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. *vSQL: Verifying Arbitrary SQL Queries over Dynamic Outsourced Databases*. Cryptology ePrint Archive, Paper 2017/1145. 2017. DOI: 10.1109/SP.2017.43. URL: https://eprint.iacr.org/2017/1145.

[Zip79] Richard Zippel. "Probabilistic algorithms for sparse polynomials". In: *Symbolic and Algebraic Computation*. Ed. by Edward W. Ng. Berlin, Heidelberg: Springer Berlin Heidelberg, 1979, pp. 216–226. ISBN: 978-3-540-35128-3.