Matching Vectors, Locally Decodable Codes and PIR

Klim Efremenko [1]

[1]Ben Gurion University

February 17, 2020

Table of Contents

Definition of LDC

Motivation

A an error correcting code C is a mapping $C : F^n \mapsto F^N$, $C(x_1, x_2, \ldots x_n) \mapsto (w_1, w_2, \ldots w_N)$:

- Decoding: $D(w_1, w_2, \ldots w_N) = (x_1, \ldots x_n)$
- Error-Correction: D can handle up to d errors

What happens if we want just one symbol x_i and not the entire massage?

Definition of LDC

Motivation

A an error correcting code C is a mapping $C : F^n \mapsto F^N$, $C(x_1, x_2, \ldots x_n) \mapsto (w_1, w_2, \ldots w_N)$:

- Decoding: $D(w_1, w_2, \ldots w_N) = (x_1, \ldots x_n)$
- Error-Correction: D can handle up to d errors

What happens if we want just one symbol x_i and not the entire massage?

Definition of LDC

Definition: Locally Decodable Codes

$C(x_1, x_2, \ldots, x_n) = (w_1, w_2, \ldots, w_N)$
is (q, δ, ε)-LDC if x_i can be recovered from q entries of $C(\vec{x})$
Even if $C(x)$ is corrupted in up-to δN coordinates
With high probability (w.p $1 - \varepsilon$)

There exists a decoding algorithm d_i s.t. $d_i(w_1, w_2, \ldots, w_N) = x_i$
d_i reads only q symbols of $\vec{w}$

Definition of LDC

Definition: Locally Decodable Codes

$C(x_1, x_2, \ldots, x_n) = (w_1, w_2, \ldots, w_N)$
is (q, δ, ε)-LDC if x_i can be recovered from q entries of $C(\vec{x})$
Even if $C(x)$ is corrupted in up-to δN coordinates
With high probability (w.p $1 - \varepsilon$)

There exists a decoding algorithm d_i s.t. $d_i(w_1, w_2, \ldots, w_N) = x_i$
d_i reads only q symbols of $\vec{w}$

Definition of LDC

Definition: Locally Decodable Codes

$C(x_1, x_2, \ldots, x_n) = (w_1, w_2, \ldots, w_N)$
is (q, δ, ε)-LDC if x_i can be recovered from q entries of $C(\vec{x})$
Even if $C(x)$ is corrupted in up-to δN coordinates
With high probability (w.p $1 - \varepsilon$)

There exists a decoding algorithm d_i s.t. $d_i(w_1, w_2, \ldots, w_N) = x_i$
d_i reads only q symbols of $\vec{w}$

Definition of LDC

Definition: Locally Decodable Codes

$C(x_1, x_2, \ldots, x_n) = (w_1, w_2, \ldots, w_N)$
is (q, δ, ε)-LDC if x_i can be recovered from q entries of $C(\vec{x})$
Even if $C(x)$ is corrupted in up-to δN coordinates
With high probability (w.p $1 - \varepsilon$)

There exists a decoding algorithm d_i s.t. $d_i(w_1, w_2, \ldots, w_N) = x_i$
d_i reads only q symbols of $\vec{w}$

Definition of LDC

Definition: Smooth LDC

A code is q-smooth LDC iff

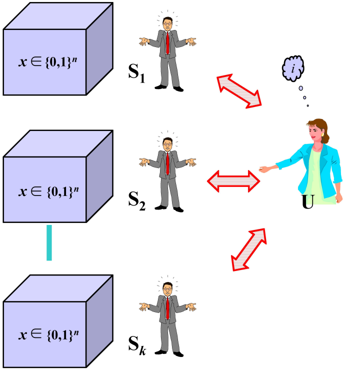
- d_i makes q queries
- Smoothness: each query of d_i is uniformly distributed
- Completeness: $d_i(C(x_1, x_2, \ldots, x_n)) = x_i$

Theorem

q-smooth-LDC is $(q, \delta, q\delta)$-LDC

LDC vs PIR

- LDC $\Rightarrow$ PIR
- Two round PIR when reply of servers small $\Rightarrow$ LDC
- PIR stronger: Large reply, many rounds.



Applications

- PIR, MPC.
- Worst Case average case reductions.
- Pseudo-randomness
- PCP.

Lower Bounds

Lower Bounds

- [KT00]: Lower bound $N = \Omega(n^{q/(q-1)})$
- [GKST01]: q=2 for linear codes $N = 2^{\Omega(n)}$
- [KdW03]: q=2 any codes $N = 2^{\Omega(n)}$
  for $q > 2$, $N = \Omega\left(\left(\frac{n}{\log n}\right)^{1+1/(\lceil q/2 \rceil -1)}\right)$

Lower Bounds

Lower Bounds

- [KT00]: Lower bound $N = \Omega(n^{q/(q-1)})$
- [GKST01]: q=2 for linear codes $N = 2^{\Omega(n)}$
- [KdW03]: q=2 any codes $N = 2^{\Omega(n)}$
  for $q > 2$, $N = \Omega\left(\left(\frac{n}{\log n}\right)^{1+1/(\lceil q/2 \rceil - 1)}\right)$

Lower Bounds

Lower Bounds

- [KT00]: Lower bound $N = \Omega(n^{q/(q-1)})$
- [GKST01]: q=2 for linear codes $N = 2^{\Omega(n)}$
- [KdW03]: q=2 any codes $N = 2^{\Omega(n)}$
  for $q > 2$, $N = \Omega\left((\frac{n}{\log n})^{1+1/(\lceil q/2 \rceil - 1)}\right)$

Upper Bounds

Upper Bounds

- Hadamard code is a two-query LDC $N = 2^n$
- [KSY11, KMZS16] RM and multiplicity codes LDC approaching the optimal rate. Query complexity: $2^{\sqrt{\log n}}$, Rate $1 - \varepsilon$

This Talk

- 4-query LDC's $N = \exp\exp(O(\sqrt{\log n \log\log n}))$
- reduce to 3 query.
- 2-server PIR.

Upper Bounds

Upper Bounds

- Hadamard code is a two-query LDC $N = 2^n$
- [KSY11, KMZS16] RM and multiplicity codes LDC approaching the optimal rate. Query complexity: $2^{\sqrt{\log n}}$, Rate $1 - \varepsilon$

This Talk

- 4-query LDC's $N = \exp\exp(O(\sqrt{\log n \log\log n}))$
- reduce to 3 query.
- 2-server PIR.

Upper and Lower Bounds(LDC)

# queries	Lower Bounds	Upper Bounds
1	Do not exist	
2	2^k	2^k
> 2	$k^{1+\varepsilon(q)}$	$\approx \exp(\exp O(\sqrt[\log q]{\log k}))$ MVC
polylog(k)	-	Poly(k), RM
$2^{\sqrt{\log k}}$	-	$1 + \delta(\varepsilon)k$, [KSY11, KMZS16]

Hadamard Code

Definition

- $C_{HAD} : \mathbb{F}_2^n \mapsto \mathbb{F}_2^{2^n}$
- Let $\vec{m} \in \mathbb{F}_q^n$
- $C_{HAD}(\vec{m}) = (\langle \vec{x}, \vec{m} \rangle)_{x \in \mathbb{F}_2^n}$ is a linear code
- C_{HAD} calculates all linear functionals on $\vec{m}$
- C_{HAD} is a $[\underbrace{2^n}_{\text{codeword}}, \underbrace{n}_{\text{message}}, \underbrace{2^{n-1}}_{\text{minimal distance}}]_2$

Hadamard Code

C_{HAD} is 2-query locally decodable

Decoding procedure

Let $\vec{w} \in \mathbb{F}_2^{2^n}$ be an encoding of $\vec{m}$ with $\delta 2^n$ errors

Choose random $x_1 \in \mathbb{F}_2^n$

Let $x_2 = x_1 + \hat{e}_i$, where e_i i^{th} unit vector

Output $w(x_2) - w(x_1)$ as a value of m_i

Hadamard Code

C_{HAD} is 2-query locally decodable

Decoding procedure

Let $\vec{w} \in \mathbb{F}_2^{2^n}$ be an encoding of $\vec{m}$ with $\delta 2^n$ errors
Choose random $x_1 \in \mathbb{F}_2^n$
Let $x_2 = x_1 + \hat{e}_i$, where e_i i^{th} unit vector
Output $w(x_2) - w(x_1)$ as a value of m_i

Hadamard Code

C_{HAD} is 2-query locally decodable

Decoding procedure

Let $\vec{w} \in \mathbb{F}_2^{2^n}$ be an encoding of $\vec{m}$ with $\delta 2^n$ errors
Choose random $x_1 \in \mathbb{F}_2^n$
Let $x_2 = x_1 + \hat{e}_i$, where e_i i^{th} unit vector
Output $w(x_2) - w(x_1)$ as a value of m_i

Hadamard Code

Theorem

Hadamard code is $(2, \delta, 2\delta) - LDC$

Proof

- Queries are uniformly distributed
- $C(\vec{m})_{x_2} - C(\vec{m})_{x_1} = \langle \vec{m}, x_2 \rangle - \langle m, x_1 \rangle = \langle \vec{m}, x_2 - x_1 \rangle = \langle \vec{m}, \hat{e}_i \rangle = m_i$
- Hadamard code is a 2-smooth-LDC

Hadamard Code

Theorem

Hadamard code is $(2, \delta, 2\delta) - LDC$

Proof

- Queries are uniformly distributed
- $C(\vec{m})_{x_2} - C(\vec{m})_{x_1} = \langle \vec{m}, x_2 \rangle - \langle m, x_1 \rangle = \langle \vec{m}, x_2 - x_1 \rangle = \langle \vec{m}, \hat{e}_i \rangle = m_i$
- Hadamard code is a 2-smooth-LDC

Overview of the construction

Plan

- The definition of S-matching vectors
- The construction of S-matching vectors
- The construction of LDCs based on S-matching vectors
- The construction of S-decoding polynomials
- The decoding algorithm
- Alphabet reduction
- 2 server PIR.
- Representation Theory and LDCs.

S-matching vectors

Fix odd number $m = p_1 p_2 \dots p_k$

Definition

$\{u_i\}_{i=1}^n, u_i \in (\mathbb{Z}_m)^h$ is S-matching :

- $\langle u_i, u_i \rangle = 0$ for every $i \in [n]$.
- $\langle u_i, u_j \rangle \in S$ for every $i \neq j$.
- $0 \notin S$

We want $n >> h$, $|S|$ to be small

S-matching vectors

Fix odd number $m = p_1 p_2 \dots p_k$

Definition

$\{u_i\}_{i=1}^n, u_i \in (\mathbb{Z}_m)^h$ is S-matching :

- $\langle u_i, u_i \rangle = 0$ for every $i \in [n]$.
- $\langle u_i, u_j \rangle \in S$ for every $i \neq j$.
- $0 \notin S$

We want $n >> h$, $|S|$ to be small

S-matching vectors

Fix odd number $m = p_1 p_2 \dots p_k$

Definition

$\{u_i\}_{i=1}^n, u_i \in (\mathbb{Z}_m)^h$ is S-matching :

- $\langle u_i, u_i \rangle = 0$ for every $i \in [n]$.
- $\langle u_i, u_j \rangle \in S$ for every $i \neq j$.
- $0 \notin S$

We want $n >> h$, $|S|$ to be small

Construction of S-matching vectors

Lemma(Grolmusz 2000)

For every integer $m = p_1 p_2 \dots p_r$ there exists a set S_m of size $2^r - 1$ s. t. for every n there exists a family of S-matching vectors $\{u_i\}_{i=1}^n, u_i \in (\mathbb{Z}_m)^h$ s.t. $h \leq \exp(c\sqrt[r]{\log n \log \log^{r-1} n})$.

We will now prove a weaker theorem.

Construction of S-matching vectors

Lemma(Grolmusz 2000)

For set $S_6 = \{1, 3, 4\}$ there exists a family of S-matching vectors $\{u_i\}_{i=1}^n, u_i \in (\mathbb{Z}_6)^h$ s.t. $h \leq \exp(c\sqrt[2]{\log n \log \log n})$.

We will do it in two steps:

- Simple construction of Matching vectors for large set S.
- Reduction of the set S to size 3.

Simple S-matching vectors

Fix some $m = p_1 p_2$ and $\tilde{h} > m$.
Let $[\tilde{h}] = [1, 2, \ldots \tilde{h}]$ set of size $\tilde{h}$

Let $\{A_i\}_{i=1}^{n}$ be all subsets of size $m - 1$ of $[\tilde{h}]$
i.e. $n = \binom{\tilde{h}}{m-1}$
Let $\tilde{u}_i \in (\mathbb{Z}_m)^{\tilde{h}}$ be an indicator vector of A_i
Add an additional coordinate to $\tilde{u}_i$ which is 1 for all $\tilde{u}_i$

Simple S-matching vectors

Claim

- $\langle \tilde{u}_i, \tilde{u}_i \rangle = 0 \bmod m$
- $\langle \tilde{u}_i, \tilde{u}_j \rangle \neq 0 \bmod m$

Proof

- $\langle \tilde{u}_i, \tilde{u}_i \rangle = 0 \bmod m$ since $\tilde{u}_i$ have exactly m ones
- $\langle \tilde{u}_i, \tilde{u}_j \rangle = 1 + |A_i \bigcap A_j|$. Since A_i, A_j are two different sets of size $m - 1 \Rightarrow |A_i \bigcap A_j| < m - 1$.
  Therefore, $\langle \tilde{u}_i, \tilde{u}_j \rangle \neq 0 \bmod m$.

S-matching sets

Tensor product

Definition

Let $\vec{u} \in R^n, \vec{v} \in R^m$ be two vectors then $u \otimes v \in R^{nm}$ such that $(u \otimes v)(i, j) = u(i) \cdot v(j)$

	u_0	u_1	u_2	u_3	u_4
v_0	$v_0 \cdot u_0$	$v_0 \cdot u_1$	$v_0 \cdot u_2$	$v_0 \cdot u_3$	$v_0 \cdot u_4$
v_1	$v_1 \cdot u_0$	$v_1 \cdot u_1$	$v_1 \cdot u_2$	$v_1 \cdot u_3$	$v_1 \cdot u_4$
v_2	$v_2 \cdot u_0$	$v_2 \cdot u_1$	$v_2 \cdot u_2$	$v_2 \cdot u_3$	$v_2 \cdot u_4$

Tensor Product

Fact

$\langle u_1 \otimes v_1, u_2 \otimes v_2 \rangle = \langle u_1, u_2 \rangle \cdot \langle v_1, v_2 \rangle$
$\langle u^{\otimes \ell}, v^{\otimes \ell} \rangle = \langle u, v \rangle^{\ell}$

Proof

$$\langle u^{\otimes \ell}, v^{\otimes \ell} \rangle = \sum_{1 \leq i_1, i_2, \ldots i_l \leq m} \left(\prod_{j=1}^{\ell} u_{i_j} \prod_{j=1}^{\ell} v_{i_j} \right) =$$
$$\left(\sum_{1 \leq i_1 \leq m} u_{i_1} v_{i_1} \right) \ldots \left(\sum_{1 \leq i_\ell \leq m} u_{i_\ell} v_{i_\ell} \right) = \langle u, v \rangle^{\ell}.$$

Reducing S

The set $\{\tilde{u}_i\}_{i=1}^n$ is an S-matching set with $S = \mathbb{Z}_m \backslash 0$
Problem is that set S is too large.

Little Fermat Theorem

If $x \neq 0 \bmod p$ then $x^{p-1} \equiv 1 \bmod p$

Solution

Let us look at $\{\tilde{u}_i^{\otimes(p_1-1)}\}$ then:

$$\langle \tilde{u}_i^{\otimes(p_1-1)}, \tilde{u}_j^{\otimes(p_1-1)} \rangle = \langle u_i, u_j \rangle^{p_1-1} = 0 \text{ or } 1 \bmod p_1$$

It is 0 only iff $\langle u_i, u_j \rangle = 0(p_1)$

Reducing S

The set $\{\tilde{u}_i\}_{i=1}^n$ is an S-matching set with $S = \mathbb{Z}_m \backslash 0$
Problem is that set S is too large.

Little Fermat Theorem

If $x \neq 0 \bmod p$ then $x^{p-1} \equiv 1 \bmod p$

Solution

Let us look at $\{\tilde{u}_i^{\otimes(p_1-1)}\}$ then:

$$\langle \tilde{u}_i^{\otimes(p_1-1)}, \tilde{u}_j^{\otimes(p_1-1)} \rangle = \langle u_i, u_j \rangle^{p_1-1} = 0 \text{ or } 1 \bmod p_1$$

It is 0 only iff $\langle u_i, u_j \rangle \equiv 0(p_1)$

Reducing S

The set $\{\tilde{u}_i\}_{i=1}^{n}$ is an S-matching set with $S = \mathbb{Z}_m \backslash 0$
Problem is that set S is too large.

Little Fermat Theorem

If $x \neq 0 \bmod p$ then $x^{p-1} \equiv 1 \bmod p$

Solution

Let us look at $\{\tilde{u}_i^{\otimes(p_1-1)}\}$ then:

$$\langle \tilde{u}_i^{\otimes(p_1-1)}, \tilde{u}_j^{\otimes(p_1-1)} \rangle = \langle u_i, u_j \rangle^{p_1-1} = 0 \text{ or } 1 \bmod p_1$$

It is 0 only iff $\langle u_i, u_j \rangle \equiv 0(p_1)$

Reducing S

Definition

Set $u_i \triangleq (p_2 \tilde{u}_i^{\otimes(p_1-1)}, p_1 \tilde{u}_i^{\otimes(p_2-1)})$ $u_i \in (\mathbb{Z}_m)^h$, where $h = \tilde{h}^{p_1-1} + \tilde{h}^{p_2-1}$

Claim

Set $\{u_i\}_{i=1}^n$ is an S-matching set with $|S| = 3$.

Reducing S

Proof

Let us prove that $\langle u_i, u_i \rangle = 0$

$$\langle u_i, u_i \rangle =$$
$$\langle (p_2 \tilde{u}_i^{\otimes p_1 - 1}, p_1 \tilde{u}_i^{\otimes p_2 - 1}), (p_2 \tilde{u}_i^{\otimes p_1 - 1}, p_1 \tilde{u}_i^{\otimes p_2 - 1}) \rangle =$$
$$p_2 \langle \tilde{u}_i, \tilde{u}_i \rangle^{p_1 - 1} + p_1 \langle \tilde{u}_i, \tilde{u}_i \rangle^{p_2 - 1} = 0$$

Reducing S

Chinese Reminder Theorem (CRT)

For every a, b there exists an unique $x \in \mathbb{Z}_m$ s.t.
$x \equiv a \bmod p_1$ and $x \equiv b \bmod p_2$.

Proof cont.

Let us now prove that $\langle u_i, u_j \rangle$ may take only 3 values.

$$\langle u_i, u_j \rangle = p_2 \langle \tilde{u}_i, \tilde{u}_j \rangle^{p_1 - 1} + p_1 \langle \tilde{u}_i, \tilde{u}_j \rangle^{p_2 - 1} \bmod p_1 p_2$$

Modulo p_1 it is either p_2 or 0. The same for p_2. We have 4 possibilities for $(\langle u_i, u_j \rangle \bmod p_1, \langle u_i, u_j \rangle \bmod p_2)$
$(0, 0)$ happens only if $\langle \tilde{u}_i, \tilde{u}_j \rangle = 0 \bmod p_1 p_2$.

Reducing S

Proof: the size of MV

We will prove only for $r = 2$.
Take $p_1 \approx p_2$. We have constructed $n = \binom{\tilde{h}}{p_1 p_2 - 1} (\sim\sim \tilde{h}^m)$
S-matching vectors $u_i \in (\mathbb{Z}_m)^h$ where
$h = \tilde{h}^{p_1 - 1} + \tilde{h}^{p_2 - 1} (\sim \tilde{h}^{p_2} = \tilde{h}^{\sqrt{m}})$.
We will get the desired result.

Construction of the code

Fix $\gamma \in \mathbb{F}_{2^t}$ generator of the mult. group of size m
i.e. $\gamma^m = 1$, $\gamma^i \neq 1$ for $i < m$
Fix S-matching vectors $\{u_i\}_{i=1}^n, u_i \in (\mathbb{Z}_m)^h$

Definition

A code $C : \mathbb{F}^n \mapsto \mathbb{F}^{m^h}$ is a linear code
$C(m_1, m_2, \ldots, m_n) = \sum m_i C(\hat{e}_i)$ by linearity
$C(\hat{e}_i) = (\gamma^{\langle u_i, x\rangle})_{x \in (\mathbb{Z}_m)^h}$

Rate

$N = m^h \leq \exp\exp(c\sqrt[r]{\log n \log\log^{r-1} n})$

S-decoding polynomials

Definition

A polynomial $P \in \mathbb{F}[x]$ is called an S-decoding iff:

- $\forall s \in S \; P(\gamma^s) = 0$,
- $P(\gamma^0) = P(1) = 1$.

Key Observation

Given S-matching vectors:

- $P(\gamma^{\langle u_i, u_i \rangle}) = 1$ for all i, since $\langle u_i, u_i \rangle = 0$
- $P(\gamma^{\langle u_i, u_j \rangle}) = 0$ for all $i \neq j$, since $\langle u_i, u_j \rangle \in S$

S-decoding polynomials

Definition

A polynomial $P \in \mathbb{F}[x]$ is called an S-decoding iff:

- $\forall s \in S \; P(\gamma^s) = 0$,
- $P(\gamma^0) = P(1) = 1$.

Key Observation

Given S-matching vectors:

- $P(\gamma^{\langle u_i, u_i \rangle}) = 1$ for all i, since $\langle u_i, u_i \rangle = 0$
- $P(\gamma^{\langle u_i, u_j \rangle}) = 0$ for all $i \neq j$, since $\langle u_i, u_j \rangle \in S$

S-decoding polynomials

Lemma

For every set S there exists an S-decoding polynomial with at most $|S| + 1$ monomials

Proof

Set $\tilde{P}(x) = \prod_{s \in S}(x - \gamma^s)$ then:
$\forall s \in S\ \tilde{P}(\gamma^s) = 0$
$\tilde{P}(1) = \tilde{P}(\gamma^0) \neq 0$, since $0 \notin S$
Set $P(x) = \tilde{P}(x)/\tilde{P}(1)$
degree $P(x) = |S|$, so $P(x)$ has $|S| + 1$ monomials

Decoding Algorithm

Set S-decoding polynomial
$P(x) = a_0 + a_1 x^{b_1} + a_2 x^{b_2} \ldots a_{q-1} x^{b_{q-1}}$

Decoding algorithm

Given i and a codeword w:

- Choose $v \in (\mathbb{Z}_m)^h$ at random.
- Query $w(v), w(v + b_1 u_i), \ldots w(v + b_{q-1} u_i)$
- $$c_i = a_0 w(v) + a_1 w(v + b_1 u_i) \ldots + a_{q-1} w(v + b_{q-1} u_i).$$
- Output $\gamma^{-<u_i,v>} c_i$

Decoding Algorithm

Set S-decoding polynomial
$P(x) = a_0 + a_1 x^{b_1} + a_2 x^{b_2} \ldots a_{q-1} x^{b_{q-1}}$

Decoding algorithm

Given i and a codeword w:

- Choose $v \in (\mathbb{Z}_m)^h$ at random.
- Query $w(v), w(v + b_1 u_i), \ldots w(v + b_{q-1} u_i)$
- $$c_i = a_0 w(v) + a_1 w(v + b_1 u_i) \ldots + a_{q-1} w(v + b_{q-1} u_i).$$
- Output $\gamma^{-<u_i, v>} c_i$

Decoding Algorithm

Set S-decoding polynomial
$P(x) = a_0 + a_1 x^{b_1} + a_2 x^{b_2} \dots a_{q-1} x^{b_{q-1}}$

Decoding algorithm

Given i and a codeword w:

- Choose $v \in (\mathbb{Z}_m)^h$ at random.
- Query $w(v), w(v + b_1 u_i), \dots w(v + b_{q-1} u_i)$
  Note $v + b_j u_i$ are uniformly distributed
- $$c_i = a_0 w(v) + a_1 w(v + b_1 u_i) \dots + a_{q-1} w(v + b_{q-1} u_i).$$
- Output $\gamma^{-<u_i, v>} c_i$

Decoding Algorithm

Set S-decoding polynomial
$P(x) = a_0 + a_1 x^{b_1} + a_2 x^{b_2} \dots a_{q-1} x^{b_{q-1}}$

Decoding algorithm

Given i and a codeword w:

- Choose $v \in (\mathbb{Z}_m)^h$ at random.
- Query $w(v), w(v + b_1 u_i), \dots w(v + b_{q-1} u_i)$
- $$c_i = a_0 w(v) + a_1 w(v + b_1 u_i) \dots + a_{q-1} w(v + b_{q-1} u_i).$$
- Output $\gamma^{-<u_i,v>} c_i$

Decoding Algorithm

Set S-decoding polynomial
$P(x) = a_0 + a_1 x^{b_1} + a_2 x^{b_2} \ldots a_{q-1} x^{b_{q-1}}$

Decoding algorithm

Given i and a codeword w:

- Choose $v \in (\mathbb{Z}_m)^h$ at random.
- Query $w(v), w(v + b_1 u_i), \ldots w(v + b_{q-1} u_i)$
- $$c_i = a_0 w(v) + a_1 w(v + b_1 u_i) \ldots + a_{q-1} w(v + b_{q-1} u_i).$$
- Output $\gamma^{-<u_i,v>} c_i$

Decoding Algorithm

Lemma

The algorithm decodes the i^{th} symbol of the code

Proof

The decoding algorithm is a linear mapping $d_i : \mathbb{F}^N \mapsto \mathbb{F}$
Therefore, $d_i(C(\sum_j m_j \hat{e}_j)) = \sum_j m_j d_i(C(\hat{e}_j))$
It is enough to prove that $d_i(C(\hat{e}_j)) = \delta_{ij}$

Decoding Algorithm

Recall

$C(\hat{e}_j) = \gamma^{\langle u_j, x\rangle}$, $P(\gamma^{\langle u_i, u_j\rangle}) = \delta_{ij}$

Proof.(Continue)

$$d_i(C(\hat{e}_j)) =$$
$$\gamma^{-<u_i,v>}(a_0\gamma^{\langle u_j,v\rangle} + a_1\gamma^{\langle u_j,v+b_1u_i\rangle} + a_2\gamma^{\langle u_j,v+b_2u_i\rangle}\dots) =$$
$$\gamma^{<u_j-u_i,v>}(a_0 + a_1\gamma^{\langle u_j,u_i\rangle b_1} + a_2\gamma^{\langle u_j,u_i\rangle b_2}\dots) =$$
$$\gamma^{<u_j-u_i,v>}P(\gamma^{\langle u_j,u_i\rangle}) = \delta_{ij}$$

LDC

Theorem

The code defined above is $(q, \delta.q\delta)$-LDC where q is the number of monomials of the S-decoding polynomial

Proof

- The decoding algorithm makes q queries
- Each query is uniformly distributed
- Decoding alg. returns the correct answer if all queries are not damaged
- The code is q-smooth LDC

LDC

Theorem

The code defined above is $(q, \delta.q\delta)$-LDC where q is the number of monomials of the S-decoding polynomial

Proof

- The decoding algorithm makes q queries
- Each query is uniformly distributed
- Decoding alg. returns the correct answer if all queries are not damaged
- The code is q-smooth LDC

LDC

Theorem

The code defined above is $(q, \delta.q\delta)$-LDC where q is the number of monomials of the S-decoding polynomial

Proof

- The decoding algorithm makes q queries
- Each query is uniformly distributed
- Decoding alg. returns the correct answer if all queries are not damaged
- The code is q-smooth LDC

LDC

Theorem

The code defined above is $(q, \delta.q\delta)$-LDC where q is the number of monomials of the S-decoding polynomial

Proof

- The decoding algorithm makes q queries
- Each query is uniformly distributed
- Decoding alg. returns the correct answer if all queries are not damaged
- The code is q-smooth LDC

3-Query LDC

3-Query LDC

We can set $m = 511 = 7 * 73$ and construct an S-decoding polynomial with 3-monomials.

2- server PIR

Private Information Retrieval

- Note that LDCs imply 3 server PIR.
- No sub-exponential 2-query LDC exist.
- 2 server PIR exist with sub-linear CC.
- Before [Dvir-Gopi] believed not to exist.

2- server PIR

Private Information Retrieval: The scheme

- Servers has database $D_1, \ldots D_n$ of bits.
- Let $\{u_i\}_{i=1}^n, u_i \in (\mathbb{Z}_6)^h$ is S-matching vectors.
- $\langle u_i, u_i \rangle = 0$, $\langle u_i, u_j \rangle \in \{1, 3, 4\}$.
- User pick $r \in (\mathbb{Z}_2)^h$ and send to one server r and to the second one $r + u_i$
- Each server on query r replies:
  $C(r) = \sum_{j=1}^n D_j (-1)^{\langle u_j, r \rangle}$
  $V(r) = \sum_{j=1}^n D_j u_j (-1)^{\langle u_j, r \rangle}$
- Compute
  $2(-1)^{\langle u_i, r \rangle} D_i = C(r) + C(r + u_i) - \langle V(r), u_i \rangle - \langle V(r + u_i), u_i \rangle$.

2- server PIR

Private Information Retrieval: Proof

- $C(r) + C(r + u_i) - \langle V(r), u_i \rangle - \langle V(r + u_i), u_i \rangle =$
  $\sum_{j=1}^{n} D_j(-1)^{\langle u_j, r \rangle} + \sum_{j=1}^{n} D_j(-1)^{\langle u_j, r+u_i \rangle} -$
  $\sum_{j=1}^{n} \langle u_j, u_i \rangle D_j(-1)^{\langle u_j, r \rangle} - \sum_{j=1}^{n} \langle u_j, u_i \rangle D_j(-1)^{\langle u_j, r+u_i \rangle} =$
  $\sum_{j=1}^{n} D_j(-1)^{\langle u_j, r \rangle} \{1 + (-1)^{\langle u_j, u_i \rangle} - \langle u_i, u_j \rangle - \langle u_i, u_j \rangle (-1)^{\langle u_j, u_i \rangle}\}$
- Check that if $\langle u_i, u_j \rangle = 0$ red part is 2. Else if $\langle u_i, u_j \rangle \in \{1, 3, 4\}$ red part is zero modulo 3.

Alphabet Reduction

- The code we have defined is over some field $\mathbb{F}_{2^n}$. We can reduce the alphabet size to 2.