CKKS Notes

Blog post part 1

Encoding and Decoding

Basic ideas

Encoding and decoding in CKKKS uses this idea of polynomial interpolation. An n-degree polynomial can be uniquely determined by n points.

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix}.$$

Encoding Vectors

- Given a plaintext vector $(y_0, y_1, \ldots, y_{n-1})$, we want to encode it as a polynomial we can work with in the CKKKS scheme
- To do this we assume the vector is the result of evaluating the polynomial at the n-th roots of unity $(1, \omega, \omega^2, \ldots, \omega^{n-1})$
- Then to compute the coefficients that uniquely determine our plaintext polynomial $(a_0, a_1, \ldots, a_{n-1})$ we need only solve the linear system pictures above by inverting the Vandermond matrix pictured above using $(1, \omega, \omega^2, \ldots, \omega^{n-1})$ as the inputs to the polynomial on each row

Decoding Polynomials

- Evaluate the polynomial at the n roots of unity $(1, \omega, \omega^2, \ldots, \omega^{n-1})$, to get an output vector $(y_0, y_1, \ldots, y_{n-1})$
- This output vector is the original plaintext vector

More accurate ideas

In CKKKS our input messages are vectors of complex numbers, note that $\mathbb{R} \subset \mathbb{C}$, so we can work with reals as well. We can perform homomorphic operations on polynomial rings, so if we can encode our input vectors as polynomials then we can use these relations to perform operations on encrypted data.

Specifically we want to work with N dimensional vectors, and N degree polynomials, so we want to define the isomorphic map

- $\sigma : \mathbb{C}[X]/(X^N + 1) \to \mathbb{C}^N$
- $\sigma^{-1} : \mathbb{C}^N \to \mathbb{C}[X]/(X^N + 1)$

The n-th cyclotomic polynomial does not necessarily have n roots IE it is not necessarily of degree n. In the example in the article they chose the simplest 4th degree cyclotomic polynomial $X^4 + 1$ for their example, but they could have used any 4th degree cyclotomic to provide the map from C^N.

The n-th cyclotomic polynomial has degree $\phi(n)$. So if $N = 2^k$ and we want a cyclotomic of degree N we can choose the $2N$-th cyclotomic since $\phi(2^{k+1}) = 2\phi(2^k)$.

This is reasoning behing this statement

> We will denote the degree of our polynomial degree modulus by N, which will be a power of 2. We denote the m-th cyclotomic polynomial (note that $M = 2N$) by $\Phi_M(X) = X^N + 1$. The plaintext space will be the polynomial ring $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$. Let us denote by ξ_M, the M-th root of unity : $\xi_M = e^{2i\pi/M}$.

Using this general definition for the n-th cyclotomic polynomial

$$\Phi_n(x) = \prod_{\substack{1 \le k \le n \\ \gcd(k,n)=1}} \left(x - e^{2i\pi\frac{k}{n}}\right).$$

and the fact that we know N is a power of 2 in our case, it should be clear why the roots of the cyclotomic are these powers

> The idea is simple: To decode a polynomial $m(X)$ into a vector z, we evaluate the polynomial on certain values, which will be the roots of the cyclotomic polynomial $\Phi_M(X) = X^N + 1$. Those N roots are : $\xi, \xi^3, \ldots, \xi^{2N-1}$.

Where $\xi = e^{2i\pi/M}$

Once we have our the roots of our cyclotomic we can encode plaintext vectors as polynomials by solving this linear system.

> $A\alpha = z$, with A the Vandermonde matrix of the $(\xi^{2i-1})_{i=1,\ldots,N}$, α the vector of the polynomial coefficients, and z the vector we want to encode.
>
> Therefore we have that : $\alpha = A^{-1}z$, and that $\sigma^{-1}(z) = \sum_{i=0}^{N-1} \alpha_i X^i \in \mathbb{C}[X]/(X^N + 1)$.

Remember that what uniquely determines the polynomial is it coefficients, and what determines the vector is just its coordinates. You always know the roots of the cyclotomic that

your working with and you either know the coefficients of the polynomial or the coordinates of your vector. Given this information you can always compute one from the other, and thats the relationship your exploiting the encode and decode messages.

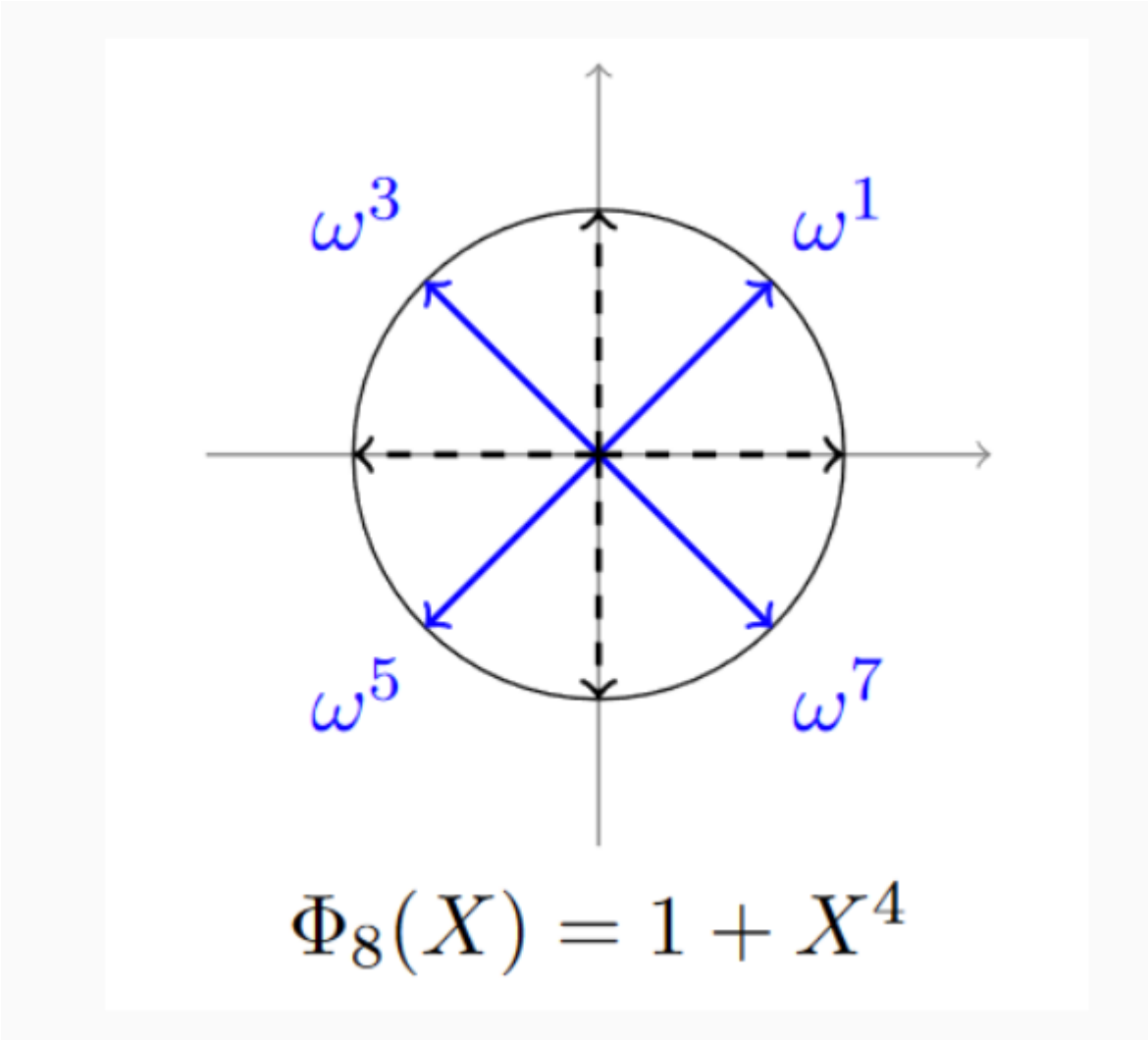
Blog post part 2

In the previous post they covered and implemented an encoder which encoded complex vectors as polynomials with complex coefficients. Now we want to extend this so we encode our complex vectors as polynomials with integer coefficients only.

We want to define the map

- $\sigma : \mathbb{Z}[X]/(X^N + 1) \to \mathbb{C}^N$
- $\sigma^{-1} : \mathbb{C}^N \to \mathbb{Z}[X]/(X^N + 1)$

A key idea in this part is that half of the complex roots of a cyclotomic polynomial are complex conjugates of the other half. We can see this in the figure below as the roots are symmetrical reflected about the y axis, meaning the sign of i is simply flipped.



Take some polynomial $P(x)$, and 2 complex conjugate numbers $r_1 = \alpha + i\beta$, and $r_2 = \alpha - i\beta$. Then $P(r_1) = \overline{P(r_2)}$.

Example:

- $2 - i$,$2 + i$
- $P(x) = x^2 + 1$
- $P(2 - i) = (2 - i)^2 + 1$
- $P(2 - i) = 4 - 4i + i^2 + 1$
- $P(2 - i) = 4 - 4i$
- $P(2 + i) = (2 + i)^2 + 1$
- $P(2 + i) = 4 + 4i + i^2 + 1$
- $P(2 + i) = 4 + 4i$

In words evaluating a polynomial at 2 complex conjugates produces 2 complex conjugates. CKKKS takes advantage of this fact to achieve the desired mapping.

> Because polynomials in $\mathcal{R}$ have integer coefficients, i.e. real coefficients, and we evaluate them on complex roots, where half are the conjugates of the other (see the previous figure), we have that: $\sigma(\mathcal{R}) \subseteq \mathbb{H} = z \in \mathbb{C}^N : z_j = \overline{z_{-j}}$.

This is pretty much stated here

> We can see on this picture that $\omega^1 = \overline{\omega^7}$ and $\omega^3 = \overline{\omega^5}$. In general, because we evaluate a real polynomial on the roots of $X^N + 1$, we will also have that for any polynomial $m(X) \in \mathcal{R}, m(\xi^j) = \overline{m(\xi^{-j})} = m(\overline{\xi^{-j}})$.

Image of $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ under σ is the set of vectors that σ maps one of the polynomials with integer coefficients in $\mathcal{R}$ to, denoted by $\sigma(\mathcal{R})$.

Because of what we've stated previously about evaluating polynomials at complex conjugates $\sigma(\mathcal{R})$ is necessarily a subset of the set of vectors of complex conjugates denoted by

$$\mathbb{H} = \{z \in \mathbb{C}^N : z_j = \overline{z_{-j}}\}$$

I believe the $-j$ here is meant to denote $M - j$, so we can add the correction

$$\sigma(\mathcal{R}) \subset \mathbb{H} = \{z \in \mathbb{C}^N : z_j = \overline{z_{M-j}}\}$$

This means we can work with plaintext messages in $\mathbb{C}^{N/2}$ and copy the coordinates doubling the size of the vector, to a vec in $\mathbb{C}^N$. This gives us a possibility of mapping the vector to a polynomial of integer coordinates, but it doesn't guarantee it will be in $\sigma(\mathcal{R}))$ is a proper subset of $\mathbb{H}$. In other words all integer polynomials map to vectors of complex conjugates, but not all vectors of complex conjugates map to integer polynomials.

This is a problem that CKKKS solves in a rather complicated (to me anyways) way.

First we can recognize the $\mathcal{R}$ is a essentially a vector space of polynomials. So it must have an orthogonal basis. Assuming we can find one using gram-schmidt or something. This basis must map to a basis in in $\sigma(\mathcal{R})$ since the 2 spaces are isomorphic.

A point of confusion is from what they've said here I would assume this basis $\beta \subset \sigma(\mathcal{R})$ generates only those vectors in $\sigma(\mathcal{R})$ however due to some rounding later it seems like this is not the case.

Anyways for any $z \in \mathbb{H}$ you want to find the closest vector to z contained in $\sigma(\mathcal{R})$. This is the projection of any z to the basis β. We form this project onto the subspace formed by β by essentially summing the norm of the inner product of z and each basis vector $b \in \beta$.

This is the confusing part

> Finally, once we have the coordinates z_i, we simply need to round them randomly, to the higher or the lower closest integer, using the "coordinate-wise random rounding". This way we will have a polynomial which will have integer coordinates in the basis $(\sigma(1), \sigma(X), \ldots, \sigma(X^{N-1}))$, therefore this polynomial will belong to $\sigma(\mathcal{R})$ and the trick is done.

I would assume once we've projected z onto β we would already have a vector that maps back to a polynomial in $\mathcal{R}$, but thats not the case. In the paper they mention that the basis β defines an ideal lattice that is $\sigma(\mathcal{R})$. So I suppose we are in fact computing the closest lattice vector to z at the end of the day and working with that.

In anycase at this point we have a way to map any $z \in \mathbb{H}$ to a vector $z' \in \sigma(\mathcal{R})$. This allows us to use σ^{-1} to get back a polynomial with integer coordinates. There is also some scaling factor Δ thats used to keep a desired level of precision in the computations. But we have successfully

defined the mapping from arbitrary vectors of complex numbers to polynomials with integer coefficients.

Summary of the encoding process

- take an element of $z \in \mathbb{C}^{N/2}$
- expand it to $\pi^{-1}(z) \in \mathbb{H}$
- multiply it by Δ for precision
- project it on $\sigma(\mathcal{R})$: $\lfloor \Delta . \pi^{-1}(z) \rceil_{\sigma(\mathcal{R})} \in \sigma(\mathcal{R})$
- encode it using σ : $m(X) = \sigma^{-1}(\lfloor \Delta . \pi^{-1}(z) \rceil_{\sigma(\mathbb{R})}) \in \mathcal{R}$.

They did not explicitly mention what π does but from the code it is literally the opposite of π^{-1}. It simply cuts the vector in half to take us from a vector in $\mathbb{H}$ to a vector in $\mathbb{C}^{N/2}$.

Summary of part 2

So in reality CKKKS works in the vector $\mathbb{C}^{N/2}$ and projects into the space $\mathbb{H} \subset \mathbb{C}^N$. The operation that goes between these 2 is the isomorphism $\pi(z), \pi^{-1}(z)$. Ultimately we want to work in the ring of polynomials with integer coefficients $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$. So want to work with the map

$$\sigma : \mathcal{R} \to \sigma(\mathcal{R}) \subset \mathbb{H}$$

To do this we take our vector $z \in \mathbb{H}$ we obtained using $\pi^{-1}(z)$ and compute its closest vector in $\sigma(\mathcal{R})$. Then we compute $\sigma^{-1}(z) \in \mathcal{R}$ to obtain our polynomial which encodes our plaintext. Then we can work this polynomial homomorphically and eventually use $\pi \cdot \sigma(z)$ to recover our plaintext. So really we are working with a couple isomorphisms and spaces already. Fairly complicated.