Sparse Packing for CKKS

Efe İzbudak
ephemer4l@liberior.org

July 30, 2024

Some number theory

Lemma

If α is odd and $n \geq 3$ then $\alpha^{2^{n-2}} = 1$ in $\mathbb{Z}/2^n\mathbb{Z}$

Proof.

Proof by induction on n.
For $n = 3$ we have $1^2 \equiv 3^2 \equiv 5^2 \equiv 7^2 \equiv 1 \mod 8$.
Suppose the lemma holds for n. Then $\alpha^{2^{n-2}} = 1 + 2^n t$ for some $t \in \mathbb{Z}$.
Squaring both sides:

$$\begin{aligned} \alpha^{2^{n-1}} &= 1 + 2^{n+1}t + 2^{2n}t^2 \\ &\equiv 1 \mod 2^{n+1} \end{aligned}$$

as desired. $\square$

Some number theory

Theorem

The order of 5 *in* $\mathbb{Z}/2^n\mathbb{Z}$ *is* 2^{n-2} *if* $n \geq 3$.

Proof.

By the lemma $5^{2^{n-2}} \equiv 1 \mod 2^n$. Then the order of 5 must divide 2^{n-2} by a group theory argument.
We wish to show that $5^{2^{n-3}} \not\equiv 1 \mod 2^n$. Then the order of 5 cannot be anything that divides 2^{n-3}:
Suppose $5^{2^{n-k}} = 1$ for some $k > 3$ then

$$(5^{2^{n-k}})^{2^{k-3}} = 5^{2^{n-k} \cdot 2^{k-3}} = 5^{2^{n-3}} \equiv 1 \mod 2^n,$$

a contradiction.
If the order must divide 2^{n-2} and it cannot be anything that divides 2^{n-3} then it must be 2^{n-2}.

Some number theory

Proof. (Cont.)

We show that $5^{2^{n-3}} \equiv 1 + 2^{n-1} \mod 2^n$ by induction on n.

For $n = 3$ we have $5 \equiv 1 + 4 \mod 8$ as desired.

Suppose the claim holds for n. Then $5^{2^{n-3}} = 1 + 2^{n-1} + 2^n t$ for some $t \in \mathbb{Z}$.

Square both sides to get:

$$\begin{aligned} 5^{2^{n-2}} &= 1 + 2^{2n-2} + 2^{2n}t^2 + 2(2^{n-1} + 2^n t + 2^{2n-1}t) \\ &= 1 + 2^{2n-2} + 2^{2n}t^2 + 2v \text{ for some } v \in \mathbb{Z} \\ &\equiv 1 + 2^n \mod 2^{n+1} \end{aligned}$$

as desired. $\square$

Some number theory

Corollary

5 *generates a subgroup of order* 2^{n-2} *in the unit group* $(\mathbb{Z}/2^n\mathbb{Z})^*$ *which has order* 2^{n-1}.

Sparse Packing: Setup

- $R = \mathbb{Z}[X]/(X^N + 1)$ where N is a power of 2.
- $n \geq 2$ is a divisor of N.
- Δ is our scale to save precision.
- R has a subring isomorphic to $\mathbb{Z}[X^{N/n}]/(X^N + 1)$ (by the lattice isomorphism theorem)
- Identify $\mathbb{Z}[X^{N/n}]/(X^N + 1)$ with $\mathbb{Z}[Y]/(Y^n + 1)$ via $X^{N/n} \mapsto Y$.

Example

Let $N = 8$ and $n = 4$ so $N/n = 2$.

Let

$$a_0 X^0 + a_1 X^2 + a_2 X^4 + a_3 X^6 \in \mathbb{Z}[X^2]/(X^8 + 1)$$

Then we can view this element as

$$a_0 Y^0 + a_1 Y + a_2 Y^2 + a_3 Y^3 \in \mathbb{Z}[Y]/(Y^4 + 1)$$

Sparse Packing: Decoding

- Let $p(X^{N/n}) \in \mathbb{Z}[X^{N/n}]/(X^N+1)$ and $\xi = e^{-\pi i/n}$ a primitive $(2n)$-th root of unity.
- Denote $\xi_j = \xi^{5^j \mod 2n}$.
- We use the canonical embedding $\mathbb{Z}[Y]/(Y^n+1) \hookrightarrow \mathbb{C}^n$ to decode an element

1. View $p(X^{N/n})$ as an element of $\mathbb{Z}[Y]/(Y^n+1)$ via $X^{N/n} \mapsto Y$.
2. Evaluate $p(Y)$ at points $\{\xi_j \mid 0 \leq j < n/2\}$ and construct
$$(p(\xi_0), p(\xi_1), \ldots, p(\xi_{\frac{n}{2}-1}))$$
3. Return
$$\frac{1}{\Delta} \cdot (p(\xi_0), p(\xi_1), \ldots, p(\xi_{\frac{n}{2}-1})$$

Sparse Packing: Decoding

Remark (Evaluation at primitive $(2N)$-th roots)

Let $\zeta = e^{-\pi i/N}$, a primitive $(2N)$-th root of unity Then

$$\zeta^{N/n} = \left(e^{-\pi i/N}\right)^{N/n} = e^{-\pi i/n} = \xi$$

$\implies$ *Evaluation at primitive $(2N)$-th roots of unity will give the above result concatenated N/n times.*

Sparse Packing: Decoding

Example

Let N=8, n=4, and scale $\Delta = 64$. Then $R = \mathbb{Z}[X^{N/n}]/(X^N + 1) = \mathbb{Z}[X^2]/(X^8 + 1)$. Pick

$$p(X^{N/n}) = 160 + 136X^2 + 96X^4 + 91X^6 \in \mathbb{Z}[X^2]/(X^8 + 1)$$

which is just

$$p(Y) = 160 + 136Y + 96Y^2 + 91Y^3 \in \mathbb{Z}[Y]/(Y^4 + 1)$$

Set

$$\xi = e^{-\pi i/4}$$

Sparse Packing: Decoding

Example (Cont.)

We evaluate $p(Y)$ at points $\{\xi_0 = e^{-\pi i/4}, \xi_1 = e^{-5\pi i/4}\}$ to get

$$\nu = (191.82 + 256.513i, 128.18 - 64.513i) \in \mathbb{C}^2$$

we rescale to attain

$$\frac{1}{\Delta} \cdot \nu = (2.997 + 4.008i, 2.003 - 1.008i) \in \mathbb{C}^2$$

Sparse Packing: Decoding

Remark

Note that

$$\{\zeta_j^2 \mid 0 \leq j < N/2\} = \{e^{-\pi i/4}, e^{-5\pi i/4}, e^{-\pi i/4}, e^{-5\pi i/4}\}$$

is just ξ_0, ξ_1 *repeated* 2 *times and so the evaluation of* $p(X^2)$ *at* ζ_j *followed by scaling gives*

$$\nu = (2.997 + 4.008i, 2.003 - 1.008i, 2.997 + 4.008i, 2.003 - 1.008i) \in \mathbb{C}^4$$

Sparse Packing: Encoding

- Let $(a_0, a_1, \ldots, a_{\frac{n}{2}-1}) \in \mathbb{C}^{n/2}$ and $\xi = e^{-\pi i/n}$ be a primitive $(2n)$-th root of unity.
- Denote $\xi_j = \xi^{5^j \mod 2n}$.
- U is the $n/2 \times n$ Vandermonde matrix of elements

$$\{\xi_j \mid 0 \leq j < n/2\}$$

- $\mathbb{H}^n$ is the vector space of elements of the form

$$(c_0, c_1, \ldots, c_{\frac{n}{2}-1}, \overline{c_{\frac{n}{2}-1}}, \ldots, \overline{c_1}, \overline{c_0}) \text{ where } c_i \in \mathbb{C}$$

- $\varphi : \mathbb{C}^{n/2} \to \mathbb{H}^n$ is an isomorphism such that

$$\varphi(c_0, c_1, \ldots, c_{\frac{n}{2}-1}) = (c_0, c_1, \ldots, c_{\frac{n}{2}-1}, \overline{c_{\frac{n}{2}-1}}, \ldots, \overline{c_1}, \overline{c_0})$$

Sparse Packing: Encoding

Orthogonal basis in $\mathrm{Im}(\sigma)$

- $\{\sigma(1), \sigma(Y), \sigma(Y^2), \dots, \sigma(Y^{n-1})\}$ is clearly a $\mathbb{Z}$-basis for $\mathrm{Im}(\sigma)$ since σ is an isomorphism and for any $\sigma(a_0 + a_1 Y + \cdots + a_{n-1} Y^{n-1})$ we can write

$$a_0\sigma(1) + a_1\sigma(Y) + \cdots + a_{n-1}\sigma(Y^{n-1})$$

- For orthogonality we note that

$$\sigma(Y^k) = (1^k, \xi_1^k, \xi_2^k, \dots, \xi_{\frac{n}{2}-1}^k)$$

It can then be directly calculated that the Hermitian inner product $\langle \sigma(Y^i), \sigma(Y^j) \rangle$ is 0 when $i = j$ and n otherwise.

Sparse Packing: Encoding

Inverse of U

- The orthogonality of $\{\sigma(1), \sigma(Y), \sigma(Y^2), \ldots, \sigma(Y^{n-1})\}$ is directly equivalent to the matrix UU^* being diagonal where U^* is the adjoint.
- Each element of $\{\sigma(1), \sigma(Y), \sigma(Y^2), \ldots, \sigma(Y^{n-1})\}$ has norm n. Together with the above this implies that

$$UU^* = n \cdot I$$

where I is the $n/2 \times n/2$ identity matrix.

$\implies \frac{1}{n}U^*$ is the inverse of U.

Sparse Packing: Encoding

We encode $(a_0, a_1, \ldots, a_{\frac{n}{2}-1})$ into $\mathbb{Z}[Y]/(Y^n+1) = \mathbb{Z}[X^{N/n}]/(X^N+1)$.

1. Scale vector by Δ for accuracy.
2. Apply φ and attain $\nu_0 = (a_0, a_1, \ldots, a_{\frac{n}{2}-1}, \overline{a_{\frac{n}{2}-1}}, \ldots, \overline{a_1}, \overline{a_0}) \in \mathbb{H}^n$
3. Take the orthogonal projection of ν_0 onto the orthogonal ideal lattice basis $\{\sigma(1), \sigma(Y), \ldots, \sigma(Y^{n-1})\}$ in the image of the canonical embedding σ to get ν_1.
4. Apply randomized rounding to get integer valued vector ν_2.
5. Encode vector into $p(Y) = \mathbb{Z}[Y]/(Y^n+1)$ using the inverse of the canonical embedding σ.
6. View as an element of $\mathbb{Z}[X^{N/n}]/(X^N+1)$ using $Y \mapsto X^{N/n}$.

Sparse Packing: Encoding

Example

Let $N = 8$, $n = 4$ and scale $\Delta = 64$. Then $R = \mathbb{Z}[X^{N/n}]/(X^N + 1) = \mathbb{Z}[X^2]/(X^8 + 1)$. Pick

$$\nu = (3 + 4i, 2 - i) \in \mathbb{C}^2$$

Set

$$\xi = e^{-\pi i/4}$$

Scale ν to $(192 + 256i, 128 - 64i)$ and extend to:

$$\nu_0 = (192 + 256i, 128 - 64i, 128 + 64i, 192 - 256i) \in \mathbb{H}^4$$

Sparse Packing: Encoding

Example (Cont.)

Now we take orthogonal projection of ν_0 onto the basis $\mathcal{S} = \{\sigma(1), \sigma(Y), \ldots, \sigma(Y^{n-1})\}$ which corresponds to the rows of the matrix

$$\begin{bmatrix} U \\ \hline \overline{U} \end{bmatrix}$$

The projection of ν_0 onto $\mathcal{S}$ is given by the sum

$$\sum_{i=0}^{n-1} \frac{\langle \nu, \sigma(Y^i) \rangle}{\langle \sigma(Y^i), \sigma(Y^i) \rangle} \sigma(Y^i)$$

Calculating this we get $\nu_1 = \nu_0$ and coordinate-wise random rounding is redundant.

Sparse Packing: Encoding

Example (Cont.)

Now we encode ν_1 into $\mathbb{Z}/(Y^n + 1)$ using the canonical embedding.

Note that we have

$$\begin{bmatrix} U \\ \overline{U} \end{bmatrix} \mathfrak{m} = \begin{bmatrix} \nu \\ \overline{\nu} \end{bmatrix}$$

for $\mathfrak{m}$ a coefficient vector for a polynomial $p(Y)$. Then multiplying by the inverse $\frac{1}{n}\begin{bmatrix} U \\ \overline{U} \end{bmatrix}^* = \begin{bmatrix} \overline{U^T} & U^T \end{bmatrix}$ we attain

$$\mathfrak{m} = \frac{1}{n}(\overline{U^T}\nu + U^T\overline{\nu})$$

By direct calculation we get $\mathfrak{m} = (160, -91, -96, -136)$ corresponding to $p(Y) = 160 - 91Y - 96Y^2 - 136Y^3$ which corresponds to $p(X^2) = 160 - 91X^2 - 96X^4 - 136X^6$.

Sparse Packing: Encoding

Remark (Compatibility with regular encoding)

Regular encoding with primitive root $\xi = e^{-\pi i/4}$ *of a vector* μ *such that*

$$\mu = \underbrace{(3+4i, 2-i, 3+4i, 2-i, \ldots, 3+4i, 2-i)}_{N/n \text{ times}} \in \mathbb{C}^{N/2}$$

will give the exact same result as above.

Complexity implications

Since the size of the Vandermonde matrix is smaller, the total complexity of operations such as rotations and linear transformations on sparsely packed ciphertexts are lower.

Python implementation: setup

```
import numpy as np
import random

class SparseEncoder:
    def __init__(self, N: int, n: int, scale: int):
        self.N = N
        self.n = n
        self.scale = scale
        root = np.exp((-1j * np.pi) / self.n)

        self.roots_of_unity = [(root) ** (5**j % ( 2 * self.n) for
         ↪  j in range(0, self.n // 2)]
        self.vandermonde = np.vander(self.roots_of_unity, self.n,
         ↪  increasing=True)
```

Python implementation: decoding

```
def decode(self, polynomial: np.polynomial.polynomial.Polynomial):
    # View the polynomial in Z[x^(N/n)]/(x^N+1) as an element of
    ↪ Z[y]/(y^n+1)
    empty = (self.N) * [0]
    for i in range(self.n):
        empty[i] = polynomial.coef[(self.N // self.n) * i]
    polynomial = np.polynomial.Polynomial(empty)
    # Evaluate the polynomial at roots of unity
    ev = polynomial(self.roots_of_unity)
    # Scale for precision
    scaled_ev = ev / self.scale
    return scaled_ev
```

Python implementation: streching and rounding

```
@staticmethod
def pi(vector: np.ndarray):
    vector_conjugate = [np.conjugate(x) for x in vector[::-1]]
    return np.concatenate([vector, vector_conjugate])

@staticmethod
def coordinate_wise_random_rounding(vector: np.ndarray):
    r = vector - np.floor(vector)
    f = np.array(
      [np.random.choice([c, c - 1], 1, p=[1 - c, c]) for c in r]
    ).reshape(-1)
    rounded_coordinates = vector - f
    rounded_coordinates = [int(coeff) for coeff in
     ↪  rounded_coordinates]
    return rounded_coordinates
```

Python implementation: projection

```
def proj(self, vector):
    basis = np.vstack(
        (self.vandermonde, np.flip(self.vandermonde.conj(), axis=0))
    ).T
    # Find the coordinates of the projection in terms of the basis
    coordinates = np.array(
        [np.real((np.vdot(vector, u) / np.vdot(u, u))) for u in
         ↪  basis]
    )
    # Round coordinates to get integer coefficients
    rounded_coordinates =
     ↪  __class__.coordinate_wise_random_rounding(coordinates)
    # Calculate the projection vector using the coordinates
    return np.matmul(basis.T, rounded_coordinates)
```

Python implementation: encoding

```
def encode(self, vector):
    # Convert possible list to np.array
    vector = np.array(vector)

    # View as an element of H^n instead of C^(n/2)
    expansion = __class__.pi(vector)
    # Scale up for better precision
    scaled_expansion = self.scale * expansion
    # Project to orthogonal basis of im(embedding) and snip the
    ↪  conjugate
    projection = self.proj(scaled_expansion)[: self.n // 2]
```

```python
# Get coefficients of the polynomial using CRT
coefficients = (
    np.real(
        np.dot(self.vandermonde.conj().T, projection)
        + np.dot(self.vandermonde.T, projection.conj())
    )
    / self.n
)

# Round numpy's numerical imprecision
rounded_coeff = np.round(np.real(coefficients)).astype(int)
```

```
# View as an element of the ring Z[x^(N/n)]/(x^N+1) by
↪  shifting coefficients
empty = (self.N) * [0]
for i in range(self.n):
    empty[self.N // self.n * i] = rounded_coeff[i]

return np.polynomial.Polynomial(empty)
```