Ligero++

Table of Contents

Definitions

Inner product argument

Prover wants to prove the verifier that $< a, b >= y$:

2.4 Inner Product Arguments

Inner product arguments (IPA) allow a verifier to validate the inner product of a committed vector from the prover and a public vector. Our protocols use the inner product arguments recently proposed by Zhang et al. in [55] as a building block. The scheme is a Reed-Solomon encoded interactive oracle proof based on the work of Aurora[20], and does not require a trusted setup. Let $y = \langle a, b \rangle$ be the inner product of two vectors. The scheme consists of the following algorithms:

- $\mathsf{pp} \leftarrow \mathsf{IPA.KeyGen}(1^\lambda)$,
- $\mathsf{com}_a \leftarrow \mathsf{IPA.Commit}(a, \mathsf{pp})$,
- $(y, \pi) \leftarrow \mathsf{IPA.Prove}(a, b, \mathsf{pp})$,
- $\{0, 1\} \leftarrow \mathsf{IPA.Verify}(\mathsf{pp}, y, \mathsf{com}_a, b, \pi)$

Inner product argument

THEOREM 2.6 ([55]). *There exists an inner product argument scheme satisfying the following properties:*

Inner product argument

- ***Completeness.*** *For any private vector* $a \in \mathbb{F}^n$, *public vector* $b \in \mathbb{F}^n$, $\mathsf{pp} \leftarrow \mathsf{IPA.KeyGen}(1^\lambda)$, $\mathsf{com} \leftarrow \mathsf{IPA.Commit}(a, \mathsf{pp})$, $\{y, \pi\} \leftarrow \mathsf{IPA.Prove}(a, b, \mathsf{pp})$, *it holds that*

$$\Pr\left[\mathsf{IPA.Verify}(\mathsf{pp}, y, \mathsf{com}_a, b, \pi) = 1\right] = 1$$

Inner product argument

- ***Soundness.*** *For any PPT adversary* $\mathcal{A}$, $\mathsf{pp} \leftarrow \mathsf{IPA.KeyGen}(1^\lambda)$, *the following probability is negligible in* λ*:*

$$\Pr\left[\begin{matrix}(a^*, \mathsf{com}^*, b, y^*, \pi^*) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp}) \\ \mathsf{IPA.Verify}(\mathsf{pp}, y^*, \mathsf{com}^*, b, \pi^*) = 1\end{matrix} : \begin{matrix}\mathsf{com}^* = \mathsf{IPA.Commit}(a^*, \mathsf{pp}) \\ \wedge \langle a^*, b\rangle \neq y^*\end{matrix}\right]$$

Inner product argument

Complexity. *Let the size of the vectors be n. The running time of* Commit *and* Prove *is $O(n \log n)$ time for the prover, and the running time of* Verify *is $O(n)$ for the verifier. The proof size is $O(\log^2 n)$.*

Reed-Solomon Code Summary

We have a message $m = (m_1, \ldots, m_k) \rightarrow$ We find polynomial p such that $(p(\zeta_1), \ldots, p(\zeta_k)) = (m_1, \ldots, m_k) \rightarrow$ We define $(p(\eta_1), \ldots, p(\eta_n)) = (u_1, \ldots, u_n) = u$ as the encoded message of m

Reed-Solomon Code

DEFINITION 4.1 (REED-SOLOMON CODE). *For positive integers n, k, finite field $\mathbb{F}$, and a vector $\eta = (\eta_1, \ldots, \eta_n) \in \mathbb{F}^n$ of distinct field elements, the code* $\mathrm{RS}_{\mathbb{F},n,k,\eta}$ *is the* $[n, k, n-k+1]$ *linear code over* $\mathbb{F}$ *that consists of all n-tuples* $(p(\eta_1), \ldots, p(\eta_n))$ *where p is a polynomial of degree* $< k$ *over* $\mathbb{F}$.

Reed-Solomon Code

Let $L = RS_{\mathbb{F},n,k,\eta}$ be an RS code and $\zeta = (\zeta_1, \ldots, \zeta_k)$ be a sequence of distinct elements in $\mathbb{F}$.

Reed-Solomon Code

For a codeword $u \in L$, we define the message $\mathrm{Dec}_{\zeta}(u)$ to be $(p_u(\zeta_1), \ldots, p_u(\zeta_k))$, where p_u is the polynomial (of degree $< k$) corresponding to u such that:

$$(p_u(\eta_1), \ldots, p_u(\eta_n)) = (u_1, \ldots, u_n)$$

Reed-Solomon Code Extended Version

For a codeword $u \in L$, we define the message $\mathrm{Dec}_\zeta(u)$ to be $(p_u(\zeta_1), \ldots, p_u(\zeta_k))$, where p_u is the polynomial (of degree $< k$) corresponding to u. For $U \in L^m$ with rows $u_1, \ldots, u_m \in L$, we let $\mathrm{Dec}_\zeta(U)$ be the length-mk vector $x = (x_{11}, \ldots, x_{1k}, \ldots, x_{m1}, \ldots, x_{mk})$ such that $(x_{i1}, \ldots, x_{ik}) = \mathrm{Dec}\left(u^i\right), i \in [m]$. Finally we say that U encodes x if $x = \mathrm{Dec}_\zeta(U)$, we use $\mathrm{Dec}(U)$ when ζ is clear from the context.

Reed-Solomon Code Complexity

In our protocol, we set $\eta_i = \omega^i$ where ω is a generator of a multiplicative group in field $\mathbb{F}$. We can evaluate $(p(\eta_0), p(\eta_1), ..., p(\eta_{n-1}))$ using the fast Fourier transform (FFT), which takes $O(n \log n)$ field operations. We use $\mathrm{RS}(a)$ to denote the RS encoding of message a.

Ligero

We have a matrix that has C elements. The dimension of this matrix is $\sqrt{C} \times \sqrt{C}$. The encoding of each row takes $\sqrt{C} \log C$. The overall encode time is $\mathcal{O}(C \log C)$. The communication time takes $t \times ColumnSize + RowSize = \mathcal{O}((t+1) \times \sqrt{C})$ time

Ligero

- **Oracle:** A purported L^m-codeword U. Depending on the context, we may view U either as a matrix in $\mathbb{F}^{m \times n}$ in which each row U_i is a purported L-codeword, or as a sequence of n symbols $(U[1], \ldots, U[n])$, $U[j] \in \mathbb{F}^m$.
- **Interactive testing:**
  (1) $\mathcal{V}$ picks a random linear combinations $r \in \mathbb{F}^m$ and sends r to $\mathcal{P}$.
  (2) $\mathcal{P}$ responds with $w = r^T U \in \mathbb{F}^n$.
  (3) $\mathcal{V}$ queries a set $Q \subset [n]$ of t random symbols $U[j]$, $j \in Q$.
  (4) $\mathcal{V}$ accepts iff $w \in L$ and w is consistent with U_Q and r. That is, for every $j \in Q$ we have $\sum_{i=1}^{m} r_j \cdot U_{i,j} = w_j$.

The following lemma follows directly from the linearity of L.

Ligero++

PROTOCOL 1 (INTERLEAVED LINEAR CODE TEST). $\mathbb{F}$ *is a prime field and* $L \subset \mathbb{F}^n$ *is a* $[n, k, d]$ *RS code. Let* $U \in \mathbb{F}^{m \times n}$ *be the matrix to be tested.*

- $\mathsf{pp} \leftarrow \mathsf{KeyGen}(1^\lambda)$.
- ***Interleaved testing:***

(1) $\mathcal{V}$ *generates a random vector* $r \in \mathbb{F}^m$ *and sends it to* $\mathcal{P}$.
(2) $\mathcal{P}$ *computes* $w = r^T U \in \mathbb{F}^n$ *and sends it to* $\mathcal{V}$.
(3) $\mathcal{V}$ *checks that* $w \in L$.
(4) $\mathcal{V}$ *generates a random set* $Q \subseteq [n]$ *and* $|Q| = t$ *and sends it to* $\mathcal{P}$.
(5) $\mathcal{V}$ *checks the consistency of* w. *In particular, for* $j \in Q$, $\mathcal{P}$ *and* $\mathcal{V}$ *invoke an IPA protocol on* $U[j]$ *and* r. $\mathcal{V}$ *accepts if all the checks pass, and rejects otherwise.*

Inner product argument

Complexity. *Let the size of the vectors be n. The running time of* Commit *and* Prove *is* $O(n \log n)$ *time for the prover, and the running time of* Verify *is* $O(n)$ *for the verifier. The proof size is* $O(\log^2 n)$.

Ligero++

We set the size of matrix as $\frac{C}{polylog(C)} \times polylog(C)$

Ligero++

Complexity. Let C be the size of matrix U. Then the prover time is $O(C \log C)$, the communication size is $O(\text{polylog} C)$ and the verifier time is $O(C)$.

Ligero

Test-Linear-Constraints-IRS($\mathbb{F}, L = \mathrm{RS}_{\mathbb{F},n,k,\eta}, m, t, \zeta, A, b; U$)

- **Oracle:** A purported L^m-codeword U that should encode a message $x \in \mathbb{F}^{m\ell}$ satisfying $Ax = b$.
- **Interactive testing:**
  (1) $\mathcal{V}$ picks a random vector $r \in \mathbb{F}^{m\ell}$ and sends r to $\mathcal{P}$.
  (2) $\mathcal{V}$ and $\mathcal{P}$ compute

  $$r^T A = (r_{11}, \ldots, r_{1\ell}, \ldots, r_{m1}, \ldots, r_{m\ell})$$

  and, for $i \in [m]$, let $r_i(\cdot)$ be the unique polynomial of degree $< \ell$ such that $r_i(\zeta_c) = r_{ic}$ for every $c \in [\ell]$.

  (3) $\mathcal{P}$ sends the $k + \ell - 1$ coefficients of the polynomial defined by $q(\bullet) = \sum_{i=1}^{m} r_i(\bullet) \cdot p_i(\bullet)$, where p_i is the polynomial of degree $< k$ corresponding to row i of U.
  (4) $\mathcal{V}$ queries a set $Q \subset [n]$ of t random symbols $U[j]$, $j \in Q$.
  (5) $\mathcal{V}$ accepts if the following conditions hold:
    (a) $\sum_{c\in[\ell]} q(\zeta_c) = \sum_{i\in[m],c\in[\ell]} r_{ic} b_{ic}$.
    (b) For every $j \in Q$, $\sum_{i=1}^{m} r_i(\eta_j) \cdot U_{i,j} = q(\eta_j)$.

We will analyze the test under the promise that the (possibly badly formed) U is close to L^m.

The following lemma easily follows by inspection.

Ligero++

PROTOCOL 2 (TESTING LINEAR CONSTRAINTS OVER INTERLEAVED RS CODES). *Let $L[n, k, d]$ be an RS code and $U \in L^m$ be an interleaved code that encodes the message x. $A \in \mathbb{F}^{m\ell \times m\ell}$ is a public matrix such that $Ax = 0$.*

- *Run* $\mathsf{pp}_{\leftarrow}\mathsf{IPA.KeyGen}(1^\lambda)$.
- ***Interleaved testing:***

(1) $\mathcal{V}$ *picks a random value* $r \in \mathbb{F}^{m\ell}$ *and sends* r *to* $\mathcal{P}$.
(2) *Both* $\mathcal{P}$ *and* $\mathcal{V}$ *computes* $a \leftarrow r \times A$ *and calculates polynomials* $a_i(\cdot)$ *such that* $a_i(\zeta_j) = a_{i\ell+j-1}$ *for all* $i \in [m], j \in [\ell]$.
(3) $\mathcal{P}$ *computes polynomials* $p_i(\cdot)$ *such that* $p_i(\eta_j) = U_{ij}$ *for* $i \in [m], j \in [n]$. $\mathcal{P}$ *constructs polynomial* $q(x) = \sum_{i=1}^{m} a_i(x) \cdot p_i(x)$ *and sends it to* $\mathcal{V}$.
(4) $\mathcal{V}$ *checks that* $\sum_{j \in [\ell]} q(\zeta_j) = 0$.
(5) $\mathcal{V}$ *generates a random set* $Q \subseteq [n]$ *and* $|Q| = t$ *and sends it to* $\mathcal{P}$.
(6) *Let* b_j *denote the vector* $(a_0(\eta_j), \ldots, a_{m-1}(\eta_j))$. $\mathcal{V}$ *checks the consistency for* $q(\cdot)$. *In particular, for* $j \in Q$, $\mathcal{P}$ *and* $\mathcal{V}$ *invoke an IPA protocol on* $U[j]$ *and* b_j. $\mathcal{V}$ *accepts if all the checks pass, and rejects otherwise.*

Ligero++

Complexity. Let C be the size of matrix U and assuming a can be computed in linear time. Then the prover time is $O(C \log C)$, the communication size is $O(\text{polylog} C)$ and the verifier time is $O(C)$.

Ligero

Test-Quadratic-Constraints-IRS($\mathbb{F}, L = \mathsf{RS}_{\mathbb{F},n,k,\eta}, m, t, \zeta, a, b; U^x, U^y, U^z$)

- **Oracle:** Purported L^m-codewords U^x, U^y, U^z that should encode messages $x, y, z \in \mathbb{F}^{m\ell}$ satisfying $x \odot y + a \odot z = b$.
- **Interactive testing:**
  1. Let $U^a = \mathsf{Enc}_\zeta(a)$ and $U^b = \mathsf{Enc}_\zeta(b)$.
  2. $\mathcal{V}$ picks a random linear combinations $r \in \mathbb{F}^m$ and sends r to $\mathcal{P}$.
  3. $\mathcal{P}$ sends the $2k-1$ coefficients of the polynomial p_0 defined by $p_0(\bullet) = \sum_{i=1}^{m} r_i \cdot p_i(\bullet)$, where $p_i(\bullet) = p_i^x(\bullet) \cdot p_i^y(\bullet) + p_i^a(\bullet) \cdot p_i^z(\bullet) - p_i^b(\bullet)$, and where p_i^x, p_i^y, p_i^z are the polynomials of degree $< k$ corresponding to row i of U^x, U^y, U^z, and p_i^a, p_i^b are the polynomials of degree $< \ell$ corresponding to row i of U^a, U^b.
  4. $\mathcal{V}$ picks a random index set $Q \subset [n]$ of size t, and queries $U^x[j], U^y[j], U^z[j], j \in Q$.
  5. $\mathcal{V}$ accepts if the following conditions hold:
     (a) $p_0(\zeta_c) = 0$ for every $c \in [\ell]$.
     (b) For every $j \in Q$, it holds that

$$\sum_{i=1}^{m} r_i \cdot \left[U^x_{i,j} \cdot U^y_{i,j} + U^a_{i,j} \cdot U^z_{i,j} - U^b_{i,j} \right] = p_0(\eta_j).$$

The following lemma follows again directly from the description.

Ligero++

Protocol 3 (Testing quadratic constraints over Interleaved RS Codes). *Let λ be the security parameter, $\mathbb{F}$ be a prime field. $L[n,k,d]$ be the intended codeword space. $U^x \in L^m$ encodes the message x, $U^y \in L^m$ encodes the message y, $U_z \in L^m$ encodes the message z. t be the repeat parameter depend on λ. x, y and z satisfies $x \odot y - z = 0$. The testing will accept if U^x, U^y, U^z correctly encodes x, y, z, let η be a root of unity of order n. Let $p_i^x(\cdot)$ be the corresponding polynomial of U_i^x, and we define $p_i^y(\cdot)$ and $p_i^z(\cdot)$ similarly.*

- *Run* $\mathsf{pp} \leftarrow \mathsf{IPA.KeyGen}(1^\lambda)$.
- ***Interleaved testing:***

(1) *$\mathcal{V}$ picks a random value $r \in \mathbb{F}^m$ and sends r to $\mathcal{P}$.*
(2) *$\mathcal{P}$ construct polynomial $q(\cdot)$ defined by $q(\cdot) = \sum_{i=1}^m r_i \cdot p_i(\cdot)$, where $p_i(\cdot) = p_i^x(\cdot)p_i^y(\cdot) - p_i^z(\cdot)$ send the polynomial q to the verifier.*
(3) *$\mathcal{V}$ checks that $\forall i \in [\ell], q(\zeta_j) = 0$.*
(4) *$\mathcal{V}$ generates a random set $Q \subseteq [n]$ and $|Q| = t$ and sends it to $\mathcal{P}$.*
(5) *$\mathcal{V}$ checks the consistency for $q(\cdot)$. In particular, for $j \in Q$, $\mathcal{P}$ and $\mathcal{V}$ invoke an IPA protocol on $U[j]$ and r where $U[j] = U^x[j] * U^y[j] - U^z[j]$. $\mathcal{V}$ accepts if all the checks pass, and rejects otherwise.*

Ligero++

Complexity. Let C be the size of matrix U and assuming a can be computed in linear time. Then the prover time is $O(C \log C)$, the communication size is $O(\text{polylog} C)$ and the verifier time is $O(C)$.