Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Linear-Time Zero-Knowledge Arguments in Practice

Master Thesis

Ran Liao

October 15, 2022

Advisors: Prof. Dr. Kenny Paterson, Dr. Jonathan Bootle

Applied Cryptography Group
Institute of Information Security
Department of Computer Science, ETH Zürich

Abstract

Interactive zero-knowledge proofs allow an untrusted prover to convince a skeptical verifier that a statement is true without revealing any further information about why the statement is true. The polynomial commitment scheme is the key component of many proof systems, which allows the prover to commit to a polynomial, and later reveal the evaluation of the polynomial at a given point, while allowing the verifier can check that the evaluation is correct. After years of research, many schemes with linear prover time have been proposed, however, there is little work on their concrete efficiency and performance.

In this thesis, we want to investigate the concrete efficiency of those polynomial commitment schemes, especially in high-dimensional situations. We implement the protocol in Rust, benchmark the performance and analyze the result. Additionally, we would investigate various ways to add zero-knowledge property into the polynomial commitment scheme and research their advantages and limitations.

We conclude that the efficiency and the soundness error of high dimensional polynomial commitment schemes are not acceptable to be used in practice because of the lack of linear code that is both efficient in encoding and provides a large relative distance guarantee. And we can add zero-knowledge property into the polynomial commitment schemes at the cost of increasing prover time, verifier time, and proof size by roughly a factor of two.

Contents

Chapter 1

Background

In cryptography, a **zero-knowledge proof** is a protocol which allow an untrusted prover $\mathcal{P}$ to convince a sceptical verifier $\mathcal{V}$ that a statement is true without revealing any further information about why the statement is true. Example use-cases include verifiable computing, where a powerful, but untrusted server proves, to a computationally weak client, that they performed a large calculation correctly. The essence of zero-knowledge proofs is that it is trivial to prove that one possesses knowledge of certain information by simply revealing it; the challenge is to prove such possession without revealing the information itself or any additional information.

Efficiency is crucial for large and complex statements especially when we want to deploy those protocols in practice. Important efficiency parameters include but not limited to the time complexity of the prover, the time complexity of the verifier, the amount of communication measured in bits, and the number of rounds the prover and verifier need to interact. In particular, we are interested in the protocol where the prover time is linear to the size of the statement. We call such protocols **linear-time zero-knowledge protocols**.

After years of researching, lots of new proof system models have been introduced. One of the most well-known models is **Interactive proofs (IPs)**. Interactive proofs were introduced by Goldwasser, Micali, and Rackoff in [11] decades ago, in which a probabilistic polynomial-time verifier exchanges k-rounds of messages with an all-powerful prover, and then accepts or rejects the statements.

One of the most famous and the most earliest interactive proof is the quadratic residuosity problem, in which we want to decide whether a is a quadratic residue mod N, given N and a. For an integer N, we say that a ($0 \leq a \leq N-1$) is a quadratic residue mod N if there is an r (a square root) such that $a \equiv r^2 (\texttt{mod}\ N)$.

The interactive proofs turn out to be very powerful, more and more protocols for problems in different areas have been proposed later. There even exists an interactive proof for language that may not rest in NP, for example, graph non-isomorphism (GNI) problem.

Another famous model is **Probabilistically checkable proofs (PCPs)**. Probabilistically checkable proofs were introduced by [9] [1]. In a probabilistically-checkable proof, a probabilistic polynomial-time verifier has oracle access to a proof string and has access to a bounded amount of randomness. The verifier is then required to accept correct proofs and reject incorrect proofs with very high probability.

Comparing to a standard non-interactive proof where the verifier deterministically reads the whole proof, always accepts correct proofs and rejects incorrect proofs, PCPs are interesting because the existence of probabilistically checkable proofs that can be checked by reading only a few bits of the proof using randomness in an essential way.

The number of queries required and the amount of randomness used are important measurements for PCPs. $\texttt{PCP}[r, q]$ is the class of languages for which the verifier uses at most r bits of randomness, and queries at most q locations of the proof. In 1990 Babai, Fortnow, and Lund [2] proved that PCP[poly(n), poly(n)] = NEXP. Later, the PCP theorem (also known as the PCP characterization theorem), a major result in computational complexity theory, states that every decision problem in the NP complexity class has probabilistically checkable proofs of constant query complexity and logarithmic randomness complexity (uses a logarithmic number of random bits). Namely, $\texttt{PCP}[O(\log n), O(1)] = \texttt{NP}$.

Also, though PCPs are protocols purely in theory due to the oracle used in the proofs, researches later have proposed methods [16] to compile PCPs into argument systems that can be implemented in reality, making PCPs impactful not only in theory but also in practice.

Later, **Interactive oracle proofs (IOPs)** was introduced by [3] and [14], which naturally combines aspects of IPs and PCPs and also generalizes interactive PCPs (which consist of a PCP followed by an IP). Namely, an IOP is a "multi-round PCP" that generalizes an interactive proof as follows: the verifier $\mathcal{V}$ has oracle access to the prover $\mathcal{P}$'s messages, and may probabilistically query them (rather than having to read them in full).

In more detail, a k-round IOP comprises k rounds of interaction. In the i-th round of interaction: the verifier $\mathcal{V}$ sends a message m_i to the prover $\mathcal{P}$, which he reads in full; then the prover $\mathcal{P}$ replies with a message f_i to the verifier $\mathcal{V}$, which he can query, as an oracle proof string, in this and all later rounds. After the k rounds of interaction, the verifier $\mathcal{V}$ either accepts or rejects the statement.

IOPs is more powerful because it retain the expressiveness of PCPs, capturing NEXP rather than only PSPACE, allowing reading only a few bits of the proof, and also the flexibility of IPs, allowing multiple rounds of communication with the prover. IOPs have already found several applications, including unconditional zero knowledge, constant-rate constant-query probabilistic checking, and doubly-efficient constant-round IPs for polynomial-time bounded-space computations.

In this thesis, we focus on **polynomial commitment schemes**, initially introduced by [15]. Later, many constructions [18] [20] [21] have been proposed. Polynomial commitment schemes are important building blocks for the construction of Succinct Non-interactive Arguments of Knowledge (SNARks), which is receiving a lot of attention as a core privacy-enhancing technology for blockchain applications recently.

Polynomial commitment schemes enable the prover to commit to a secret polynomial and convince the verifier that the evaluation of the committed polynomial is correct at a public point later. Comparing to the homomorphic commitment schemes in the literature, whose sizes of the commitments are linear in the degree of the committed polynomial, that can be used to achieve the same goal, polynomial commitment schemes are of constant size (single elements) and the overhead of opening a commitment is also constant; even opening multiple evaluations requires only a constant amount of communication overhead. Therefore, polynomial commitment schemes are useful tools to reduce the communication cost in many cryptographic protocols.

In coding theory, a linear code is an error-correcting code for which any linear combination of codewords is also a codeword. The encoding function maps vectors in space $\mathbb{F}^k$ to vectors in space $\mathbb{F}^n$, where $\mathbb{F}$ is a finite field with q elements. The relative distance is defined to be the minimum distance between any valid codewords divided by n, the length of the codeword.

Reed-Solomon code is one of the examples which can encode efficiently using FFT algorithm running in $O(n \log n)$ time. Random linear code is another example. And it is well known that a random linear code, has a good minimal distance with high probability. However, one major disadvantage of random linear codes is that their encoding complexity grows quadratically with the message length.

Those linear codes are important because many protocols rely on special families of error-correcting codes, whose properties influence the final performance of the proof systems. For example, [12] [4] [5] rely on codes with a tensor structure. The lower the dimension of the tensors, the smaller the proof size and verification time of the zero-knowledge proofs.

One of the main research directions is minimizing the encoding complexity of codes. Since the best asymptotic encoding complexity one could hope for is linear in k, the length of input vectors, it is natural to ask whether there are asymptotically good families of **linear-time encodable codes**. The first proof that such codes exist is due to Gelfand, Dobrushin and Pinsker [10], who presented a randomized construction of linear-time encodable linear codes over the binary field which have positive rate and relative minimal distance. An explicit construction of such codes, which also admits a linear-time decoding algorithm, was given in a celebrated work of Spielman [17].

The concrete rate/distance tradeoff achieved by Spielman's codes is far from the GV bound. Guruswami and Indyk [13] construct linear-time encodable codes whose rate and distance parameters can get arbitrarily close to the GV bound. Unfortunately, the closer one wishes to get to the bound, the larger the size of the underlying field becomes. These results leave open the existence of linear-time encodable codes which meet the GV bound, or even get close to this bound in the binary case.

Chapter 2

Preliminaries

2.1 Combinatorics

Definition 2.1 (d-regular Graph) *A graph $G = (V, E)$ is d-regular if every vertex in V has degree d.*

Definition 2.2 (Set) *$[n]$ is the shorthand for the set $\{i : 1 \leq i \leq n\}$*

2.2 Interactive Oracle Proofs

Definition 2.3 (Relation) *A **relation** R is a set of pairs $(\mathbb{X}, \mathbb{W})$ where $\mathbb{X}$ is the instance and $\mathbb{W}$ is the witness. The corresponding **language** $L(R)$ is the set of instances $\mathbb{X}$ for which there exists a witness $\mathbb{W}$ such that $(\mathbb{X}, \mathbb{W}) \in R$.*

Definition 2.4 (Interactive Proof (IP)) *An **Interactive Proof (IP)** is defined by a pair of interactive randomized algorithms **IP = (P, V)**, where **P** denotes the prover and **V** the verifier. The number of rounds of interaction is called the **round complexity** of the system. During a single round, the prover sends a message to the verifier, and the verifier reply a message back to the prover. The **proof length** is the sum of lengths of all messages sent by the prover. We denote by $\langle P \leftrightarrow V\rangle(\mathbb{X}, \mathbb{W})$ the output of **V** after interacting with **P** on instance $\mathbb{X}$ and witness $\mathbb{W}$; this output is either ACCEPT or REJECT.*

An interactive proof IP = (P, V) for a relation R has completeness 1 and soundness error ϵ if the following holds.

1. ***Completeness.*** *For every pair $(\mathbb{X}, \mathbb{W}) \in R$, the probability that $P(\mathbb{X}, \mathbb{W})$ convinces $V(\mathbb{X})$ to accept is 1.*
2. ***Soundness.*** *For every instance $\mathbb{X} \notin L(R)$ and malicious prover $\tilde{P}$, the probability that $\tilde{P}$ convinces $V(\mathbb{X})$ to accept is at most ϵ.*

Definition 2.5 (Point Query Interactive Oracle Proof (IOP)) *An **Interactive Oracle Proof (IOP)** is defined by a pair of interactive randomized algorithms **IOP = (P, V)**, where **P** denotes the prover and **V** the verifier. The number of rounds of interaction is called the **round complexity** of the system. During a single round the prover sends a message to which the verifier is given oracle access, and the verifier responds with a message to the prover. The **proof length** is the sum of lengths of all messages sent by the prover. Specifically, the prover is allowed to send a large array message π to the verifier, and the verifier is allowed to query this message π at position $i \in [N]^t$. The message π will work as an oracle and the verifier will learn $\pi[i]$ through this query. The **query complexity** of the protocol is the number of entries read by **V** from the various prover messages. We denote by $\langle P \leftrightarrow V\rangle(\mathbb{X}, \mathbb{W})$ the output of **V** after interacting with **P** on instance $\mathbb{X}$ and witness $\mathbb{W}$; this output is either* ACCEPT *or* REJECT.

An interactive oracle proof IOP = (P, V) for a relation R has completeness 1 and soundness error ϵ if the following holds.

1. ***Completeness.*** *For every pair $(\mathbb{X}, \mathbb{W}) \in R$, the probability that $P(\mathbb{X}, \mathbb{W})$ convinces $V(\mathbb{X})$ to accept is 1.*
2. ***Soundness.*** *For every instance $\mathbb{X} \notin L(R)$ and malicious prover $\tilde{P}$, the probability that $\tilde{P}$ convinces $V(\mathbb{X})$ to accept is at most ϵ.*

Definition 2.6 (Interactive Oracle Proof of Proximity (IOPP)) *An **Interactive Oracle Proof of Proximity (IOPP)** is defined by a pair of interactive randomized algorithms **IOPP = (P, V)**, where **P** denotes the prover and **V** the verifier. The number of rounds of interaction is called the **round complexity** of the system. During a single round the prover sends a message to which the verifier is given oracle access, and the verifier responds with a message to the prover. The **proof length** is the sum of lengths of all messages sent by the prover. Specifically, the prover is allowed to send a large array message π to the verifier, and the verifier is allowed to query this message π at position I. The message π will work as an oracle and the verifier will learn $\pi[I]$ through this query. The **query complexity** of the protocol is the number of entries read by **V** from the various prover message. We denote by $\langle P \leftrightarrow V\rangle(\mathbb{X}, \mathbb{W})$ the output of **V** after interacting with **P** on instance $\mathbb{X}$ and witness $\mathbb{W}$; this output is either* ACCEPT *or* REJECT. *The protocol's goal is to show that a particular string is close to a valid witness. An interactive oracle proof of proximity IOPP = (P, V) for a relation R has completeness 1 and soundness error ϵ with distance function $\Delta(w_1, w_2) \in \mathbb{F}$ ($w_1, w_2 \in \mathbb{F}^N$) if the following holds.*

1. ***Completeness.*** *For every pair $(\mathbb{X}, \mathbb{W}) \in R$, the probability that $P(\mathbb{X}, \mathbb{W})$ convinces $V^{\mathbb{W}}(\mathbb{X})$ to accept is 1.*
2. ***Soundness.*** *For every instance $\mathbb{X} \notin L(R)$ and malicious prover $\tilde{P}$, the probability that $\tilde{P}$ convince $V^{\mathbb{W}}(\mathbb{X})$ to accept is at most $\epsilon(\Delta(\mathbb{W}, R|_{\mathbb{X}}))$. Here the soundness error ϵ is a function of the Δ-distance of $\mathbb{W}$ to the set of valid witnesses $R|_{\mathbb{X}} := \{\mathbb{W}' | (\mathbb{X}, \mathbb{W}') \in R\}$.*

In practice, we use Merkle tree commitment to compile the IOP or IOPP to an real argument system. Each element in the large array message π sent by the prover will be considered to be a leaf node of a Merkle tree. And the corresponding Merkle tree root will be sent to the verifier instead. For each query at position I, the prover will responds with $\pi[I]$ and the corresponding Merkle tree path, which will be authenticated later by the verifier.

Definition 2.7 *A interactive oracle proof IOP = (**P**, **V**) for a relation R is **semi-honest verifier zero-knowledge** if there exists a polynomial-time simulator algorithm **S** such that, for every $(\mathbb{X}, \mathbb{W}) \in R$ and choice of verifier randomness ρ, the random variables $\boldsymbol{S}^{\boldsymbol{V}(\mathbb{X};\rho)}(\mathbb{X})$ and $View(\boldsymbol{P}(\mathbb{X}, \mathbb{W}), \boldsymbol{V}(\mathbb{X}; \rho))$ are identically distributed.*

Definition 2.8 *A interactive oracle proof of proximity IOPP = (**P**, **V**) for a relation R is **semi-honest verifier zero-knowledge** if there exists a polynomial-time simulator algorithm **S** such that, for every $(\mathbb{X}, \mathbb{W}) \in R$ and choice of verifier randomness ρ, the random variables $\boldsymbol{S}^{\boldsymbol{V}(\mathbb{X};\rho)}(\mathbb{X})$ and $View(\boldsymbol{P}(\mathbb{X}, \mathbb{W}), \boldsymbol{V}(\mathbb{X}; \rho))$ are identically distributed.*

Definition 2.9 *Let A be an algorithm with adaptive query access to oracles $O_1, \cdots, O_n$. Let Q be a stateful query-checker algorithm which receives the adaptive queries of A and may output $\perp$ at any point. We say that A is a Q-query algorithm if Q never outputs $\perp$.*

Definition 2.10 *A interactive oracle proof of proximity IOPP = (**P**, **V**) for a relation R is **perfect zero-knowledge** with query-checker Q if there exists a polynomial-time simulator algorithm **S** such that, for every $(\mathbb{X}, \mathbb{W}) \in R$, Q-query algorithm $\widetilde{V}$ and the choice of verifier randomness ρ, the random variables $\boldsymbol{S}^{\widetilde{V}(\mathbb{X};\rho)}(\mathbb{X})$ and $View(\boldsymbol{P}(\mathbb{X}, \mathbb{W}), \widetilde{V}(\mathbb{X}; \rho))$ are identically distributed.*

2.3 Polynomials

Definition 2.11 (Monomials of Polynomial) *A polynomial g over $\mathbb{F}$ is an expression consisting of a sum of **monomials** where each monomial is the product of a constant (from $\mathbb{F}$) and powers of one or more variables (which take values from $\mathbb{F}$); all arithmetic is performed over $\mathbb{F}$.*

Definition 2.12 (Degree of Polynomial) *The degree of a monomial is the sum of the exponents of variables in the monomial; the (total) degree of a polynomial g is the maximum degree of any monomial in g. Furthermore, the degree of a polynomial g in a particular variable x_i is the maximum exponent that x_i takes in any of the monomials in g.*

Definition 2.13 (Multivariate / Univariate Polynomial) *A **multivariate** polynomial is a polynomial with more than one variable; otherwise it is called a **univariate** polynomial.*

Definition 2.14 (Multilinear Polynomial) *A **multivariate** polynomial is called a **multilinear** polynomial if the degree of the polynomial in each variable is at most one.*

Definition 2.15 (Polynomial Commitment) *A **Polynomial Commitment** is an interactive proof (IP), which defines a relation $R = ((C, x, y), (\phi(\cdot))$. And it consists of three algorithms (*PC.Setup*,* PC.Commit*,* PC.Verify*) and an evaluation protocol* PC.Eval*, where:*

- PC.Setup(λ, d)*: the algorithm outputs public parameters pp for committing to polynomials of degree d. The parameters pp include a specification of an field $\mathbb{F}$.*
- PC.Commit$(pp, \phi(\cdot))$*: the algorithm outputs a commitment $\mathcal{C}$ of the polynomial $\phi(\cdot)$ with degree at most d.*
- PC.Verify$(pp, \phi(\cdot), \mathcal{C})$*: given $\phi(\cdot), \mathcal{C}$, the algorithm checks if $\mathcal{C}$ is a valid commitment for polynomial $\phi(\cdot)$ with degree at most d. The algorithm outputs* accept *or* reject.
- PC.Eval$(\mathcal{P}(\phi(\cdot)), \mathcal{V}(pp, \mathcal{C}, x, y))$ *: this is an interactive protocol between a prover $\mathcal{P}$ who has the polynomial $\phi(\cdot)$ as private input and a verifier $\mathcal{V}$ who has $(\mathcal{R}, x, y)$ as common public input. The purpose of the protocol is to convince the verifier that $\phi(x) = y$ and the degree of $\phi(\cdot)$ is at most d.*

Completeness. Let $\mathcal{C} \leftarrow$ PC.Commit$(pp, \phi(\cdot))$ be the commitment of a polynomial. For all polynomials $\phi(\cdot)$ and all points x, with probability 1 the verification PC.Verify$(pp, \phi(\cdot), \mathcal{C})$ outputs accept. And likewise, $\mathcal{V}$ output accept in interaction with $\mathcal{P}$ in the PC.Eval protocol on valid inputs. The formal completeness requirement is:

$$Pr\begin{pmatrix} b_1 = \textsc{accept} \wedge b_2 = \textsc{accept} : \\ pp \leftarrow \text{PC.}\textsc{Setup}(\lambda, d) \\ \mathcal{C} \leftarrow \text{PC.}\textsc{Commit}(pp, \phi(\cdot)) \\ b_1 \leftarrow \text{PC.}\textsc{Verify}(pp, \phi(\cdot), \mathcal{C}) \\ (\perp, b_2) \leftarrow \text{PC.}\textsc{Eval}(\mathcal{P}(\phi(\cdot)), \mathcal{V}(pp, \mathcal{C}, x, y)) \end{pmatrix} = 1$$

Binding. For all adversaries $\mathcal{A}$, the binding error ϵ_{bind} is defined to be the following probability:

$$\epsilon_{bind} = Pr\left(\begin{array}{c} b_1 = \textsc{accept} \wedge \\ b_2 = \textsc{accept} \wedge \\ \phi(\cdot) \neq \phi'(\cdot) : \\ pp \leftarrow \text{PC.}\textsc{Setup}(\lambda, d) \\ (\mathcal{C}, \phi(\cdot), \phi'(\cdot)) \leftarrow \mathcal{A}(x) \\ b_1 \leftarrow \text{PC.}\textsc{Verify}(pp, \phi(\cdot), \mathcal{C}) \\ b_2 \leftarrow \text{PC.}\textsc{Verify}(pp, \phi'(\cdot), \mathcal{C}) \end{array}\right)$$

Hiding. The polynomial commitment scheme is hiding if the commitments to distinct polynomials are statistically indistinguishable. Formally speaking, for all adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, the hiding error ϵ_{hide} is defined to be the following probability:

$$\epsilon_{hide} = Pr\left(\begin{array}{c} b = b' : \\ pp \leftarrow \text{PC.}\textsc{Setup}(\lambda, d) \\ (\phi_0(\cdot), \phi_1(\cdot)) \leftarrow \mathcal{A}_0 \\ b \xleftarrow{\$} \{0, 1\} \\ \mathcal{C} \leftarrow \text{PC.}\textsc{Commit}(pp, \phi_b(\cdot)) \\ b' \leftarrow \mathcal{A}_1(\mathcal{C}) \end{array}\right)$$

Soundness. For all malicious prover $\tilde{P}$, the soundness error ϵ_{sound} is defined to be the following probability:

$$\epsilon_{sound} = Pr\left(\begin{array}{c} b = \textsc{accept} \wedge \\ pp \leftarrow \text{PC.}\textsc{Setup}(\lambda, d) \\ \forall \tilde{\mathcal{P}} \\ \mathcal{C} \leftarrow \text{PC.}\textsc{Commit}(pp, \phi(\cdot)) \\ (\perp, b) \leftarrow \text{PC.}\textsc{Eval}(\tilde{\mathcal{P}}(\phi(\cdot)), \mathcal{V}(pp, \mathcal{C}, x, y)) \end{array}\right)$$

Zero-knowledge. At the end of the protocol, the verifier will know the evaluation result of the polynomial at some evaluation points, but nothing more than that. The polynomial commitment scheme is zero-knowledge if the view of the verifier $\mathcal{V}$ generated by the interactive protocol PC.Eval is statistically indistinguishable to the view of the verifier generated by a simulator $\mathcal{S}$. Formally speaking, for all adversaries, the zero-knowledge error ϵ_{zk} is defined to be the following probability:

$$\epsilon_{zk} = Pr\left(\begin{array}{c} b = b' : \\ pp \leftarrow \text{PC.}\textsc{Setup}(\lambda, d) \\ b \xleftarrow{\$} \{0, 1\} \\ \mathcal{C} \leftarrow \text{PC.}\textsc{Commit}(pp, \phi(\cdot)) \\ t_0 \leftarrow \mathcal{S} \\ t_1 \leftarrow \textsc{View}(\text{PC.}\textsc{Eval}(\mathcal{P}(\phi(\cdot)), \mathcal{V}(pp, \mathcal{C}, x, y))) \\ b' \leftarrow \mathcal{A}(t_b, C) \end{array}\right)$$

2.4 Linear Codes

Definition 2.16 (Linear Code) *If $\mathbb{F}$ is a field and $C \subset \mathbb{F}^n$ is a subspace of $\mathbb{F}^n$ then C is said to be a linear code.*

The weight of a codeword is the number of its elements that are nonzero and the distance between two codewords is the **Hamming distance** between them, that is, the number of elements in which they differ. The distance d of the linear code is the minimum weight of its nonzero codewords, or equivalently, the **minimum distance** between distinct codewords. A linear code of length n, dimension k, and distance d is called an $[n, k, d]$ code.

As C is a subspace, there exists a basis $c_1, c_2, \cdots, c_k$ where k is the dimension of the subspace. Any codeword can be expressed as the linear combination of these basis vectors. We can write these vectors in matrix form as the rows of a $k \times n$ matrix. Such a matrix is called a **generator matrix**.

Additionally, any linear combination of valid codeword is also a valid codeword, which is the **linearity property**. Formally speaking, given $x_1, x_2, \cdots, x_n$ are valid codeword, and given $r_1, r_2, \cdots, r_n$ are some constants, $x' = r_1x_1 + r_2x_2 + \cdots + r_nx_n$ is also a valid codeword.

Definition 2.17 (Tensor Product Code) *The tensor product code $C^{\otimes t}$ is the linear code in $\mathbb{F}^{n^t}$ with message length k^t, block length n^t and distance d^t where any axis-parallel line of elements is in C.*

Definition 2.18 (Relation $R_\otimes$) *The relation $R_\otimes$ is the sets of tuples*

$$(\mathbb{X}, \mathbb{W}) = ((\mathbb{F}, C, l, q, t), (c_0^0, \{c_1^{(s)}\}_s, \cdots, \{c_{t-1}^{(s)}\}_s))$$

such that $c_0^0 \in (C^{\otimes t})^l$ and for all $r \in [t-1]$ and $s \in [q]$, we have $c_r^{(s)} \in (C^{\otimes t-r})^k$.

Definition 2.19 (Relation R_{cons}) *The relation R_{cons} is the sets of tuples*

$$(\mathbb{X}, \mathbb{W}) = ((\mathbb{F}, C, l, q, t, \{q^{(s)}\}), c)$$

such that $c = \textsc{Enc}_{C^{\otimes t}}(f) \in \mathbb{F}^{l\cdot n^t}$ for some $f \in \mathbb{F}^{l\cdot k^t}$, for each $s \in [q]$, $q^{(s)} = (q_0^{(s)}, \cdots, q_t^{(s)}) \in \mathbb{F}^l \times (\mathbb{F}^k)^t$, and for all $s \in [q]$, $\langle \otimes_i q_i^{(s)}, f \rangle = v^{(s)}$.

Definition 2.20 (Distance $\Delta_\otimes$) *Let $\mathbb{W} = (c_0^0, \{c_1^{(s)}\}_s, \cdots, \{c_{t-1}^{(s)}\}_s)$ be such that $c_0^{(0)} \in \mathbb{F}^{l\cdot n^t}$ and, for all $r \in [t-1]$ and $s \in [q]$, we have $c_r^{(s)} \in \mathbb{F}^{k\cdot n^{t-r}}$. Given $\mathbb{X} = (\mathbb{F}, C, l, q, t)$, the $\Delta_\otimes$ distance of $\mathbb{W}$ to $R_\otimes|_\mathbb{X}$ is*

$$\Delta_\otimes(\mathbb{W}, R_\otimes|_\mathbb{X}) := \max\{\Delta_0, \Delta_1, \cdots, \Delta_{t-1}\}$$

where $\Delta_0 := \Delta(c_0^{(0)}, C^{\otimes t})$ and $\forall r \in [t-1], \Delta_r := \Delta(\{c_r^{(s)}\}_s, C^{\otimes t-r})$.

Chapter 3

Polynomial Commitment

In this chapter, we present a general polynomial commitment scheme in the language of IOP for arbitrary dimension t. The scheme is an extension of the polynomial commitment scheme for $t = 2$ described in [12]. We first extend the scheme to the $t = 3$ situation so that readers can have a good intuition on how it works. Then we generalize it to arbitrary t with detailed analysis available.

3.1 Notation

Let g be a multilinear polynomial with n coefficients. For simplicity we assume that $n = m^t$ for some integer m. And let u denote the coefficient vector of g in the Lagrange basis, which means u represents all evaluations of g over inputs in hypercube $\{0,1\}^{\log n}$. We can rearrange u to be a $\underbrace{n^{\frac{1}{t}} \times n^{\frac{1}{t}} \times \cdots \times n^{\frac{1}{t}}}_{t \text{ times}}$ matrix, such that we can index entries in this matrix easily by elements from set $[m]^t$.

Let $N = \rho^{-1} \cdot m$ and Enc: $\mathbb{F}^m \to \mathbb{F}^N$ represent the encoding function of a linear code with a constant rate $\rho > 0$ and a constant minimum relative distance $\gamma > 0$.

Let $\mathrm{Enc}_i(M)$ denote the function that encode every stripes in the ith dimension of matrix M using encoding function Enc. For example, $\mathrm{Enc}_1(M)$ will encode each column of a $n \times n$ matrix and produce a $N \times m$ matrix.

Lemma 3.1 (Polynomial Evaluation [12]) *For an l-variate multilinear polynomial g represented in the Lagrange basis via a vector $u \in \mathbb{F}^n$ where $2^l = n$, given an evaluation point $x \in \mathbb{F}^l$, $g(x)$ can be evaluated using the following tensor product identity:*

$$g(x) = \langle (x_1, 1 - x_1) \otimes (x_2, 1 - x_2) \otimes \cdots \otimes (x_l, 1 - x_l), u \rangle$$

And for any $1 \leq t \leq l$, there always exist vectors $q_1, q_2, \cdots, q_t \in \mathbb{F}^{n^{\frac{1}{t}}}$ such that the following holds:

$$(x_1, 1 - x_1) \otimes (x_2, 1 - x_2) \otimes \cdots \otimes (x_l, 1 - x_l) = q_1 \otimes q_2 \otimes \cdots \otimes q_t$$

3.2 Proximity Test for Arbitrary t

Proximity test is the core component of the polynomial commitment scheme, which will test whether $(\mathbb{X}, \mathbb{W})$ is in relation $R_{\otimes}$ (definition 2.18). The purpose of this protocol is to convince the verifier $\mathcal{V}$ that a matrix M is very close to a valid tenser code $C^{\otimes t}$.

3.2.1 Formal Description

Prover $\mathcal{P}$'s input:

$$M_0 \in \mathbb{F}^{\overbrace{m \times m \times \cdots \times m}^{t \text{ times}}}$$

$$M_0' = \text{Enc}_1 \circ \text{Enc}_2 \circ \cdots \circ \text{Enc}_{t-1}(M_0) \in \mathbb{F}^{\overbrace{N \times N \times \cdots \times N}^{t-1 \text{ times}} \times m}$$

Verifier $\mathcal{V}$'s input: nothing.

In high level, the protocol consists of $t - 1$ rounds, with each round reducing the dimension by 1. The protocol proceeds as follows.

- $\mathcal{P}$ sends M_0' to $\mathcal{V}$.
- Round i for $i \in [t - 1]$
  - $\mathcal{V}$ sample a random variable $r_i \in \mathbb{F}^m$ and send r_i to $\mathcal{P}$.
  - $\mathcal{P}$ computes a linear combination $M_i \in \mathbb{F}^{\overbrace{N \times N \times \cdots \times N}^{t-1 \text{ times}} \times m}$ of the last dimension of matrix M_{i-1}. Namely, for $1 \leq j_1, j_2, \cdots, j_{t-i} \leq m$:
  $$M_i[j_1, j_2, \cdots, j_{t-i}] = \sum_{k=1}^{m} r_i[k] \cdot M_{i-1}[j_1, j_2, \cdots, j_{t-i}, k]$$
  - $\mathcal{P}$ computes
  $$M_i' = \text{Enc}_1 \circ \text{Enc}_2 \circ \cdots \circ \text{Enc}_{t-i-1}(M_i) \in \mathbb{F}^{\overbrace{N \times N \times \cdots \times N}^{t-i-1 \text{ times}} \times m}$$
  and sends M_i' to $\mathcal{V}$.

- $\mathcal{V}$ performs a probabilistic check to make sure M'_0, M'_1, $M'_2, \cdots, M'_{t-1}$ are consistent with each other. Formally speaking, $\mathcal{V}$ will sample l random tuple $(j_1, j_2, \cdots, j_t)$ from space $\overbrace{[N] \times [N] \times \cdots \times [N]}^{t \text{ times}}$. For each tuple $(j_1, j_2, \cdots, j_t)$, $\mathcal{V}$ will check whether the following equation holds for every $i \in [t-1]$:

$$\text{Enc}(M'_i[j_1, j_2, \cdots, j_{t-i-1}, *])[j_{t-i}] \stackrel{?}{=} \sum_{k=1}^{m} r_i[k] \cdot M'_{i-1}[j_1, j_2, \cdots, j_{t-i}, k]$$

3.3 Consistency Test

Let $q_1, q_2, \cdots, q_t \in \mathbb{F}^m$ be vectors such that $g(x) = \langle q_1 \otimes q_2 \otimes \cdots \otimes q_t, u \rangle$. The consistency test is identical to the proximity test, except that in round i, the random linear combination r_i is replaced by q_i. It will test whether $(\mathbb{X}, \mathbb{W})$ is in relation R_{cons} (definition 2.19). The full description of the consistency test is written below.

3.3.1 Formal Description

Prover $\mathcal{P}$'s input:

$$M_0 \in \mathbb{F}^{\overbrace{m \times m \times \cdots \times m}^{t \text{ times}}}$$

$$M'_0 = \text{Enc}_1 \circ \text{Enc}_2 \circ \cdots \circ \text{Enc}_{t-1}(M_0) \in \mathbb{F}^{\overbrace{N \times N \times \cdots \times N}^{t-1 \text{ times}} \times m}$$

Verifier $\mathcal{V}$'s input: $q_1, q_2, \cdots, q_t \in \mathbb{F}^m$ such that $g(x) = \langle q_1 \otimes q_2 \otimes \cdots \otimes q_t, u \rangle$.

In high level, the protocol consists of $t-1$ rounds, with each round reducing the dimension by 1. The protocol proceeds as follows.

- $\mathcal{P}$ sends M'_0 to $\mathcal{V}$.
- Round i for $i \in [t-1]$
  - $\mathcal{V}$ send q_i to $\mathcal{P}$.
  - $\mathcal{P}$ computes a linear combination $M_i \in \mathbb{F}^{\overbrace{N \times N \times \cdots \times N}^{t-1 \text{ times}} \times m}$ of the last dimension of matrix M_{i-1}. Namely, for $1 \leq j_1, j_2, \cdots, j_{t-i} \leq m$:

$$M_i[j_1, j_2, \cdots, j_{t-i}] = \sum_{k=1}^{m} q_i[k] \cdot M_{i-1}[j_1, j_2, \cdots, j_{t-i}, k]$$

  - $\mathcal{P}$ computes

$$M_i' = \text{Enc}_1 \circ \text{Enc}_2 \circ \cdots \circ \text{Enc}_{t-i-1}(M_i) \in \mathbb{F}^{\overbrace{N \times N \times \cdots \times N}^{t-i-1 \text{ times}} \times m}$$

    and sends M_i' to $\mathcal{V}$.

- $\mathcal{V}$ performs a probabilistic check to make sure M_0', M_1', $M_2', \cdots, M_{t-1}'$ are consistent with each other. Formally speaking, $\mathcal{V}$ will sample l random tuple $(j_1, j_2, \cdots, j_t)$ from space $\overbrace{[N] \times [N] \times \cdots \times [N]}^{t \text{ times}}$. For each tuple $(j_1, j_2, \cdots, j_t)$, $\mathcal{V}$ will check whether the following equation holds for every $i \in [t-1]$:

$$\text{Enc}(M_i'[j_1, j_2, \cdots, j_{t-i-1}, *])[j_{t-i}] \overset{?}{=} \sum_{k=1}^{m} q_i[k] \cdot M_{i-1}'[j_1, j_2, \cdots, j_{t-i}, k]$$

3.4 Polynomial Commitment for Arbitrary t

Prover $\mathcal{P}$'s input: $u \in \mathbb{F}^{\overbrace{m \times m \times \cdots \times m}^{t \text{ times}}}$.

Verifier $\mathcal{V}$'s input: $x, y \in \mathbb{F}$.

3.4.1 Protocol

Commitment Phase.

Let $M_0 = u \in \mathbb{F}^{\overbrace{m \times m \times \cdots \times m}^{t \text{ times}}}$ and $M_0' = \text{Enc}_1 \circ \text{Enc}_2 \circ \cdots \circ \text{Enc}_{t-1}(M_0) \in \mathbb{F}^{\overbrace{N \times N \times \cdots \times N}^{t-1 \text{ times}} \times m}$. $\mathcal{P}$ sends M_0' to $\mathcal{V}$.

Evaluation Phase.

Execute the consistency test protocol. The prover $\mathcal{P}$'s input is (M_0, M_0') and the verifier $\mathcal{V}$'s input is $(q_1, q_2, \cdots, q_t)$ such that $g(x) = \langle q_1 \otimes q_2 \otimes \cdots \otimes q_t, u \rangle$. If all consistency checks passed, then the verifier $\mathcal{V}$ will consider $\langle q_t, M_{t-1} \rangle$ as the evaluation result $g(x)$.

Testing Phase.

For each $0 \leq i \leq t-1$, execute the proximity test protocol. The prover $\mathcal{P}$'s input is (M_i, M_i').

If all tests passed, the verifier $\mathcal{V}$ will output the evaluation result. Otherwise, the verifier $\mathcal{V}$ will reject the protocol.

3.5 Analysis

We refer to the result in [4] and summarize to the following lemmas.

Lemma 3.2 *The testing phase (proximity test) has perfect completeness.*

Lemma 3.3 *The testing phase (proximity test) has soundness error:*

$$\epsilon(\Delta_\otimes, t, l) = \frac{d(d^t - 1)}{4(d-1)|\mathbb{F}|} + (1 - \ min\{\frac{\delta^t}{4}, \Delta_\otimes\})^l$$

where $d = \delta \cdot N$, and δ denotes the relative distance.

3.6 Benchmark

3.6.1 Runtime

Dimension	Message Length	Code Length	Commit Time [ms]	Verify Time [ms]	Soundness Error	Communication Complexity [Field Element]
2	1024	1762	41737	3057	0.37	1206579
3	101	174	99642	623	1.76	235621
4	32	56	153558	204	1.98	114701

Table 3.1: Runtime of polynomial commitment scheme with 2^{20} coefficients, 1 threads, linear code with relative distance 0.07, and 1000 test tuples.

We benckmark the above polynomial commitment scheme on a computer with Intel ® Core ^{TM} i7-7700HQ CPU @ 2.80GHz (Kabylake), L1 cache: 128KB, L2 cache: 256KB and L3 cache: 6MB. There are 8 physical CPU cores available on this machine. The runtimes are summarized in the table 3.1 and table 3.2.

Running the polynomial commitment scheme with the same setting, using 8-threads-parallelism can provides approximately a 4x speedup.

Dimension	Message Length	Code Length	Commit Time [ms]	Verify Time [ms]	Soundness Error	Communication Complexity [Field Element]
2	1024	1762	10048	776	0.37	1206579
3	101	174	24314	165	1.76	235621
4	32	56	37961	63	1.98	114701

Table 3.2: Runtime of polynomial commitment scheme with 2^{20} coefficients, 8 threads, linear code with relative distance 0.07, and 1000 test tuples.

As the dimension increases, it is generally require more time to complete the commit phase for the prover. And less time is required to complete the verify phase for the verifier. Also high dimensional polynomial commitment scheme will have less communication complexity. However, since the relative distance is decreasing as the tensor code's dimension is increasing, the soundness error will also increase. In fact, the soundness error for 3-dimensional and 4-dimensional polynomial commitment scheme is higher than 1, which is unusable in practice.

3.6.2 Soundness Error

According to lemma 3.3, we can compute the soundness error summarized in the table 3.3.

Dimension	Number of Test Tuples	Code Length	Code Relative Distance	Soundness Error
2	100	1762	0.07	1.66
	1000	1762	0.07	0.37
	100	1762	0.55*	0.0003
3	100	174	0.07	1.97
	1000	174	0.07	1.76
	100	174	0.55*	0.01
4	100	56	0.07	1.99
	1000	56	0.07	1.98
	100	56	0.55*	0.10

Table 3.3: Soundness error of polynomial commitment scheme. (* represents an imaginary linear code with relative distance 0.55)

The theoretically computed soundness error for the setting used in the above benchmark experiment is large, even above 1, making it not usable in practice. The soundness error can be decreased by either increasing the number of tested tuples or by increasing the relative distance of the underlying linear code. However, the soundness error is not sensitive to the number of tested tuples and the length of the code is usually quite limited. Therefore, using a linear code with a large relative distance is the only promising solution here. One of our conclusion would be high dimension polynomial commitment scheme is not worth using unless we can improve the relative distance of these linear codes used in the constructions significantly. However, improving relative distance seems to be a difficult task.

Chapter 4

Simple Zero-Knowledge Polynomial Commitment

In this chapter, we describe a simple method to add the zero-knowledge property to a given polynomial commitment scheme. This method uses random numbers to hide the actual coefficients and it works similarly to one-time pad encryption.

4.1 Proximity Test

Prover $\mathcal{P}$'s input:

$$M_0 \in \mathbb{F}^{\overbrace{m \times m \times \cdots \times m}^{t \text{ times}}}$$

$$M_0' = \text{Enc}_1 \circ \text{Enc}_2 \circ \cdots \circ \text{Enc}_{t-1}(M_0 \oplus PAD_0) \in \mathbb{F}^{\overbrace{N \times N \times \cdots \times N}^{t-1 \text{ times}} \times m}$$

Verifier $\mathcal{V}$'s input: nothing.

In high level, the protocol consists of $t-1$ rounds, with each round reducing the dimension by 1. The protocol proceeds as follows.

- Let $M_0 = u$ and PAD_0 be a tensor with dimensions identical to M_0 filled with random elements from $\mathbb{F}$. Let

  $$M_0' = \text{Enc}_1 \circ \text{Enc}_2 \circ \cdots \circ \text{Enc}_{t-1}(M_0 \oplus PAD_0) \in \mathbb{F}^{\overbrace{N \times N \times \cdots \times N}^{t-1 \text{ times}} \times m}$$

  $$PAD_0' = \text{Enc}_1 \circ \text{Enc}_2 \circ \cdots \circ \text{Enc}_{t-1}(PAD_0) \in \mathbb{F}^{\overbrace{N \times N \times \cdots \times N}^{t-1 \text{ times}} \times m}$$

  where $\oplus$ denotes elements-wise tensor addition. $\mathcal{P}$ sends M_0' and $PAD_0\prime$ to $\mathcal{V}$.

- Round i for $i \in [t-1]$
  - $\mathcal{V}$ sample a random variable $r_i \in \mathbb{F}^m$ and send r_i to $\mathcal{P}$.
  - $\mathcal{P}$ computes a linear combination for $M_i, PAD_i \in \mathbb{F}^{\overbrace{m \times m \times \cdots \times m}^{t-i \text{ times}}}$ of their last dimension. Namely, for $1 \le j_1, j_2, \cdots, j_{t-i} \le m$:

$$M_i[j_1, j_2, \cdots, j_{t-i}] = \sum_{k=1}^{m} r_i[k] \cdot M_{i-1}[j_1, j_2, \cdots, j_{t-i}, k]$$

$$PAD_i[j_1, j_2, \cdots, j_{t-i}] = \sum_{k=1}^{m} r_i[k] \cdot PAD_{i-1}[j_1, j_2, \cdots, j_{t-i}, k]$$

  - $\mathcal{P}$ computes

$$M_i' = \mathrm{Enc}_1 \circ \mathrm{Enc}_2 \circ \cdots \circ \mathrm{Enc}_{t-i-1}(M_i \oplus PAD_i) \in \mathbb{F}^{\overbrace{N \times N \times \cdots \times N}^{t-i-1 \text{ times}} \times m}$$

$$PAD_i' = \mathrm{Enc}_1 \circ \mathrm{Enc}_2 \circ \cdots \circ \mathrm{Enc}_{t-i-1}(PAD_i) \in \mathbb{F}^{\overbrace{N \times N \times \cdots \times N}^{t-i-1 \text{ times}} \times m}$$

  - $\mathcal{P}$ sends M_i' and PAD_i' to $\mathcal{V}$.
- $\mathcal{V}$ will perform a probabilistic check to make sure $M_0', M_1', M_2', \cdots, M_t', PAD_0', PAD_1', PAD_2', \cdots, PAD_t'$ are consistent with each other.

  Formally speaking, in step 1, the verifier will sample l_1 random tuple $(j_1, j_2, \cdots, j_t)$ from space $\underbrace{[N] \times [N] \times \cdots \times [N]}_{t \text{ times}}$. Denote this set of tuples as L_1. For each sampled tuple $(j_1, j_2, \cdots, j_t)$, the verifier will check the following equation holds for every $i \in [t-1]$.

$$\mathrm{Enc}(M_i'[j_1, j_2, \cdots, j_{t-i-1}, *])[j_{t-i}] \stackrel{?}{=} \sum_{k=1}^{m} r_i[k] \cdot M_{i-1}'[j_1, j_2, \cdots, j_{t-i}, k]$$

  Then, in step 2, the verifier will sample another l_2 random tuple $(j_1', j_2', \cdots, j_t')$ from space $\underbrace{[N] \times [N] \times \cdots \times [N]}_{t \text{ times}}$ with the restriction that $j_k' \neq j_k$ for $\forall (j_1, j_2, \cdots, j_t) \in L_1$. Denote this set of tuples as L_2. For each sampled tuple $(j_1', j_2', \cdots, j_t')$, the verifier will check the following equation holds for every $1 \le i \le t-2$.

$$\mathrm{Enc}(PAD_i'[j_1, j_2, \cdots, j_{t-i-1}, *])[j_{t-i}] \stackrel{?}{=} \sum_{k=1}^{m} r_i[k] \cdot PAD_{i-1}'[j_1, j_2, \cdots, j_{t-i}, k]$$

4.2 Formal Description

4.2.1 Notation

Fold Operation

Define $\mathbf{Fold}_i(X, r)$ to be the operation taking a linear combination of X across the i-th dimension according to coefficient r.

Namely, for indexes $j_1, \cdots, j_{i-1}, j_{i+1}, \cdots, j_k \geq 1$:

$$\mathbf{Fold}_i(X, r)[j_1, \cdots, j_{i-1}, j_{i+1}, \cdots, j_k] = \sum_{k=1}^{m} r_i[k] \cdot X[j_1, \cdots, j_{i-1}, k, j_{i+1}, \cdots, j_k]$$

Encode Operation

Define $\mathbf{Enc}_{1,\cdots,i}$ be short-hand for $\mathrm{Enc}_1 \circ \mathrm{Enc}_2 \circ \cdots \circ \mathrm{Enc}_i$.

4.2.2 Proximity Test

In this section, we describe the testing phase in the above protocol formally in terms of a IOPP (interactive oracle proof of proximity) with point queries for the relation $R_{\otimes}(\mathbb{F}, C, m, N, t)$ between a prover $\mathbf{P}$ and a verifier $\mathbf{V}$.

The prover $\mathbf{P}$ takes as input an instance $\mathbb{X} = (\mathbb{F}, C, m, N, t)$ and witness $\mathbb{W} = (M'_0, M'_1, \cdots, M'_{t-1}, PAD'_0, PAD'_1, \cdots, PAD'_{t-1})$. The verifier $\mathbf{V}$ takes as input the instance $\mathbb{X}$.

1. *Interactive phase.*

   In the beginning, $\mathbf{P}$ sends the proof message M'_0 and PAD'_0 computed as:

   $$M_0 = u \in \mathbb{F}^{m^t}$$

   $$M'_0 = \mathbf{Enc}_{1,\cdots,t-1}(M_0 \oplus PAD_0) \in \mathbb{F}^{N^{t-1}\cdot m}$$

   $$PAD'_0 = \mathbf{Enc}_{1,\cdots,t-1}(PAD_0) \in \mathbb{F}^{N^{t-1}\cdot m}$$

   Note that PAD_0 is a matrix with dimension identical to M_0 filled with random elements from $\mathbb{F}$. And $\oplus$ denotes elements-wise matrix addition.

   For each round $i \in [t-1]$:

   - $\mathbf{V}$ sends random challenge message $r_i \in \mathbb{F}^m$.
   - $\mathbf{P}$ sends the proof message M'_i computed as:

   $$PAD_i = \mathbf{Fold}_{t-i+1}(PAD_{i-1}, r_i) \in \mathbb{F}^{m^{t-i}}$$

   $$M_i = \mathbf{Fold}_{t-i+1}(M_{i-1}, r_i) \in \mathbb{F}^{m^{t-i}}$$

$$M_i' = \mathbf{Enc}_{1,\cdots,t-i-1}(M_i \oplus PAD_I) \in \mathbb{F}^{N^{t-i-1}\cdot m}$$

$$PAD_i' = \mathbf{Enc}_{1,\cdots,t-i-1}(PAD_i) \in \mathbb{F}^{N^{t-i-1}\cdot m}$$

2. *Query phase.*

   In step 1, the verifier **V** samples l_1 tuples of the form $(j_1, \cdots, j_t)$ in space $[N]^t$. Denote this set of tuples as L_1. The verifier **V** proceeds as follows for each sampled tuple.

   For each $0 \leq i \leq t-1$, the verifier **V** will query M_i' at $(j_1, \cdots, j_{t-i-1}, j_k)$ for each $j_k \in [m]$.

   Then the verifier **V** will check the following equation for $i \in [t-1]$:

$$\text{Enc}_{t-i}(M_i')[i_1, \cdots, i_{t-i}] \overset{?}{=} \mathbf{Fold}_{t-i+1}(M_{i-1}', r_i)[i_1, \cdots, i_{t-i}] \tag{4.1}$$

   In step 2, the verifier **V** samples l_2 tuples of the form $(j_1', \cdots, j_t')$ in space $[N]^t$ with the restriction that $j_k' \neq j_k$ for $\forall (j_1, j_2, \cdots, j_t) \in L_1$. Denote this set of tuples as L_2. The verifier **V** proceeds as follows for each sampled tuple.

   For each $0 \leq i \leq t-1$, the verifier **V** will query PAD_i' at $(j_1', \cdots, j_{t-i-1}', j_k')$ for each $j_k' \in [m]$.

   Then the verifier **V** will check the following equation for $i \in [t-2]$:

$$\text{Enc}_{t-i}(PAD_i')[i_1, \cdots, i_{t-i}] \overset{?}{=} \mathbf{Fold}_{t-i+1}(PAD_{i-1}', r_i)[i_1, \cdots, i_{t-i}] \tag{4.2}$$

4.2.3 Proximity Test Completeness

Lemma 4.1 *IOPP = (**P**, **V**) has **perfect completeness**.*

Proof We begin by noting that the queries made by **V** suffice to perform the checks in the query phase (see equation 4.1 and 4.2).

Next, observe that the verifier **V** checks the following equation:

$$\text{Enc}_{t-i}(M_i') \overset{?}{=} \mathbf{Fold}_{t-i+1}(M_{i-1}', r_i)$$

Note that the left side of this equation is equivalent to:

$$\begin{aligned}
\text{Enc}_{t-i}(M_i') &= \text{Enc}_{t-i}(\mathbf{Enc}_{1,\cdots,t-i-1}(M_i \oplus PAD_i)) \\
&= \mathbf{Enc}_{1,\cdots,t-i}(M_i \oplus PAD_i) \\
&= \mathbf{Enc}_{1,\cdots,t-i}(\mathbf{Fold}_{t-i+1}(M_{i-1} \oplus PAD_{i-1}, r_i)) \qquad (4.3) \\
& \qquad (4.4)
\end{aligned}$$

And the right side of this equation is equivalent to:

$$\mathbf{Fold}_{t-i+1}(M'_{i-1}, r_i) = \mathbf{Fold}_{t-i+1}(\mathbf{Enc}_{1,\cdots,t-i}(M_{i-1} \oplus PAD_{i-1}), r_i) \tag{4.5}$$

$$\tag{4.6}$$

□

Since both **Fold** and **Enc** operations are linear operation, expression 4.3 and expression 4.5 are equivalent to each other. And similar argument applied to the equation 4.2. The equations checked by the verifier **V** holds.

4.2.4 Proximity Test Soundness

Lemma 4.2 *IOPP = (**P**, **V**) has soundness error at most:*

$$\epsilon_{ZK}(\Delta_\otimes, t, l_1, l_2) = \epsilon(\Delta_\otimes, t, l_1) + \frac{\epsilon(\Delta_\otimes, t, l_2)}{\epsilon(\Delta_\otimes, 2, l_2)}$$

Proof This protocol performs two proximity tests in parallel. One on M'_i tensor and the other on PAD_i tensor. The soundness error would be the sum of the soundness error introduced by the first proximity test and the second proximity test.

Formally speaking, suppose

$$((\mathbb{F}, C, m, N, t), (M'_0, M'_1, \cdots, M'_{t-1}, PAD'_0, PAD'_1, \cdots, PAD'_{t-1}))$$

is not in relation $R_\otimes$. Then either $((\mathbb{F}, C, m, N, t), (M'_0, M'_1, \cdots, M'_{t-1}))$ is not in relation $R_\otimes$, or $((\mathbb{F}, C, m, N, t), (PAD'_0, PAD'_1, \cdots, PAD'_{t-1}))$ is not in relation $R_\otimes$.

If $((\mathbb{F}, C, m, N, t), (M'_0, M'_1, \cdots, M'_{t-1}))$ is not in relation $R_\otimes$, then, the soundness error introduced by this part is $\epsilon(\Delta_\otimes, t, l_1)$.

If $((\mathbb{F}, C, m, N, t), (PAD'_0, PAD'_1, \cdots, PAD'_{t-1}))$ is not in relation $R_\otimes$, then, the soundness error introduced by this part is $\frac{\epsilon(\Delta_\otimes, t, l_2)}{\epsilon(\Delta_\otimes, 2, l_2)}$.

In a complete proximity test, we use E_{last} denote the event that the last round of test is passed. And we use E_{other} denote the event that all other tests are passed. The soundness error is the probability the verifier is convinced by a malicious input. The soundness error of a complete proximity test is $P_t = \epsilon(\Delta_\otimes, t, l_2)$. And it is also the probability where both event E_{last} and event E_{other} occurs. Therefore, $P_t = P_{E_{last}} \cdot P_{E_{last}}$. Note that $P_{E_{last}}$ is actually the soundness error when $t = 2$, namely, $P_{E_{last}} = \epsilon(\Delta_\otimes, 2, l_2)$. And $P_{E_{other}}$ is the soundness error introduced by the second proximity test here, where the input is malicious and all tests except the last one are passed. Therefore, $P_{E_{other}} = \frac{P_t}{P_{E_{last}}} = \frac{\epsilon(\Delta_\otimes, t, l_2)}{\epsilon(\Delta_\otimes, 2, l_2)}$.

4.2.5 Proximity Test Zero-Knowledge

Lemma 4.3 *IOPP = (**P**, **V**) is **perfect zero-knowledge***

Proof For every $(\mathbb{X}, \mathbb{W}) \in R_{\otimes}$ and choice of verifier randomness ρ, we can construct the polynomial-time simulator algorithm **S** as follows:

- Generate matrix M_0 and PAD_0 randomly from field $\mathbb{F}$. Then compute M'_0 and PAD'_0 as follows:

$$M'_0 = \mathbf{Enc}_{1,\cdots,t-1}(M_0) \in \mathbb{F}^{N^{t-1}\cdot m}$$

$$PAD'_0 = \mathbf{Enc}_{1,\cdots,t-1}(PAD_0) \in \mathbb{F}^{N^{t-1}\cdot m}$$

- Then compute M'_i and PAD'_i for $i \in [t-1]$:

$$PAD_i = \mathbf{Fold}_{t-i+1}(PAD_{i-1}, r_i) \in \mathbb{F}^{m^{t-i}}$$

$$M_i = \mathbf{Fold}_{t-i+1}(M_{i-1}, r_i) \in \mathbb{F}^{m^{t-i}}$$

$$M'_i = \mathbf{Enc}_{1,\cdots,t-i-1}(M_i) \in \mathbb{F}^{N^{t-i-1}\cdot m}$$

$$PAD'_i = \mathbf{Enc}_{1,\cdots,t-i-1}(PAD_i) \in \mathbb{F}^{N^{t-i-1}\cdot m}$$

If the verifier query M'_i or PAD'_i at index $I = (j_1, j_2, \cdots, j_t)$:

- If $j_k \leq m$ for $\forall k \in [t]$ (this is the message part),

  In the simulation world, both $M'_0[I]$ and $PAD'_0[I]$ are uniformly random variables. And both $M'_i[I]$ and $PAD'_i[I]$ for $i > 0$ are linear combination of a set of uniformly random variables, which are also uniformly random variables.

  In the real world, $PAD'_0[I]$ is a uniformly random variable by definition. $M'_0[I]$ is also a uniformly random variable because $M'_0[I] = u[I] + PAD'_0[I]$. Similarly, both $M'_i[I]$ and $PAD'_i[I]$ for $i > 0$ are linear combination of a set of uniformly random variables, which are also uniformly random variables.

  Therefore, the verifier will see a uniformly distributed random element from $\mathbb{F}$ both in simulation world and in real world.

- Otherwise,

  In the simulation world and in the real world, $M'_i[I]$ can be determined by a set of random elements in M_i. Denote this computation equation as Func, namely, $M'_i[I] = \text{Func}(M_i[I_1], \cdots, M_i[I_x])$. And $M'_i[I]$ will represent a distribution that is uniquely determined by function Func and the distribution of variables $M_i[I_1], \cdots, M_i[I_x]$. Similarly, $PAD'_i[I]$

will represent a distribution that is uniquely determined by function Func and the distribution of variables $PAD_i[I_1], \cdots, PAD_i[I_x]$.

Note that both in simulation world and in the real world, $M_i[I_1], \cdots, M_i[I_x]$ and $PAD_i[I_1], \cdots, PAD_i[I_x]$ will represent uniformly random variables. And since the function Func is identical in both cases, distribution of $M_i'[I]$ and $PAD_i'[I]$ will be identical in two worlds.

The random variables in $\mathbf{S}^{\mathbf{V}(\mathbb{X};\rho)}(\mathbb{X})$ and in $\text{View}(\mathbf{P}(\mathbb{X}, \mathbb{W}), \mathbf{V}(\mathbb{X}; \rho))$ are indistinguishable to each other. They are identically distributed.

Note that although PAD_i and M_i are correlated (the subtraction of them is the underlying polynomial coefficients), the verifier $\mathcal{V}$ will not be able to observe this correlation because verifier is not allowed to query both PAD_i and M_i at the same index. The verifier is only allowed to query one of them.

Chapter 5

Brakedown Linear Code

We use the practical linear code presented in paper [12] to implement and benchmark our polynomial commitment schemes.

5.1 Notation

Let $0 < \alpha < 1$ and $0 < \beta < \frac{\alpha}{1.28}$ be parameters with no explicit meanings. r denotes the ratio between the length of codeword and the length of input message. δ denotes the relative distance. n is the length of encoded message. Let q be a prime power and $\mathbb{F}_q$ be the field of size q. And for $p \in [0,1]$, we denote the binary entropy function as $H(p) = -p\log_2(p) - (1-p)\log_2(1-p)$. Let $\mathcal{M}_{n,m,d}$ be a distribution of matrices $M \in \mathbb{F}^{n\times m}$, where in each row d distinct uniformly random elements are assigned uniformly random non-zero elements of $\mathbb{F}$.

5.2 Construction

The encoding function $\mathbf{Enc}_n$ works as follows. First we generate a random sparse matrix $A \leftarrow \mathcal{M}_{n,\alpha n,c_n}$ for

$$c_n = \left\lceil \min\Big(\max(1.28\beta n, \beta n + 4), \frac{1}{\beta\log_2\frac{\alpha}{1.28\beta}}(\frac{110}{n} + H(\beta) + \alpha H(\frac{1.28\beta}{\alpha}))\Big)\right\rceil$$

And compute $y = x \cdot A \in \mathbb{F}^{\alpha n}$. Then we apply $\mathbf{Enc}$ function recursively to y, let $z = \mathbf{Enc}_{\alpha n}(y) \in \mathbb{F}^{\alpha r n}$. Finally, we generate a random sparse matrix $B \leftarrow \mathcal{M}_{\alpha r n,(r-1-r\alpha,d_n)}$ for

$$d_n = \left\lceil \min\Big((2\beta + \frac{(r-1)+\frac{110}{n}}{\log_2 q})n, \frac{r\alpha H(\frac{\beta}{r}) + \mu H(\frac{\nu}{\mu}) + \frac{110}{n}}{\alpha\beta\log_2\frac{\mu}{\nu}}\Big)\right\rceil$$

$$\mu = r - 1 - r\alpha$$

$$\nu = \beta + \alpha\beta + 0.03$$

Let $v = z \cdot B \in \mathbb{F}^{(r-1-r\alpha)n}$ The resulting codeword is the concatenation of x, z and v.

$$w = \mathbf{Enc}(x) := \begin{pmatrix} x \\ z \\ v \end{pmatrix} \in \mathbb{F}^{rn}$$

5.3 Theoretical Limits for Relative Distance

In Brakedown paper [12], there are a few explicit constrains for parameter α, β and r. And since binary entropy function used in the linear code is only well-defined between 0 and 1, there is also one more implicit constrain. The full list of constrains are as follows,

$$\begin{aligned} & 0 < \alpha < 1 \\ & 0 < \beta < \frac{\alpha}{1.28} && (5.1) \\ & r > \frac{1+2\beta}{1-\alpha} > 1 && (5.2) \\ & \delta = \frac{\beta}{r} && (5.3) \\ & \beta + \alpha\beta + 0.03 < r - 1 - r\alpha && (5.4) \\ & && (5.5) \end{aligned}$$

Combine constrain 5.3 and constrain 5.1, we have,

$$\alpha > 1.28 \cdot \delta \cdot r \tag{5.6}$$

Combine constrain 5.3 and constrain 5.2, we have,

$$\alpha > 1 - 2\delta - \frac{1}{r} \tag{5.7}$$

Combine constrain 5.3 and constrain 5.4, we have,

$$\alpha < \frac{r(1-\delta) - 1.03}{r(1+\delta)} \tag{5.8}$$

To make sure α has a valid value, we have,

$$\frac{r(1-\delta)-1.03}{r(1+\delta)} > 1.28 \cdot \delta \cdot r \tag{5.9}$$

$$\frac{r(1-\delta)-1.03}{r(1+\delta)} > 1 - 2\delta - \frac{1}{r} \tag{5.10}$$

$$\tag{5.11}$$

Equation 5.9 and equation 5.10 make the maximum possible relative distance δ to be around 0.12.

Chapter 6

Zero-Knowledge Linear Code

In this chapter, we use the construction presented in paper [8] to add zero-knowledge property to a normal linear code through code transformation.

6.1 Random d-regular Bipartite Graph

First, we present a algorithm to generate a random d-regular bipartite graph. To make sure each vertex has degree d, we can first sample d random perfect matching for 2 sets of n vertices. Then take the union of them. Note that it is possible to generate parallel edges. But this should not be a concern for our purpose here. And it can be shown that this happens with low probability.

Algorithm 1: Random d-regular Bipartite Graph Generation

```
  Data: $n \geq 0$, $d <= n$
  Result: A random $d$-regular bipartite graph $G = (L, R, E)$ with
          $|L| = |R| = n$
  $L \leftarrow$ a set of $n$ nodes;
  $R \leftarrow$ a set of $n$ nodes;
  $E \leftarrow \emptyset$;
  $P \leftarrow [1, 2, \cdots, n]$;
  for $i$ in $1, 2, \cdots, d$ do
      Permute $P$ randomly ;        /* sample a perfect matching */
      for $j$ in $1, 2, \cdots, n$ do
          $E \leftarrow E \cup (L_j, R_{P_j})$ ;
      end
  end
  return (L, R, E)
```

6.2 Expander Graph

Lemma 6.1 *For any $0 < \epsilon < 1$, there exist a degree d such that a random d-regular bipartite graph $G = (L, R, E)$ with $|L| = |R| = n$ generated according to algorithm 1 satisfy the following property with high probability.*

- *Expansion: For every set $X \subset L$ with $|X| \geq \epsilon n$, if Y is the set of neighbors of X in G, then $|Y| \geq (1-\epsilon)n$.*

Proof Negating the statement, we can say that the randomly generated graph G does not satisfy the expansion property if and only if $\exists S \subseteq L$, $|S| \geq \epsilon n$, $\exists M \subseteq R$, $|M| \geq \epsilon n$ such that there is no edge connecting between set S and set M. We bound the probability that this negating statement is true as follows:

For every vertex $a \in L$ and every vertex $b \in R$, the probability that a and b are not connected in the random graph G is:

$$P_1 = (\frac{n-1}{n})^d$$

For a set of vertices $S \subset L$ with $|S| = s \geq \epsilon n$, the probability that non of vertices in S is connected to b is:

$$P_2 = (P_1)^s = (\frac{n-1}{n})^{ds}$$

The probability that there exists at least ϵn vertices in R are not connected to any vertex in S is:

$$P_3 = \binom{n}{\epsilon n}(P_2)^{\epsilon n} = \binom{n}{\epsilon n}(\frac{n-1}{n})^{ds\epsilon n}$$

For $0 \leq x \leq 1$, we denote the binary entropy function to be:

$$H(x) = -x\log_2 x - (1-x)\log_x(1-x)$$

where we adopt the convention that $0\log_2 0 = 0$.

Then, we take a union bound over all possible sets S,

$$
\begin{aligned}
P_4 &= \sum_{s=\epsilon n}^{n} \binom{n}{s} P_3 \\
&= \sum_{s=\epsilon n}^{n} \binom{n}{s} \binom{n}{\epsilon n} (\frac{n-1}{n})^{ds\epsilon n} \\
&\leq \sum_{s=\epsilon n}^{n} \binom{n}{s} \binom{n}{\epsilon n} (\frac{n-1}{n})^{d\epsilon^2 n^2} && \text{since } s \geq \epsilon n \text{ and } \frac{n-1}{n} < 1 \\
&\leq \sum_{s=\epsilon n}^{n} \binom{n}{s} 2^{nH(\frac{\epsilon n}{n})} (\frac{n-1}{n})^{d\epsilon^2 n^2} && \binom{n}{k} \leq 2^{nH(\frac{k}{n})} \\
&= \sum_{s=\epsilon n}^{n} \binom{n}{s} 2^{nH(\epsilon)} ((1-\frac{1}{n})^n)^{d\epsilon^2 n} \\
&\leq \sum_{s=\epsilon n}^{n} \binom{n}{s} 2^{nH(\epsilon)} (\frac{1}{e})^{d\epsilon^2 n} && (1-\frac{1}{x})^x \leq \frac{1}{e} \text{ for } x \geq 1 \text{ (lemma A.2)} \\
&= \sum_{s=\epsilon n}^{n} \binom{n}{s} (e^{H(\epsilon)\ln 2 - d\epsilon^2})^n \\
&\leq \sum_{s=0}^{n} \binom{n}{s} (e^{H(\epsilon)\ln 2 - d\epsilon^2})^n \\
&= 2^n (e^{H(\epsilon)\ln 2 - d\epsilon^2})^n && \sum_{i=0}^{n} \binom{n}{i} = 2^n \\
&= (e^{\ln 2 + H(\epsilon)\ln 2 - d\epsilon^2})^n
\end{aligned}
\tag{6.1}
$$

□

P_4 is the probability that a randomly generated graph G does not satisfy the expansion property. Suppose we want the failing probability be smaller than p, let $(e^{\ln 2 + H(\epsilon)\ln 2 - d\epsilon^2})^n < p$. By rearranging the above equation, we have $d > \frac{\ln 2 + H(\epsilon)\ln 2 - \frac{\ln p}{n}}{\epsilon^2}$.

For example, if $\epsilon = 0.05$, $n = 5000$, $p = 2^{-256}$, then degree d need to be greater than 370.86.

Lemma 6.2 *For any $0 < \epsilon < 1$, there exist a degree d such that a random d-regular bipartite graph $G = (L, R, E)$ with $|L| = |R| = n$ generated according to algorithm 1 satisfy the following property.*

- *Expansion: For every set $X \subset L$ with $|X| \geq \epsilon n$, if Y is the set of neighbors of X in G, then $|Y| \geq (1-\epsilon)n$ with high probability.*

Proof We use the same trick as in lemma 6.1. Negating the statement, we can say that the randomly generated graph G does not satisfy the expansion property if and only if for every $S \subseteq L$, $|S| \geq \epsilon n$, $\exists M \subseteq R$, $|M| > \epsilon n$ such that there is no edge connecting between set S and set M with low probability. We bound the probability true as follows:

For every vertex $a \in L$ and every vertex $b \in R$, the probability that a and b are not connected in the random graph G is:

$$P_1 = (\frac{n-1}{n})^d$$

For a set of vertices $S \subset L$ with $|S| = \epsilon n$, the probability that non of vertices in S is connected to b is:

$$P_2 = (P_1)^{\epsilon n} = (\frac{n-1}{n})^{d\epsilon n}$$

The probability that there exists at least ϵn vertices in R are not connected to any vertex in S is:

$$\begin{aligned}
P_3 &= \binom{n}{\epsilon n}(P_2)^{\epsilon n} \\
&= \binom{n}{\epsilon n}(\frac{n-1}{n})^{d\epsilon^2 n^2} \\
&\leq 2^{nH(\frac{\epsilon n}{n})}(\frac{n-1}{n})^{d\epsilon^2 n^2} && \binom{n}{k} \leq 2^{nH(\frac{k}{n})} \\
&= 2^{nH(\epsilon)}((1-\frac{1}{n})^n)^{d\epsilon^2 n} \\
&\leq 2^{nH(\epsilon)}(\frac{1}{e})^{d\epsilon^2 n} && (1-\frac{1}{x})^x \leq \frac{1}{e} \text{ for } x \geq 1 \text{ (lemma A.2)} \\
&= (e^{H(\epsilon)\ln 2 - d\epsilon^2})^n
\end{aligned} \tag{6.2}$$

□

P_3 is the probability that a set S in a randomly generated graph does not satisfy the expansion property. Suppose we want the failing probability be smaller than p, let $(e^{H(\epsilon)\ln 2 - d\epsilon^2})^n < p$. By rearranging the above equation, we have $d > \frac{H(\epsilon)\ln 2 - \frac{\ln p}{n}}{\epsilon^2}$.

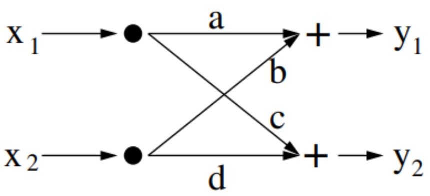
For example, if $\epsilon = 0.05$, $n = 5000$, $p = 2^{-256}$, then degree d need to be greater than 93.60.

Compared with lemma 6.1, lemma 6.2 produces a much tighter bound by weakening the expansion property. A graph satisfy the expansion property in lemma 6.2 may not satisfy the expansion property in lemma 6.1. There may exist a set $S \subset L$ in graph such that the expansion property fails. But lemma 6.2 guarantees that such set is hard to found. Similar with hash functions, hash collision must exist somewhere, but this collision is hard to be found.

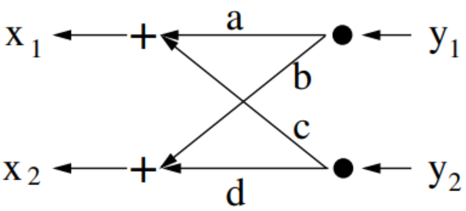
6.3 Reversed Linear Code

The transposition principle, sometimes referred to as Tellegen's principle [7], asserts that a linear algorithm that performs a matrix-vector product can be transposed, producing an algorithm that computes the transposed matrix-vector product. Further, the transposed algorithm has almost the same complexity as the original one.

The following example illustrates this principle, using the computation graph representation, where • represents a fan-out gate and + represents a addition gate. Taking x_1, x_2 as input, it computes $y_1 = ax_1 + bx_2$, $y_2 = cx_1 + dx_2$; edges perform multiplications by the constant values a, b, c, d.



Reversing all arrows and exchanging vertices + and • yield the following graph:



Taking y_1, y_2 as input, it computes the transposed map $x_1 = ay_1 + cy_2, x_2 = by_1 + dy_2$.

In this section, we transpose the Brakedown linear code to get a reverse encoding algorithm. Figure 6.1 is the construction of Brakedown linear code. And figure 6.2 is the reversed construction.

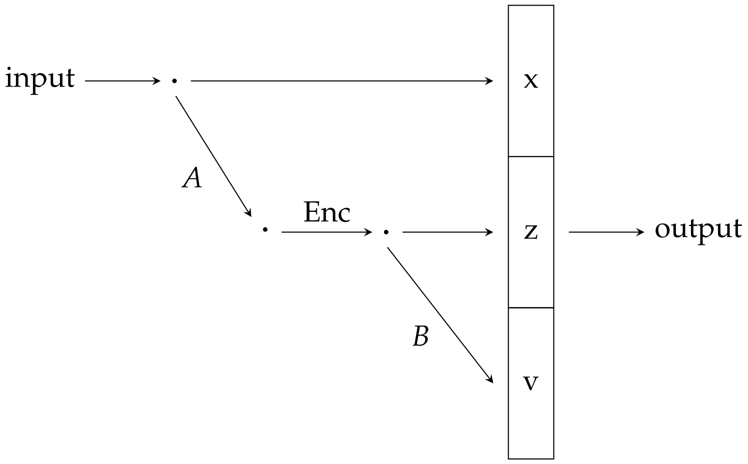


Figure 6.1: Brakedown Linear Code

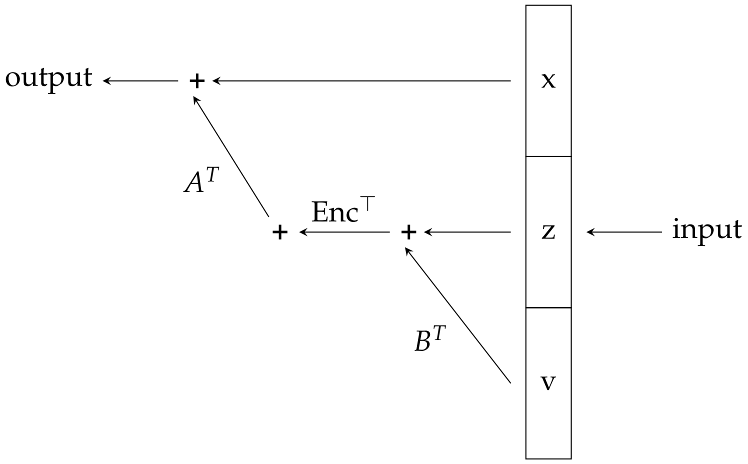


Figure 6.2: Reversed Brakedown Linear Code

6.4 Construction

This construction transform an existing linear to another linear code. The linear code constructed will have better relative distance and is equipped with zero-knowledge property.

6.4.1 Redistribution

Given the normal encoding function **Enc()** and message x, we first compute the codeword $y = \mathbf{Enc}(x) \in \mathbb{F}^n$. Then a random expander graph $G = (L, R, E)$ with degree Δ satisfying lemma 6.1 will be generated. We will redistribute the symbols in y according to G. More concretely, for everty $i \in [n]$ and $j \in [\Delta]$, let $\gamma(i, j)$ be the index of the j-th vertex in R. The $(i-1) \cdot \Delta + j$-th entry of z is defined to be the $y_{\gamma(i,j)}$.

6.4.2 Randomization

Given $z \in \mathbb{F}^{n \cdot \Delta}$, we generate a random block diagonal matrix H with n blocks each of size $\Delta \cdot \Delta$. We compute $v = H \cdot z \in \mathbb{F}^{n \cdot \Delta}$.

6.4.3 Reverse Encoding

Given the reverse encoding function $\mathbf{Enc}^{\top}$, the final output is $w = \mathbf{Enc}^{\top}(v)$.

6.5 Performance

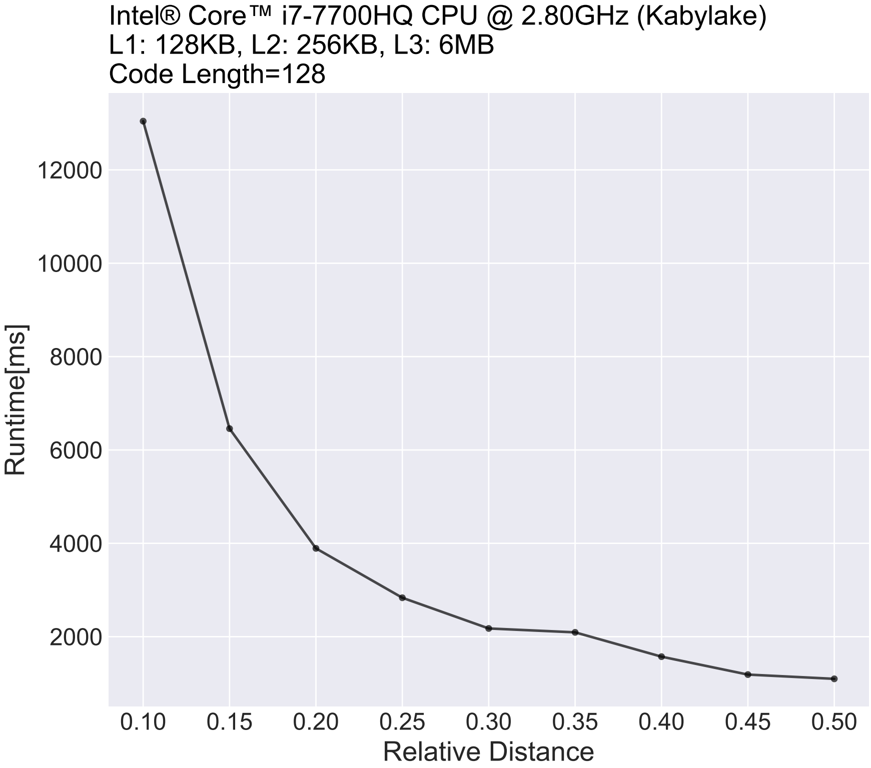


Figure 6.3: Runtime of Redistribution and Randomization Step

We have implement the above construction. We measure the runtime required for the redistribution step and the randomization step, whose execution is irrelevant to the actual underlying linear code.

According to lemma 6.2, the degree of the expander graph is every sensitive to the relative distance of underlying linear code. The larger the relative distance, the smaller the degree. And larger degree will cause the algorithm more time consuming.

Figure 6.3 presents the relation between relative distance and runtime. As relative distance approaches 0.1, the runtime increases dramatically. And even for larger relative distance, the construction is still significant slower than the original linear code, making this construction unacceptable in practice.

6.6 Improvement

The performance of our zero-knowledge linear code construction suffers from the degree of underlying random expander graph. Recently, we notice a idea presented in [19] that might solve this problem. They propose a new algorithm to test whether a random bipartite graph is a lossless expander graph or not based on the densest subgraph algorithm, which helps to sample lossless expanders with an overwhelming probability.

We can prove a graph is not a expander graph by providing one counter example. The densest-subgraph algorihtm used by this detection algorithm can help us to find the counter example efficiently. Basically, if the graph is not a good expander graph, the detection algorithm in [19] can identify this situation with some probability p using a random input r. And if the graph is a good expander graph, the detection algorithm will not falsely identify it (no false positive). Then we can run this detection algorithm λ times with different random inputs and amplify the detection probability to $1 - (1 - p)^{\lambda}$. Additionally, if we find the graph is not a good expander graph, then we can simply discard it and generate a new random graph.

And if we have a efficient detection algorithm like this, we can randomly generate the expander graph with a much smaller degree d and run detection algorithm on it. Depending on the output of detection algorithm, we can either be convinced that it is a good expander graph or we can re-run the generation algorithm one more time. The redistribution step in our construction will be much more efficient with such a small degree expander graph.

Definition 6.3 *Let $\delta > 0$ and $0 < \epsilon < 1$ be a constant. Let $G = (L, R, E)$ be a d-regular bipartite graph with $|L| = n$. Graph G is a expander graph if the following expansion property is satisfied:*

- *Expansion: For every set $X \subset L$ with $|X| \leq \frac{\delta n}{d}$, if Y is the set of neighbors of X in G, then $|Y| \geq (1 - \epsilon)d|X|$.*

However, due to the difference in expander graph definition, it is fundamentally not possible to reuse this densest-subgraph based detection algorithm in our project to imporve efficiency. Definition 6.3 is the expander graph used in [19] and lemma 6.1 is the expander graph used in our project. The key difference is their expansion property. Informally speaking, definition

6.3 needs the graph has good expansion property when we choose a small subset of vertices. And lemma 6.1 needs the graph has good expansion property when we choose a large subset of vertices. The counter example found by the densest-subgraph algorithm may have a small subset of vertices. And this is enough to be a good counter example according to definition 6.3, but not enough according to lemma 6.1.

Therefore, it is remain unclear how to improve efficiency on this construction. And using the method similar to one-time-pad encryption, we can add zero-knowledge property into the polynomial commitment scheme. And it is practical and efficient compared with the alternative approach.

Chapter 7

Zero-Knowledge Proofs for LWE

7.1 Introduction

LWE (Learning with error) problem is one of the fundamental lattice problem upon which lots of the lattice-based cryptography rests. LWE states that for a tuple (A, u) it is hard to find a small s and a small e such that $u = As + e$. In this chapter, we explore a protocol in [6] that makes use of polynomials to prove knowledge of s and e with small elements that satisfy:

$$As + e = u$$

Definition 7.1 (Relation R_{LWE}) *The relation R_{LWE} is the sets of tuples*

$$(\mathbb{X}, \mathbb{W}) = ((\mathbb{F}, n, m, A, u), (s, e))$$

such that $A \in \mathbb{F}^{n\times m}$, $u \in \mathbb{F}^n$, $s \in \{-1, 0, 1\}^m$, $e \in \{-1, 0, 1\}^n$ and $As + e = u$.

7.2 LWE Protocol

7.2.1 Formal Description

Prover $\mathcal{P}$'s input: $A \in \mathbb{F}^{n\times m}$, $u \in \mathbb{F}^n$, $s \in \{-1, 0, 1\}^m$ and $e \in \{-1, 0, 1\}^n$ such that $u = As + e$.

Verifier $\mathcal{V}$'s input: $A \in \mathbb{F}^{n\times m}$, $u \in \mathbb{F}^n$.

The protocol proceeds as follows.

- $\mathcal{P}$ samples $t \leftarrow \mathbb{F}^m$ and computes the polynomials:

$$f(X) = tX + s = f_1X + f_0 \tag{7.1}$$

$$d(X) = u - Af(X) = d_1X + d_0 \tag{7.2}$$

  If the prover is honest, $(f_1, f_0) \in (\mathbb{F}^m, \mathbb{F}^m)$ should equal to (t, s) and $(d_1, d_0) \in (\mathbb{F}^n, \mathbb{F}^n)$ should equal to $(-At, u - As)$.

- $\mathcal{P}$ computes the polynomials:

$$\frac{1}{X} f(X) \circ [f(X) - 1^m] \circ [f(X) + 1^m] = v_2 X^2 + v_1 X + v_0 \tag{7.3}$$

$$\frac{1}{X} d(X) \circ [d(X) - 1^n] \circ [d(X) + 1^n] = w_2 X^2 + w_1 X + w_0 \tag{7.4}$$

  where $(v_2, v_1, v_0) \in (\mathbb{F}^m, \mathbb{F}^m, \mathbb{F}^m)$ and $(w_2, w_1, w_0) \in (\mathbb{F}^n, \mathbb{F}^n, \mathbb{F}^n)$.

- $\mathcal{P}$ samples $r_2, r_1, r_0 \leftarrow \mathbb{F}^N$.

- $\mathcal{P}$ computes the encodings:

$$H'_2 = \widetilde{\textsc{Enc}}(H_2, r_2) \in \mathbb{F}^{2N}$$

$$H'_1 = \widetilde{\textsc{Enc}}(H_1, r_1) \in \mathbb{F}^{2N}$$

$$H'_0 = \widetilde{\textsc{Enc}}(H_0, r_0) \in \mathbb{F}^{2N}$$

  where,

$$\begin{aligned} \widetilde{\textsc{Enc}}(H, r) &= (\textsc{Enc}(H) + r, r) \\ H_2 &= f_2, v_2, w_2 \in \mathbb{F}^{2m+n} \quad (f_2 = 0^m) \\ H_1 &= f_1, v_1, w_1 \in \mathbb{F}^{2m+n} \\ H_0 &= f_0, v_0, w_0 \in \mathbb{F}^{2m+n} \end{aligned}$$

- $\mathcal{P}$ sends H'_2, H'_1, H'_0 to $\mathcal{V}$, and $\mathcal{V}$ has point query access to each of these messages.

- $\mathcal{V}$ samples a random challenge $x \leftarrow \mathbb{F}^*$ and sends it to $\mathcal{P}$.

- $\mathcal{P}$ computes $\overline{H} = x^2 H_2 + x H_1 + H_0 \in \mathbb{F}^{2m+n}$.

- $\mathcal{P}$ computes $\bar{r} = x^2 r_2 + x r_1 + r_0 \in \mathbb{F}^{2m+n}$.

- $\mathcal{P}$ sends $\overline{H}$ and $\bar{r}$ to $\mathcal{V}$.

- $\mathcal{V}$ samples an indices set I with λ indices from space $[2N]$ with the restriction that $\forall i_1, i_2 \in I, |i_1 - i_2| \neq N$. Then for each index i, $\mathcal{V}$ will check whether the following equation holds through point queries to H'_2, H'_1 and H'_0.

$$\widetilde{\textsc{Enc}}(\overline{H}, \bar{r})[i] \overset{?}{=} H'_2[i] x^2 + H'_1[i] x + H'_0[i] \tag{7.5}$$

- Let $(\bar{f}, \bar{g}, \bar{h}) \leftarrow \overline{H}$, where $(\bar{f}, \bar{g}, \bar{h}) \in (\mathbb{F}^m, \mathbb{F}^m, \mathbb{F}^n)$.

- $\mathcal{V}$ computes $\bar{d} = u - A\bar{f}$.

- $\mathcal{V}$ will check whether the following equation holds:

$$\overline{g} \overset{?}{=} \frac{1}{x}(\overline{f} \circ [\overline{f} - 1^m] \circ [\overline{f} + 1^m]) \tag{7.6}$$

$$\overline{h} \overset{?}{=} \frac{1}{x}(\overline{d} \circ [\overline{d} - 1^n] \circ [\overline{d} + 1^n]) \tag{7.7}$$

Lemma 7.2 *LWE = ($\mathcal{P}$, $\mathcal{V}$) has* ***perfect completeness****.*

Proof Equation 7.5 is checking whether $\overline{H}$ and $\overline{r}$ is a correct linear combination of H_2', H_1' and H_0'. If the prover $\mathcal{P}$ is honest, it will succeed.

For equation 7.6, because $\mathcal{P}$ computes it honestly according to equation 7.3, it will succeed.

$$\begin{aligned}\overline{g} &= v_2x^2 + v_1x + v_0 \\ &= \frac{1}{x}f(x) \circ [f(x) - 1^m] \circ [f(x) + 1^m] \\ &= \frac{1}{x}(\overline{f} \circ [\overline{f} - 1^m] \circ [\overline{f} + 1^m])\end{aligned}$$

For equation 7.7, because $\mathcal{P}$ computes it honestly according to equation 7.4, it will succeed.

$$\begin{aligned}\overline{h} &= w_2x^2 + w_1x + w_0 \\ &= \frac{1}{x}d(x) \circ [d(x) - 1^m] \circ [d(x) + 1^m] \\ &= \frac{1}{x}(\overline{d} \circ [\overline{d} - 1^m] \circ [\overline{d} + 1^m])\end{aligned}$$ □

We cite the following lemma from paper [6] to complete the soundness proof.

Lemma 7.3 *If there exist some $c^* \in \mathbb{F}^3$ such that $d(C, c^*E^*) \geq \frac{\delta}{10}$, then*

$$Pr\left[d(C, (x^2, x, 1)E^*) \leq \frac{\delta}{30}\right] \leq \frac{2}{q-1}$$

where q is the size of the underlying field, d is the relative distance function and C is the codeword.

Lemma 7.4 *LWE = ($\mathcal{P}$, $\mathcal{V}$) has has soundness error at most*

$$\max\left\{\frac{2}{q} + \frac{q-2}{q}(1-\delta)^\lambda, \frac{2}{q-1} + \frac{q-3}{q-1}(1 - \frac{29\delta}{30})^\lambda, (1 - \frac{7\delta}{10})^\lambda\right\}$$

where q is the size of the underlying field and δ is the relative distance of the codeword represented by the encoding function $\widetilde{\text{ENC}}$.

Proof Suppose H'_2, H'_1 or H'_0 is at least $\frac{\delta}{10}$ far away from a valid codeword. Without loss of generality, we assume H'_2 is not a valid codeword. Then let $c^* = (1, 0, 0)$, according to lemma 7.3, the probability that a structured linear combination of H'_2, H'_1 and H'_0 is $\frac{\delta}{30}$ close to a codeword is bound by $\frac{2}{q-1}$. Therefore, the soundness error is at most $\frac{2}{q-1} + \frac{q-3}{q-1}(1 - \frac{29\delta}{30})^{\lambda}$.

Otherwise, H'_2, H'_1 and H'_0 are $\frac{\delta}{10}$ close to a valid codeword, and it is possible to decode them to a instance/witness $(\mathbb{X}, \mathbb{W}) = ((\mathbb{F}, n, m, A, u), (s, e))$.

Suppose the malicious prover $\mathcal{P}$ does not follow the protocol honestly and sends incorrect messages.

- If $\overline{f}, \overline{g}$, or $\overline{h}$ is incorrect, then according to the relative distance property of the encoding function $\widetilde{\textsc{Enc}}$, at least δ portion of $\widetilde{\textsc{Enc}}(\overline{H}, \overline{r})$ and $x^2\widetilde{\textsc{Enc}}(H_2, r_2) + x\widetilde{\textsc{Enc}}(H_1, r_1) + \widetilde{\textsc{Enc}}(H_0, r_0)$. And since $\widetilde{\textsc{Enc}}(H_i, r_i)$ and H'_i are $\frac{\delta}{10}$ close to each other, at least $\frac{7\delta}{10}$ portion of $\widetilde{\textsc{Enc}}$ and $H'_2x^2 + H'_1x + H'_0$ will be different. The probability that all λ random checks (equation 7.5) are passed is at most $(1 - \frac{7\delta}{10})^{\lambda}$.
- If $f_2, f_1, f_0, v_2, v_1, v_0, w_2, w_1$ or w_0 is incorrect, denote the incorrect polynomial as f', g' or h'. Since f and f', g and g' or h and h' are polynomials with degree at most 2. According to Schwartz-Zippel lemma, they can agree on at most 2 evaluation points. And since evaluation point x is sampled randomly, the probability this event happens is at most $\frac{2}{q}$. If this event does not happen, then according to the relative distance property of the encoding function $\widetilde{\textsc{Enc}}$, at least δ portion of $\widetilde{\textsc{Enc}}(\overline{H}, \overline{r})$ and $H'_2x^2 + H'_1x + H'_0$ will be different. The probability that all λ random checks (equation 7.5) are passed is at most $(1 - \delta)^{\lambda}$. Therefore, the soundness error is at most $\frac{2}{q} + \frac{q-2}{q}(1 - \delta)^{\lambda}$.

Otherwise, the prover $\mathcal{P}$ follows the protocol honestly. Suppose $(\mathbb{X}, \mathbb{W}) = ((\mathbb{F}, n, m, A, u), (s, e))$ is not in relation R_{LWE}. Then at least one of the following conditions is satisfied:

- $s \notin \{-1, 0, 1\}^m$: Then there is an $v_{-1}X^{-1}$ term in $\frac{1}{X}f(X) \circ [f(X) - 1^m] \circ [f(X) + 1^m]$. Therefore, polynomial g is incorrect, denote the incorrect polynomial as g'. g and g' are polynomials with degree 2. According to Schwartz-Zippel lemma, they can agree on at most 2 evaluation points. And since evaluation point x is sampled randomly, the probability this event happens so that equation 7.6 is satisfied is at most $\frac{2}{q}$.
- $e \notin \{-1, 0, 1\}^n$: Then there is an $w_{-1}X^{-1}$ in $\frac{1}{X}d(X) \circ [d(X) - 1^n] \circ [d(X) + 1^n]$. Therefore, polynomial h is incorrect, denote the incorrect polynomial as h'. h and h' are polynomials with degree 2. According to Schwartz-Zippel lemma, they can agree on at most 2 evaluation points.

And since evaluation point x is sampled randomly, the probability this event happens so that equation 7.7 is satisfied is at most $\frac{2}{q}$.

- $u \neq As + e$: Then polynomial d will be incorrect, denote the incorrect polynomial as d'. Then, $\overline{h}$ and $\frac{1}{x}(d' \circ [d' - 1^n] \circ [d' + 1^n])$ can agree on at most 2 evaluation points. And since evaluation point x is sampled randomly, the probability this event happens so that equation 7.7 is satisfied is at most $\frac{2}{q}$.

Lemma 7.5 *LWE = ($\mathcal{P}$, $\mathcal{V}$) is semi-honest zero-knowledge.*

Proof The verifier $\mathcal{V}$'s view includes $(x, \overline{f}, \overline{g}, \overline{h}, \overline{r})$ and $(H_2'[i], H_1'[i], H_0'[i])$ for $\forall i \in I$. The simulator $\mathcal{S}(A, u)$ can generate the verifier $\mathcal{V}$'s view as follows:

- $\mathcal{S}$ samples $x \in \mathbb{F}$ uniformly at random.
- $\mathcal{S}$ samples $\overline{f} \in \mathbb{F}^m$ uniformly at random.
- $\mathcal{S}$ computes $\overline{d} = u - A\overline{f} \in \mathbb{F}^n$.
- $\mathcal{S}$ computes $\overline{g} = \frac{1}{x}(\overline{f} \circ [\overline{f} - 1^m] \circ [\overline{f} + 1^m]) \in \mathbb{F}^m$.
- $\mathcal{S}$ computes $\overline{h} = \frac{1}{x}(\overline{d} \circ [\overline{d} - 1^n] \circ [\overline{d} + 1^n]) \in \mathbb{F}^n$.
- $\mathcal{S}$ samples $r_2, r_1, r_0 \in \mathbb{F}^N$ uniformly at random.
- $\mathcal{S}$ computes $\overline{r} = x^2 r_2 + x r_1 + r_0 \in \mathbb{F}^N$.
- $\mathcal{S}$ samples $f_2, f_1 \in \mathbb{F}^m$ uniformly at random.
- $\mathcal{S}$ samples $v_2, v_1 \in \mathbb{F}^m$ uniformly at random.
- $\mathcal{S}$ samples $w_2, w_1 \in \mathbb{F}^n$ uniformly at random.
- $\mathcal{S}$ computes $f_0 = \overline{f} - x^2 f_2 - x f_1 \in \mathbb{F}^m$.
- $\mathcal{S}$ computes $v_0 = \overline{g} - x^2 v_2 - x v_1 \in \mathbb{F}^m$.
- $\mathcal{S}$ computes $w_0 = \overline{h} - x^2 w_2 - x w_1 \in \mathbb{F}^n$.
- $\mathcal{S}$ computes $H_2' = \widetilde{\textsc{Enc}}(H_2, r_2) \in \mathbb{F}^{2N}$, where $H_2 = f_2, v_2, w_2$.
- $\mathcal{S}$ computes $H_1' = \widetilde{\textsc{Enc}}(H_1, r_1) \in \mathbb{F}^{2N}$, where $H_1 = f_1, v_1, w_1$.
- $\mathcal{S}$ computes $H_0' = \widetilde{\textsc{Enc}}(H_0, r_0) \in \mathbb{F}^{2N}$, where $H_0 = f_0, v_0, w_0$.
- $\mathcal{S}$ outputs $(x, \overline{f}, \overline{g}, \overline{h}, \overline{r})$ and $(H_2'[i], H_1'[i], H_0'[i])$ for $\forall i \in I$. □

x is uniformly random both in the simulated transcripts and real transcripts.

For the simulated transcripts, $\overline{f}$ is randomly sampled. For the real transcripts, $\overline{f}$ also looks random because $\overline{f} = tx + s$ where t is randomly sampled.

For both the simulated transcripts and the real transcripts, $\overline{g}$ equals to $\frac{1}{x}(\overline{f} \circ [\overline{f} - 1^m] \circ [\overline{f} + 1^m])$. Since $\overline{f}$ looks random, they are indistinguishable to each other.

For both the simulated transcripts and the real transcripts, $\overline{h}$ equals to $\frac{1}{x}(\overline{d} \circ [\overline{d} - 1^n] \circ [\overline{d} + 1^n])$. Since $\overline{d} = u - A\overline{f}$ looks random, they are indistinguishable to each other.

For both the simulated transcripts and the real transcripts, r_2, r_1, r_0 are randomly sampled. Therefore, $\overline{r} = x^2 r_2 + x r_1 + r_0$ looks random.

In the real transcripts, H_2', H_1', and H_0' look random because the mask r_2, r_1, r_0 are randomly sampled and all other elements are hided by the random mask. In the simulated transcripts, for the same reason, H_2', H_1', and H_0' also look random. Therefore, as long as $\forall i_1, i_2 \in I, |i_1 - i_2| \neq N$, they are indistinguishable to each other.

7.3 Benchmark

We benckmark the above LWE protocol on a computer with Intel ® Core ™ i7-7700HQ CPU @ 2.80GHz (Kabylake), L1 cache: 128KB, L2 cache: 256KB and L3 cache: 6MB. There are 8 physical CPU cores available on this machine. The runtimes are summarized in the table 7.1.

As n and m increases, it is generally require more time to complete the committing phase for the prover and the checking phase for the verifier. Also, larger n and m will result in larger proof size.

Also figure 7.1 shows the relation between soundness error and the number of testing tuples (λ). As mentioned in lemma 7.4, the soundness error is the maximum value of three individual terms. Figure 7.1 labels these terms using lines with different colors. When λ is small, the soundness error is dominated by $(1 - \frac{7\delta}{10})^\lambda$. As λ increasing, the soundness is decreasing and gradually dominated by $\frac{2}{q-1}$. Hence, the minimum possible soundness is actual independent to the number of testing tuples (λ), and is determined solely by the field size. In our benchmark, we use a field with size roughly equals to 2^{256}, therefore, the minimum possible soundness error is around 2^{-256}.

n	m	Code Length	Prover Time [ms]	Verifier Time [ms]	Proof Size [bytes]
128	128	661	56	42	10624
	256	1101	116	66	12448
	512	1982	168	130	14496
	1024	3743	453	178	19392
	2048	7266	564	329	28384
256	128	881	80	60	10624
	256	1321	131	102	12448
	512	2202	231	165	15296
	1024	3963	545	344	19392
	2048	7486	824	653	28384
512	128	1321	155	103	11424
	256	1762	220	162	12448
	512	2642	385	295	15296
	1024	4404	904	654	20192
	2048	7926	1450	1229	28384
1024	128	2202	292	192	12224
	256	2642	533	276	13248
	512	3523	743	647	15296
	1024	5284	1403	1166	20192
	2048	8807	2679	2166	29184
2048	128	3963	526	399	12224
	256	4404	916	603	14048
	512	5284	1334	922	16096
	1024	7046	2178	2097	20192
	2048	10568	4078	3746	29184

Table 7.1: Runtime of LWE protocol with 1 thread and 200 test tuples. The soundness error is around 0.007.

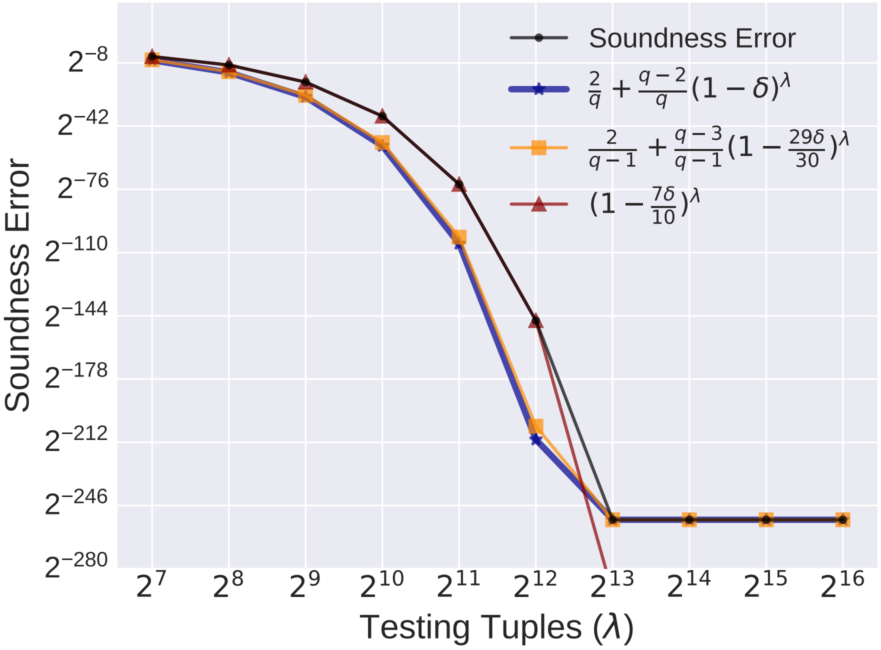


Figure 7.1: Soundness Error of LWE Protocol

Chapter 8

Implementation Details

8.1 Merkle Tree Commitment

A Merkle Tree is a data structure that allows one to commit to $l = 2^d$ messages by a single hash value h, such that revealing any bit of the message require $d + 1$ hash values. A Merkle hash tree is presented by a binary tree of depth d where l messages elements $m_1, m_2, \cdots, m_l$ are assigned to the leaves of the tree. The values assigned to internal nodes are computed by hashing the value of its two child nodes. To reveal m_i, we need to reveal m_i together with the values on the path from m_i to the root. We denote the algorithm as follows:

1. $h \leftarrow \text{Merkle.Commit}(m_1, m_2, \cdots, m_l)$

2. $(m_i, \phi_i) \leftarrow \text{Merkle.Open}(m, i)$

3. $\{\text{accept}, \text{reject}\} \leftarrow \text{Merkle.Verify}(\phi_i, m_i, h)$

In practice, we use Merkle tree commitment to compile the IOP or IOPP to an real argument system. Each element in the large array message π sent by the prover will be considered to be a leaf node of a Merkle tree. And the corresponding Merkle tree root will be sent to the verifier instead. For each query at position I, the prover will responds with $\pi[I]$ and the corresponding Merkle tree path, which will be authenticated later by the verifier.

Coefficient matrices $M_0', M_1', \cdots, M_{t-1}'$ sent by the prover may be replaced by a Merkle tree commitment to that matrix. And since each time the verifier will query a strip of elements in a matrix (i.e. $M_i'[i_1, i_2, \cdots, i_{t-i-1}, *]$), it is possible to zip such a strip of elements into a single node in Merkle-tree's leaf level to decrease runtime complexity and communication complexity.

8.2 Zero-knowledge Merkle Tree Commitment

To implement a zero-knowledge polynomial commitment scheme, we also need a zero-knowledge Merkle tree commitment to prevent information leaking from the Merkle tree path. If we use the random oracle model, we can argue that the Merkle hash is completely random, thus, leaking no information at all. On the other hand, we can prevent information leaking by adding randomness into the leaf nodes. The leaf node is $hash(data_i||r_i)$ where r_i is some random elements.

8.3 Parallelism

Most of the computations for the polynomial commitment scheme can be done in parallel in a natural fashion. There is little data dependence among them. Therefore, it is possible to run the commitment scheme using multiple threads to increase efficiency significantly for both the prover and the verifier.

Appendix A

Mathematical Preliminaries

Lemma A.1 *$\binom{n}{m} \leq (\frac{en}{m})^m$, for $n, m \in \mathbb{Z}^+$*

Proof

$$\begin{aligned}
\log m! &= \sum_{i=1}^{m} \log i \\
&\geq \int_1^m \log x \, dx \\
&= [x \log x - x]_1^m \\
&= m \log m - m + 1 \qquad \text{(A.1)}
\end{aligned}$$

$$\begin{aligned}
m! &= e^{\ln m!} \\
&\geq e^{m \log m - m + 1} && \text{apply equation A.1} \\
&= e^{\log m^m} \cdot e^{-m} \cdot e \\
&= m^m \cdot e^{-m} \cdot e \\
&= (\frac{m}{e})^m \cdot e \\
&\geq (\frac{m}{e})^m \qquad \text{(A.2)}
\end{aligned}$$

$$\begin{aligned}\binom{n}{m} &= \frac{n\cdot(n-1)\cdot(n-2)\cdots(n-m+1)}{m!}\\ &\le \frac{n^m}{m!}\\ &\le \frac{n^m}{(\frac{m}{e})^m} && \text{apply equation A.2}\\ &= (\frac{en}{m})^m && \text{(A.3)}\end{aligned}$$

□

Lemma A.2 $(1-\frac{1}{x})^x \le \frac{1}{e}$*, for* $x \ge 1$

Proof

Recall that for $x \in \mathbb{R}$

$$1 + x \le e^x$$

Then for $x \in \mathbb{R}$

$$1 - x \le e^{-x}$$

Then for $x \neq 0$

$$1 - \frac{1}{x} \le e^{-\frac{1}{x}}$$

And, since $t \mapsto t^x$ is increasing on $[0, \infty]$ for $x \ge 1$

$$(1-\frac{1}{x})^x \le \frac{1}{e} \tag{A.4}$$

□

Lemma A.3 $(\frac{a}{x})^x \le e^{\frac{a}{e}}$*, for* $x > 0$*,* $a > 0$

Proof

Let $f(x) = (\frac{a}{x})^x$

$$\ln f(x) = x\cdot\ln(\frac{a}{x}) = -x\cdot\ln\frac{x}{a}$$

Take derivative from both side

$$\begin{aligned}\frac{1}{f(x)}\frac{df(x)}{dx} &= -\ln\frac{x}{a} - x\cdot\frac{a}{x}\cdot\frac{1}{a} = -\ln\frac{x}{a} - 1\\ \frac{df(x)}{dx} &= -f(x)\cdot(\ln\frac{x}{a}+1) = -(\frac{a}{x})^x\cdot(\ln\frac{x}{a}+1)\end{aligned}$$

Let $\frac{df(x)}{dx} = 0$

$$\begin{aligned}-(\frac{a}{x})^x\cdot(\ln\frac{x}{a}+1) &= 0\\ (\ln\frac{x}{a}+1) &= 0\\ x &= \frac{a}{e}\end{aligned}$$

$\frac{df(x)}{dx} > 0$ when $x < \frac{a}{e}$, and $\frac{df(x)}{dx} < 0$ when $x > \frac{a}{e}$
Therefore, $x = \frac{a}{e}$ is a maximum point

$$(\frac{a}{x})^x = f(x) \leq f(\frac{a}{e}) = e^{\frac{a}{e}} \tag{A.5}$$

□

Bibliography

[1] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 21–31. ACM, 1991.

[2] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 16–25. IEEE Computer Society, 1990.

[3] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 31–60, 2016.

[4] Jonathan Bootle, Alessandro Chiesa, and Jens Groth. Linear-time arguments with sublinear verification from tensor codes. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part II*, volume 12551 of *Lecture Notes in Computer Science*, pages 19–46. Springer, 2020.

[5] Jonathan Bootle, Alessandro Chiesa, and Siqi Liu. Zero-knowledge iops with linear-time prover and polylogarithmic-time verifier. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 275–304. Springer, 2022.

[6] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. More efficient amortization of exact zero-knowledge proofs for LWE. In Elisa Bertino, Haya Shulman, and Michael Waidner, editors, *Computer Security - ESORICS 2021 - 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4-8, 2021, Proceedings, Part II*, volume 12973 of *Lecture Notes in Computer Science*, pages 608–627. Springer, 2021.

[7] Alin Bostan, Grégoire Lecerf, and Éric Schost. Tellegen's principle into practice. In J. Rafael Sendra, editor, *Symbolic and Algebraic Computation, International Symposium ISSAC 2003, Drexel University, Philadelphia, Pennsylvania, USA, August 3-6, 2003, Proceedings*, pages 37–44. ACM, 2003.

[8] Erez Druk and Yuval Ishai. Linear-time encodable codes meeting the gilbert-varshamov bound and their cryptographic applications. In *Proceedings of the 5th Conference on Innovations in Theoretical Computer Science*, ITCS '14, page 169–182, New York, NY, USA, 2014. Association for Computing Machinery.

[9] Lance Fortnow, John Rompel, and Michael Sipser. On the power of multi-prover interactive protocols. *Theor. Comput. Sci.*, 134(2):545–557, 1994.

[10] S. I. Gelfand, R. L. Dobrushin, and M. S. Pinsker. On the complexity of coding. In *Second International Symposium on Information Theory*, pages 177–184, 1973.

[11] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In Oded Goldreich, editor, *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 203–225. ACM, 2019.

[12] Alexander Golovnev, Jonathan Lee, Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. Brakedown: Linear-time and post-quantum snarks for R1CS. *IACR Cryptol. ePrint Arch.*, page 1043, 2021.

[13] Venkatesan Guruswami and Piotr Indyk. Linear-time encodable/decodable codes with near-optimal rate. *IEEE Trans. Inf. Theory*, 51(10):3393–3400, 2005.

[14] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: The power of no-signaling proofs. *J. ACM*, 69(1):1:1–1:82, 2022.

[15] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 177–194. Springer, 2010.

[16] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 723–732. ACM, 1992.

[17] Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. In Frank Thomson Leighton and Allan Borodin, editors, *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA*, pages 388–397. ACM, 1995.

[18] Alin Tomescu, Robert Chen, Yiming Zheng, Ittai Abraham, Benny Pinkas, Guy Golan-Gueta, and Srinivas Devadas. Towards scalable threshold cryptosystems. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 877–893. IEEE, 2020.

[19] Tiancheng Xie, Yupeng Zhang, and Dawn Song. Orion: Zero knowledge proof with linear prover time. Cryptology ePrint Archive, Paper 2022/1010, 2022. https://eprint.iacr.org/2022/1010.

[20] Thomas Yurek, Licheng Luo, Jaiden Fairoze, Aniket Kate, and Andrew K. Miller. hbacss: How to robustly share many secrets. *IACR Cryptol. ePrint Arch.*, page 159, 2021.

[21] Jiaheng Zhang, Tiancheng Xie, Thang Hoang, Elaine Shi, and Yupeng Zhang. Polynomial commitment with a One-to-Many prover and applications. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 2965–2982, Boston, MA, August 2022. USENIX Association.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):

First name(s):

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date

Signature(s)

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.