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

Contents

Chapter 1

Preliminaries

Definition 1.1 (*d*-regular Graph) *A graph $G = (V, E)$ is d-regular if every vertex in V has degree d.*

Definition 1.2 (Set) *$[n]$ is the shorthand for the set $\{i : 1 \leq i \leq n\}$*

Definition 1.3 (Relation) *An **relation** R is a set of pair $(\mathbb{X}, \mathbb{W})$ 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 1.4 (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. 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.

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 $\tilde{P}$ convince $V(\mathbb{X})$ to accept is at most ϵ.*

Definition 1.5 (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. 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 Δ 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 $R|_{\mathbb{X}} := \{\mathbb{W}' | (\mathbb{X}, \mathbb{W}') \in R\}$.*

Definition 1.6 (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 1.7 (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 1.8 (Multivariate / Univariate Polynomial) *A **multivariate** polynomial is a polynomial with more than one variable; otherwise it is called a **univariate** polynomial.*

Definition 1.9 (Multilinear Polynomial) *A **multivariate** polynomial is called a **multilinear** polynomial if the degree of the polynomial in each variable is at most one.*

Definition 1.10 (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.*

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**.

Chapter 2

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 [2]. 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.

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

Lemma 2.1 (Polynomial Evaluation [2]) *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:

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

2.2 Polynomial Commitment for t = 3

Commitment Phase.

Let $M_0 = u$ and $M_0' = \mathrm{Enc}_1(\mathrm{Enc}_2(M_0)) \in \mathbb{F}^{N \times N \times m}$. Send M_0' to the verifier.

Testing Phase.

The testing phase consists of 3 rounds, with each round reducing the dimension by 1.

In round 1, the verifier will send a random value $r_1 \in \mathbb{F}^m$ to the prover. The prover will compute a linear combination $M_1 \in \mathbb{F}^{m \times m}$ of the 3rd dimension of matrix M_0. Namely, $M_1[i,j] = \sum_{k=1}^{m} r_1[k] \cdot M_0[i,j,k]$ for $1 \leq i, j \leq m$. Let $M_1' = \mathrm{Enc}_1(M_1) \in \mathbb{F}^{N \times m}$. Then the prover sends M_1' to the verifier.

In round 2, the verifier will send a random value $r_2 \in \mathbb{F}^m$ to the prover. The prover will compute a linear combination $M_2 \in \mathbb{F}^m$ of the 2nd dimension of matrix M_1. Namely, $M_2[i] = \sum_{k=1}^{m} r_2[k] \cdot M_1[i,k]$. Let $M_2' = M_2 \in \mathbb{F}^m$. Then the prover sends M_2' to the verifier.

In round 3, the verifier will send a random value $r_3 \in \mathbb{F}^m$ to the prover. The prover will compute a linear combination $M_3 \in \mathbb{F}$ of the first dimension of matrix M_2. Namely, $M_3 = \sum_{k=1}^{m} r_3[k] \cdot M_1[k]$. Then the prover sends M_3 to the verifier.

Then the verifier will perform a probabilistic check to make sure M_0', M_1', M_2' and M_3 are consistent with each other. Formally speaking, the verifier will sample l random tuple (j_1, j_2, j_3) from space $[N] \times [N] \times [N]$. For each tuple (j_1, j_2, j_3), let $M_1'[j_1, *]$ denotes the j_1-th row in M_1'. The verifier will check whether the following equation holds:

$$\mathrm{Enc}(M_1'[j_1, *])[i_2] \stackrel{?}{=} \sum_{k=1}^{m} r_1[k] \cdot M_0'[j_1, j_2, k]$$

$$\mathrm{Enc}(M_2'[*])[j_1] \stackrel{?}{=} \sum_{k=1}^{m} r_2[k] \cdot M_1'[j_1, k]$$

$$M_3[*] \stackrel{?}{=} \sum_{k=1}^{m} r_3[k] \cdot M_2'[k]$$

Evaluation Phase.

Let $q_1, q_2, q_3 \in \mathbb{F}^m$ be vectors such that $g(x) = \langle q_1 \otimes q_2 \otimes q_3, u \rangle$. The evaluation phase is identical to the testing phase, except that in round i, the random value r_i is replaced by q_i. If all consistency checks passed, then the verifier outputs M_3 as $g(x)$.

2.3 Polynomial Commitment for Arbitrary t

2.3.1 Protocol

Commitment Phase.

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

Testing Phase.

The testing phase consists of t rounds, with each round reducing the number of dimensions by 1.

In round i, the verifier will send a random value $r_i \in \mathbb{F}^m$ to the prover. The prover will compute a linear combination $M_i \in \mathbb{F}^{\overbrace{m \times m \times \cdots \times m}^{t-i \text{ times}}}$ of the last dimension of matrix M_{i-1}. Namely, for $1 \leq j_1, j_2, \cdots, j_{t-i} \leq m$:

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

Let

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

Then the prover sends M_i' to the verifier.

Note that in the last round, M_t is degenerate to a single value in $\mathbb{F}$ and no encoding takes place.

Then the verifier will perform a probabilistic check to make sure M_0', M_1', $M_2', \cdots, M_{t-1}'$ are consistent with each other. Formally speaking, the verifier 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 $(j_1, j_2, \cdots, j_t)$, the verifier will check whether the following equation holds for every $i \in [t]$:

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

Evaluation Phase.

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 evaluation phase is identical to the testing phase, except that in round i, the random value r_i is replaced by q_i. If all consistency checks passed, then the verifier outputs M'_t as $g(x)$.

2.4 Analysis

We refer to the result in [1] and summarize to the following lemmas.

Lemma 2.2 *The testing phase (proximity test) has perfect completeness.*

Lemma 2.3 *The testing phase (proximity test) has soundness error:*

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

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

2.5 Implementation Details

2.5.1 Merkle Tree Commitment

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 each query the verifier makes to a matrix is answered by the prover with Merkle tree authentication path for the answer.

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's 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.

2.5.2 Parallelism

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

Chapter 3

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.

3.1 Protocol

Commitment Phase.

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

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

where $\oplus$ denotes elements-wise tensor addition. Send M_0' to the verifier.

Testing Phase.

The testing phase consists of t rounds, with each round reducing the number of dimensions by 1.

In round i, the verifier will send a random value $r_i \in \mathbb{F}^m$ to the prover. The prover will compute 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 $1 \leq j_1, j_2, \cdots, j_{t-i} \leq m$:

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

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

Let

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

where $\oplus$ denotes element-wise addition.

Then the prover sends M'_i to the verifier.

Note that in the last round, M_t is degenerate to a single value in $\mathbb{F}$ and no encoding takes place. Additionally, the prover will send PAD_t to the verifier in the last round.

Then the verifier will perform a probabilistic check to make sure M'_0, M'_1, $M'_2, \cdots, M'_t$ are consistent with each other. Formally speaking, the verifier will sample l random tuple $(j_1, j_2, \cdots, j_t)$ from space $\underbrace{[N] \times [N] \times \cdots \times [N]}_{t \text{ times}}$. For each sampled tuple $(j_1, j_2, \cdots, j_t)$, the verifier will check the following equation holds for every $i \in [t]$.

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

Evaluation Phase.

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 evaluation phase is identical to the testing phase, except that in round i, the random value r_i is replaced by q_i. If all consistent checks passed, then the verifier outputs $M'_t - PAD_t$ as $g(x)$.

3.2 Formal Description

3.2.1 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 $\mathrm{Enc}_1 \circ \mathrm{Enc}_2 \circ \cdots \circ \mathrm{Enc}_i$.

3.2.2 Testing Phase (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 **P** and a verifier **V**.

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

1. *Interactive phase*.

   In the beginning, **P** sends the proof message M'_0 computed as:

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

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

   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 $i \in [t]$:

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

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

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

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

   Note that in the last round, M_t is degenerate to a single value in $\mathbb{F}$ and no encoding takes place.

2. *Query phase*. The verifier **V** samples l tuples of the form $(j_1, \cdots, j_t)$ in space $[N]^t$. The verifier **V** proceeds as follows for each sampled tuple.

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

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

   $$\mathrm{Enc}_{t-i}(M'_i)[i_1, \cdots, i_{t-i}] \stackrel{?}{=} \mathbf{Fold}_{t-i+1}(M'_{i-1}, r_i)[i_1, \cdots, i_{t-i}] \tag{3.1}$$

3.2.3 Testing Phase Completeness

Lemma 3.1 *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 3.1).

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

$$\mathrm{Enc}_{t-i}(M'_i) \stackrel{?}{=} \mathbf{Fold}_{t-i+1}(M'_{i-1}, r_i)$$

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

$$\begin{aligned}\mathrm{Enc}_{t-i}(M'_i) &= \mathrm{Enc}_{t-i}(\mathbf{Enc}_{1,\cdots,t-i-1}(M_i \oplus PAD_i)) \\ &= \mathbf{Enc}_{1,\cdots,t-i}(M_i \oplus PAD_i) \\ &= \mathbf{Enc}_{1,\cdots,t-i}(\mathbf{Fold}_{t-i+1}(M_{i-1} \oplus PAD_{i-1}, r_i)) \qquad (3.2) \\ & \qquad (3.3)\end{aligned}$$

And the right side of this equation is equivalent to:

$$\mathbf{Fold}_{t-i+1}(M'_{i-1}, r_i) = \mathbf{Fold}_{t-i+1}(\mathbf{Enc}_{1,\cdots,t-i}(M_{i-1} \oplus PAD_{i-1}), r_i) \qquad (3.4)$$
$$(3.5)$$

□

Since both **Fold** and **Enc** operations are linear operation, expression 3.2 and expression 3.4 are equivalent to each other. The equations checked by the verifier **V** holds.

3.2.4 Testing Phase Soundness

Lemma 3.2 *IOPP = (**P**, **V**) has soundness error at most:*

$$\epsilon'(\Delta_\otimes) = \epsilon(\Delta_\otimes)$$

Proof We prove this lemma by arguing that the soundness (lemma 2.3) of original proximity test implies the soundness of the simple zero-knowledge proximity test. And we will prove this argument by doing a reduction formally described in figure 3.1.

For the simple zero-knowledge proximity test, suppose we have an malicious prover algorithm $\tilde{\mathbf{P}}_{ZK}$ that can pass the proximity test with probability p, we will run it as a black box and construct another malicious prover algorithm $\tilde{\mathbf{P}}$ that can pass the original proximity test with the same probability p.

- $\tilde{\mathbf{P}}$ will receive an input M representing the tensor. $\tilde{\mathbf{P}}$ pass M to $\tilde{\mathbf{P}}_{ZK}$.
- $\tilde{\mathbf{P}}_{ZK}$ sends out M'_0 as the commitment to the underlying tensor. $\tilde{\mathbf{P}}$ sends M'_0 to the verifier **V** as the commitment to the underlying tensor.
- For $i \in [t]$, verifier **V** will send the random linear combination coefficients r_i to $\tilde{\mathbf{P}}$. $\tilde{\mathbf{P}}$ simply forward r_i to $\tilde{\mathbf{P}}_{ZK}$. $\tilde{\mathbf{P}}_{ZK}$ will output M'_i. $\tilde{\mathbf{P}}$ then send M'_i to the verifier **V**.

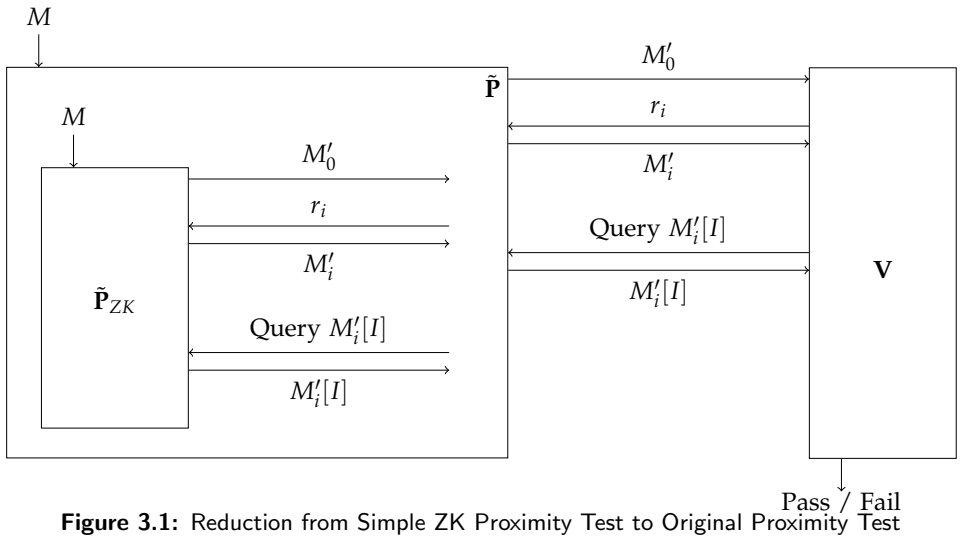

Figure 3.1: Reduction from Simple ZK Proximity Test to Original Proximity Test

- If verifier **V** query M'_i at index I for $0 \leq i \leq t$, $\tilde{\mathbf{P}}$ forward the query to $\tilde{\mathbf{P}}_{ZK}$ and forward the reply back to the verifier. □

During the query phase, the original verifier **V** and the zero-knowledge verifier will check the same equation:

$$\text{Enc}_{t-i}(M'_i) = \mathbf{Fold}_{t-i+1}(M'_{i-1}, r_i)$$

As long as malicious prover $\tilde{\mathbf{P}}_{ZK}$ can pass the check in zero-knowledge proximity test, our constructed prover $\tilde{\mathbf{P}}$ can pass the check in original proximity test.

Therefore, the soundness error of the simple zero-knowledge protocol is at most

$$\epsilon'(\Delta_\otimes) = \epsilon(\Delta_\otimes)$$

3.2.5 Testing Phase Zero-Knowledge

Definition 3.3 *A interactive oracle proof of proximity IOPP = (**P**, **V**) for a relation R is **perfect 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 $\boldsymbol{S}^{V(\mathbb{X};\rho)}(\mathbb{X})$ and View$(\boldsymbol{P}(\mathbb{X}, \mathbb{W}), \boldsymbol{V}(\mathbb{X}; \rho))$ are identically distributed.*

Lemma 3.4 *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:

- Query M''_0 or internally access PAD_0 at index I:

If index I has never been accessed before, generate a random element x from field $\mathbb{F}$. Store this value x in a internal dictionary and return it.

Otherwise lookup index I in the internal dictionary and return the stored value.

- Query M_i'' and PAD_i at index I for $i \in [t-1]$:

  If index I has never been accessed before, recursively access the corresponding values in M_{i-1}'' and PAD_{i-1} and take the linear combination of them as x. Store this value x in a internal dictionary and return it.

  Otherwise lookup index I in the internal dictionary and return the stored value.

Both the random variables $\mathbf{S}^{\mathbf{V}(\mathbb{X};\rho)}(\mathbb{X})$ and $\text{View}(\mathbf{P}(\mathbb{X},\mathbb{W}),\mathbf{V}(\mathbb{X};\rho))$ are uniformly distributed. They are identically distributed.

Chapter 4

Brakedown Linear Code

4.1 Notation

α and β are parameters with no explicit meanings. r denotes the ratio between the length of codeword and the length of input message. δ denotes the relevant distance.

4.2 Theoretical Limits for Relevant Distance

In Brakedown paper [2], there're 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's also one more implicit constrain. The full list of constrains are as follows,

$$
\begin{aligned}
&0 < \alpha < 1 \\
&0 < \beta < \frac{\alpha}{1.28} && (4.1)\\
&r > \frac{1+2\beta}{1-\alpha} > 1 && (4.2)\\
&\delta = \frac{\beta}{r} && (4.3)\\
&\beta + \alpha\beta + 0.03 < r - 1 - r\alpha && (4.4)\\
& && (4.5)
\end{aligned}
$$

Combine constrain 4.3 and constrain 4.1, we have,

$$\alpha > 1.28 \cdot \delta \cdot r \tag{4.6}$$

Combine constrain 4.3 and constrain 4.2, we have,

$$\alpha > 1 - 2\delta - \frac{1}{r} \tag{4.7}$$

Combine constrain 4.3 and constrain 4.4, we have,

$$\alpha < \frac{r(1-\delta) - 1.03}{r(1+\delta)} \tag{4.8}$$

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

$$\frac{r(1-\delta) - 1.03}{r(1+\delta)} > 1.28 \cdot \delta \cdot r \tag{4.9}$$

$$\frac{r(1-\delta) - 1.03}{r(1+\delta)} > 1 - 2\delta - \frac{1}{r} \tag{4.10}$$

$$\tag{4.11}$$

Equation 4.9 and equation 4.10 make the maximum possible relevant distance δ to be around 0.12.

Chapter 5

Zero-Knowledge Linear Code

5.1 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's 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)*

5.2 Expander Graph

Lemma 5.1 *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 doesn't 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's 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,

$$
\begin{aligned}
P_4 &= \sum_{s=\epsilon n}^{n} \binom{n}{s} P_3 \\
&= \sum_{s=\epsilon n}^{n} \binom{n}{s} \binom{n}{\epsilon n} (\frac{n-1}{n})^{ds\epsilon n} \\
&\leq \sum_{s=\epsilon n}^{n} \binom{n}{s} \binom{n}{\epsilon n} (\frac{n-1}{n})^{d\epsilon^2 n^2} && \text{since } s \geq \epsilon n \text{ and } \frac{n-1}{n} < 1 \\
&\leq \sum_{s=\epsilon n}^{n} \binom{n}{s} 2^{nH(\frac{\epsilon n}{n})} (\frac{n-1}{n})^{d\epsilon^2 n^2} && \binom{n}{k} \leq 2^{nH(\frac{k}{n})} \\
&= \sum_{s=\epsilon n}^{n} \binom{n}{s} 2^{nH(\epsilon)} ((1-\frac{1}{n})^n)^{d\epsilon^2 n} \\
&\leq \sum_{s=\epsilon n}^{n} \binom{n}{s} 2^{nH(\epsilon)} (\frac{1}{e})^{d\epsilon^2 n} && (1-\frac{1}{x})^x \leq \frac{1}{e} \text{ for } x \geq 1 \text{ (lemma A.2)} \\
&= \sum_{s=\epsilon n}^{n} \binom{n}{s} (e^{H(\epsilon)\ln 2 - d\epsilon^2})^n \\
&\leq \sum_{s=0}^{n} \binom{n}{s} (e^{H(\epsilon)\ln 2 - d\epsilon^2})^n \\
&= 2^n (e^{H(\epsilon)\ln 2 - d\epsilon^2})^n && \sum_{i=0}^{n} \binom{n}{i} = 2^n \\
&= (e^{\ln 2 + H(\epsilon)\ln 2 - d\epsilon^2})^n
\end{aligned}
\tag{5.1}
$$

□

P_4 is the probability that a randomly generated graph G doesn't 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 5.2 *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 5.1. Negating the statement, we can say that the randomly generated graph G doesn't 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's 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:

$$\begin{aligned}
P_3 &= \binom{n}{\epsilon n}(P_2)^{\epsilon n} \\
&= \binom{n}{\epsilon n}(\frac{n-1}{n})^{d\epsilon^2 n^2} \\
&\leq 2^{nH(\frac{\epsilon n}{n})}(\frac{n-1}{n})^{d\epsilon^2 n^2} && \binom{n}{k} \leq 2^{nH(\frac{k}{n})} \\
&= 2^{nH(\epsilon)}((1-\frac{1}{n})^n)^{d\epsilon^2 n} \\
&\leq 2^{nH(\epsilon)}(\frac{1}{e})^{d\epsilon^2 n} && (1-\frac{1}{x})^x \leq \frac{1}{e} \text{ for } x \geq 1 \text{ (lemma A.2)} \\
&= (e^{H(\epsilon)\ln 2 - d\epsilon^2})^n
\end{aligned} \tag{5.2}$$

□

P_3 is the probability that a set S in a randomly generated graph doesn't 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 5.1, lemma 5.2 produces a much tighter bound by weakening the expansion property. A graph satisfy the expansion property

in lemma 5.2 may not satisfy the expansion property in lemma 5.1. There may exist a set $S \subset L$ in graph such that the expansion property fails. But lemma 5.2 guarantees that such set is hard to be found. Similar with hash functions, hash collision must exist somewhere, but this collision is hard to be found.

5.3 Reversed Linear Code

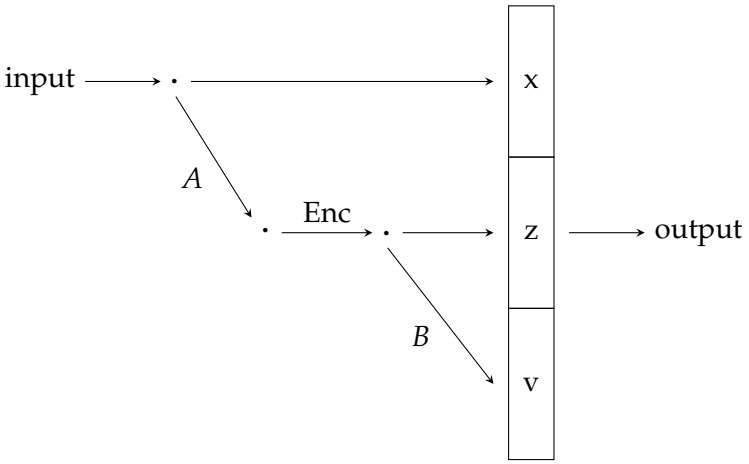

Figure 5.1: Linear Code

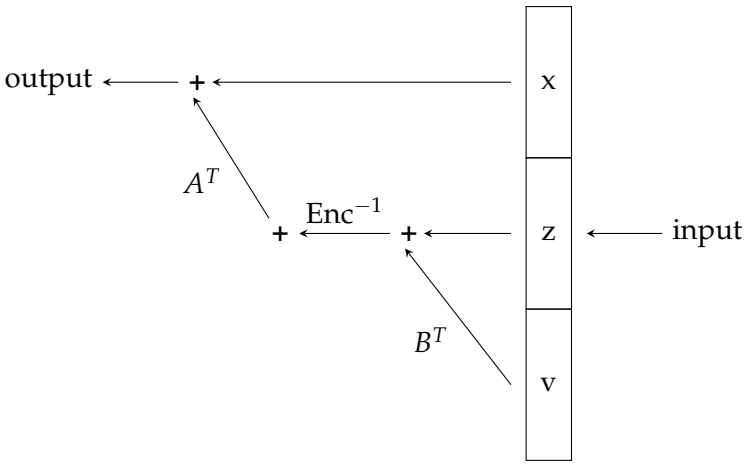

Figure 5.2: Reversed Linear Code

Appendix A

Mathematical Preliminaries

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

Proof

$$\begin{aligned} \log m! &= \log 1 + \log 2 + \cdots + \log m \\ &\geq \int_1^m \log x \, dx \\ &= [x \log x - x]_1^m \\ &= m \log m - m + 1 \end{aligned} \tag{A.1}$$

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

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

□

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

Proof

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

$$1 + x \leq e^x$$

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

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

Then for $x \neq 0$

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

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

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

□

Lemma A.3 $(\frac{a}{x})^x \leq e^{\frac{a}{e}}$, *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

$$\frac{1}{f(x)} \frac{df(x)}{dx} = -\ln \frac{x}{a} - x \cdot \frac{a}{x} \cdot \frac{1}{a} = -\ln \frac{x}{a} - 1$$

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

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

$$-(\frac{a}{x})^x \cdot (\ln \frac{x}{a} + 1) = 0$$

$$(\ln \frac{x}{a} + 1) = 0$$

$$x = \frac{a}{e}$$

$\frac{df(x)}{dx} > 0$ 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] Jonathan Bootle, Alessandro Chiesa, and Jens Groth. Linear-time arguments with sublinear verification from tensor codes. Cryptology ePrint Archive, Paper 2020/1426, 2020. https://eprint.iacr.org/2020/1426.

[2] Alexander Golovnev, Jonathan Lee, Srinath Setty, Justin Thaler, and Riad S. Wahby. Brakedown: Linear-time and post-quantum snarks for r1cs. Cryptology ePrint Archive, Report 2021/1043, 2021. https://ia.cr/2021/1043.

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.