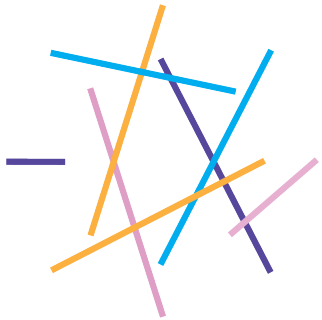
How To Build A Computer Out Of Polynomials

Holden Mui

The Future Of (Super) Computing

Oct 2025



Background

Traditional computers are insecure: if someone has physical access, then they can read the data being processed.

Question

Can we build a computer so that an observer with full system access still learns nothing about the data being processed?

Idea:

- Data will be encrypted
- Secret key needed to decrypt ciphertexts
- Computations happen on encrypted data

But how does one compute on encrypted data?

Computing on Encrypted Data

Question

Is there a way to encrypt real numbers such that

- one can "sum" an encryption of a and an encryption of b to get an encryption of $a + b$, and
- one can "multiply" an encryption of a and an encryption of b to get an encryption of ab?

Answer: Yes... with polynomials in a *polynomial ring*!

- Ciphertexts will be pairs of polynomials
- Secret key will be a polynomial

With this technology, we can build our Super Computer because

- AND gates are products: $AND(b_1, b_2) = b_1 b_2$
- NOT gates are additions and products: $NOT(b) = 1 + (-1) \cdot b$
- All computation can be built with AND and NOT gates

Polynomial Rings

Ciphertexts: pairs of polynomials in $\mathbb{Z}[X]/(X^{65536}+1)$.

Question

What is $\mathbb{Z}[X]/(X^{65536}+1)$?

- Fancy notation for "set of polynomials with degree less than 65536"
- Addition happens normally
- For multiplication, replace all X^{65536}'s with -1's after multiplying.

Example

In the ring $\mathbb{Z}[X]/(X^{65536}+1)$,

$$\begin{aligned}(X^{30000}+2X)(X^{40000}+3) &= X^{70000}+2X^{40001}+3X^{30000}+6X \\ &= X^{65536}\cdot X^{4464}+2X^{40001}+3X^{30000}+6X \\ &= -X^{4464}+2X^{40001}+3X^{30000}+6X.\end{aligned}$$

Secret Key

The secret key is a random polynomial in $\mathbb{Z}[X]/(X^{65536}+1)$ with small coefficients.

Example

$$s(X) = 3 + 2X - X^2 + \cdots + 2X^{65535} \in \mathbb{Z}[X]/(X^{65536}+1).$$

Only the owner of the Super Computer knows the secret key. The secret key is not stored on the Super Computer.

Encryption and Decryption

To encrypt real number m, generate random $C_1(X) \in \mathbb{Z}[X]/(X^{65536}+1)$ modulo 10^{300} and compute

$$C_2(X) = s(X)C_1(X) + \lfloor 1000000m \rfloor + \text{error} \pmod{10^{300}}.$$

The ciphertext is $(C_1(X), C_2(X))$.

To decrypt ciphertext $(C_1(X), C_2(X))$ using secret key $s(X)$, compute

$$\begin{aligned} C_2(X) - s(X)C_1(X) &\approx s(X)C_1(X) + \lfloor 1000000m \rfloor - s(X)C_1(X) \\ &\approx \lfloor 1000000m \rfloor \pmod{10^{300}}. \end{aligned}$$

Then divide by 1000000 to recover m.

Security

Recall a ciphertext $(C_1(X), C_2(X))$ of m is generated via

$$C_1(X) = \text{random polynomial in } \mathbb{Z}[X]/(X^{65536}+1) \pmod{10^{300}}$$
$$C_2(X) = s(X)C_1(X) + \lfloor 1000000m \rfloor + \text{error} \pmod{10^{300}}.$$

Question

Can one deduce m from the ciphertext without knowing the secret key?

No, since the *Ring Learning With Errors assumption* says the distribution of $(C_1(X), C_2(X))$ is indistinguishable from random.

Adding Ciphertexts

Question

Suppose $(C_1(X), C_2(X))$ is a ciphertext for m and $(D_1(X), D_2(X))$ is a ciphertext for n. How does one get a ciphertext for $m+n$?

Answer: $(C_1(X) + D_1(X), C_2(X) + D_2(X))$ is a ciphertext for $m+n$!

Proof.

Decrypting $(C_1(X) + D_1(X), C_2(X) + D_2(X))$ gives

$$\begin{aligned}(C_2(X) + D_2(X)) - s(X)(C_1(X) + D_1(X)) \\ &= (C_2(X) - s(X)C_1(X)) + (D_2(X) - s(X)D_1(X)) \\ &\approx \lfloor 1000000m \rfloor + \lfloor 1000000n \rfloor \\ &\approx 1000000(m+n).\end{aligned}$$

Dividing by 1000000 gives $m+n$. □

Multiplying Ciphertexts

To multiply ciphertexts, the Super Computer owner must first generate a *relinearization key* from the secret key.

This is done by randomly generating $\ell_1(X) \in \mathbb{Z}[X]/(X^{65536}+1)$ modulo 10^{550} and computing

$$\ell_2(X) = s(X)\ell_1(X) + 10^{250}s(X)^2 + \text{error} \pmod{10^{550}}$$

The relinearization key is $(\ell_1(X), \ell_2(X))$. It is generated once and stored on the Super Computer.

Question

Can one deduce the secret key from the relinearization key?

No, by the Ring Learning With Errors assumption. So it is safe to store the relinearization key on the Super Computer.

Multiplying Ciphertexts

Question

Suppose (C_1, C_2) is a ciphertext for m and (D_1, D_2) is a ciphertext for n. Using relinearization key (ℓ_1, ℓ_2), how does one get a ciphertext for mn?

Answer: $\left(\left\lfloor \frac{C_1 D_2 + C_2 D_1 + C_1 D_1 \ell_1 / 10^{250}}{1000000} \right\rfloor, \left\lfloor \frac{C_2 D_2 + C_1 D_1 \ell_2 / 10^{250}}{1000000} \right\rfloor \right)$ modulo 10^{294}!

Proof.

Decrypting gives

$$\begin{aligned}
&\left\lfloor \frac{C_2 D_2 + C_1 D_1 \ell_2 / 10^{250}}{1000000} \right\rfloor - s \left\lfloor \frac{C_1 D_2 + C_2 D_1 + C_1 D_1 \ell_1 / 10^{250}}{1000000} \right\rfloor \\
&\quad \approx \frac{C_2 D_2 + C_1 D_1 (s\ell_1 + 10^{250} s^2)/10^{250} - sC_1 D_2 - sC_2 D_1 - sC_1 D_1 \ell_1 / 10^{250}}{1000000} \\
&\quad = \frac{C_2 D_2 - sC_1 D_2 - sC_2 D_1 + C_1 D_1 s^2}{1000000} \\
&\quad = \frac{1}{1000000} (C_1 - sC_2)(D_1 - sD_2) \approx 1000000 mn.
\end{aligned}$$

Dividing by 10^6 gives mn. □

Multiplying Ciphertexts

We built addition and multiplication, so did we finish building our Super Computer? Not quite...

- Multiplying ciphertexts with modulus 10^{300} gives ciphertexts with modulus 10^{294}.
- More generally, multiplying ciphertexts with modulus 10^{ℓ} gives ciphertext with modulus $10^{\ell-6}$...
- ...so at most 50 consecutive multiplications allowed.

Question

Can we fix this so that our computer can compute *any* arithmetic circuit, regardless of multiplicative depth?

Answer: yes!

Bootstrapping

Bootstrapping turns ciphertexts modulo 10^{12} into ciphertexts with larger modulus. If (C_1, C_2) is a ciphertext modulo 10^{12} encrypting m, then

$$C_2(X) - s(X)C_1(X) \approx \lfloor 1000000m \rfloor \pmod{10^{12}}.$$

Reinterpreting (C_1, C_2) modulo 10^{300} gives

$$C_2(X) - s(X)C_1(X) \approx \lfloor 1000000m \rfloor + 10^{12}t(X) \pmod{10^{300}}$$

for some polynomial $t(X)$. It has small coefficients because the secret key has small coefficients. Two steps:

- Step 1: turn ciphertext for polynomial $p(X)$ into ciphertext for its constant term
- Step 2: turn ciphertext for $\lfloor 1000000m \rfloor + 10^{12}t_0$ for unknown (but small) t_0 into ciphertext for $\lfloor 1000000m \rfloor$

Bootstrapping (Step 1)

Goal: turn ciphertext for $p(X) \in \mathbb{Z}[X]/(X^{65536}+1)$ into ciphertext for its constant term.

Observation

Suffices to get ciphertexts of $p(X^j)$ for any j.

Proof.

Let $p(X) = p_0 + p_1X + p_2X^2 + \cdots + p_{65535}X^{65535}$. Then

$$\begin{aligned} P(X) + P(X^3) + P(X^5) + \cdots + P(X^{131071}) \\ &= 65536p_0 + p_1(X + X^3 + X^5 + \cdots + X^{131071}) + \\ &\quad + p_2(X^2 + X^6 + X^{10} + \cdots + X^{262142}) + \dots \\ &= 65536p_0. \end{aligned}$$

So summing ciphertexts for $p(X), p(X^3), p(X^5), \dots, P(X^{131071})$ and multiplying by ciphertext for $\frac{1}{65536}$ gives ciphertext for p_0. □

Bootstrapping (Step 1)

Question

How to turn ciphertext for $p(X)$ into ciphertext for $p(X^j)$ for any j?

Answer: use *rotation key* $(r_1(X), r_2(X))$, generated by randomly choosing $r_1(X)$ modulo 10^{550} and computing

$$r_2(X) \approx s(X)r_1(X) - 10^{250}s(X^j) + \text{error} \pmod{10^{550}}.$$

Then $\left(\lfloor \frac{r_1(X)}{10^{250}} C_1(X^j) \rfloor, C_2(X^j) + \lfloor \frac{r_2(X)}{10^{250}} C_1(X^j) \rfloor\right)$ is a ciphertext for $p(X^j)$, given $(C_1(X), C_2(X))$ is a ciphertext for $p(X)$.

Proof.

That ciphertexts decrypts to

$$\begin{aligned} &\approx C_2(X^j) + \tfrac{s(X)r_1(X) - 10^{250}s(X^j)}{10^{250}} C_1(X^j) - s(X)\tfrac{r_1(X)}{10^{250}} C_1(X^j) \\ &= C_2(X^j) - s(X^j)C_1(X^j) = p(X^j). \end{aligned}$$

□

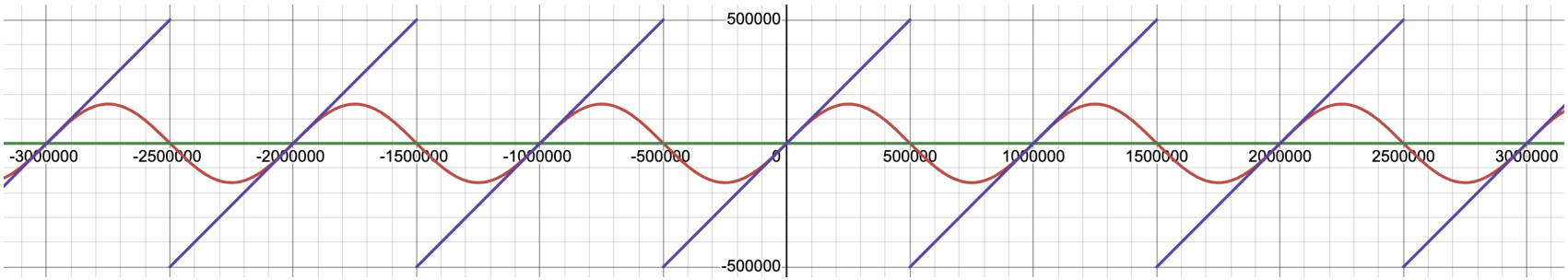
Bootstrapping (Step 2)

Goal: turn ciphertext for $\lfloor 1000000m \rceil + 10^{12}t_0$ for unknown (but small) t_0 into ciphertext for $\lfloor 1000000m \rceil$.

Idea

Evaluate the "modulo 10^{12}" function on the ciphertext.

Issue: we can only add ciphertexts and multiply ciphertexts.



Fix: approximate the scaled sine function with period 10^{12} using its Taylor series. Then evaluate the Taylor series on the ciphertext.

Recap

We have built the Super Computer! Our Super Computer can:

- add ciphertexts
- multiply ciphertexts (which turns ciphertexts with modulus 10^{ℓ} into a ciphertext with modulus $10^{\ell-6}$)
- bootstrap ciphertexts, which turns ciphertexts with small modulus into ciphertexts with large modulus

This gives us *cryptographically enhanced computing*.

Question

Is this computer performant?

As defined in this talk, no. But optimizations give a state-of-the-art complex number multiplication time of 2.6 milliseconds (per 16384 multiplications), and a bootstrapping time of 15 milliseconds.

Extensions

- SIMD operations: ciphertexts can encrypt 32768 complex numbers
- Public-key encryption: anyone can encrypt
- Multiparty decryption: ciphertexts can only be decrypted if enough people agree

Thank you!

Questions?