Homomorphic Encryption for Large Integers from Nested Residue Number Systems

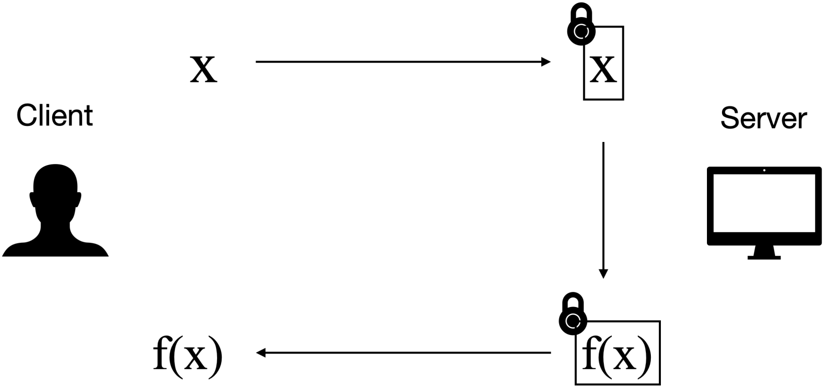
Dan Boneh and Jaehyung Kim

Stanford University

The Big Picture

- In some applications of fully homomorphic encryption (FHE), we need computations over a **prescribed large modulus**.
- We design a dedicated **FHE scheme** by introducing a **nested CRT** structure inside RLWE.

Background: Fully Homomorphic Encryption (FHE)



Example Application: Homomorphic Signing

In the following cases, we may need large (prescribed) modulus:

- **Universal Thresholdizer** [BGG+18]:

$$\forall \text{ signature} \xrightarrow{\text{Threshold FHE}} \text{one-round threshold signature}$$

- **Universal Blinder**:

$$\forall \text{ signature} \xrightarrow{\text{Verifiable FHE}} \text{one-round blind signature}$$

When thresholdizing/blinding well known signature schemes like ECDSA and Schnorr, one needs arithmetic over some large elliptic curve primes (e.g. 256 or 384 bit).

Which FHE scheme to choose?

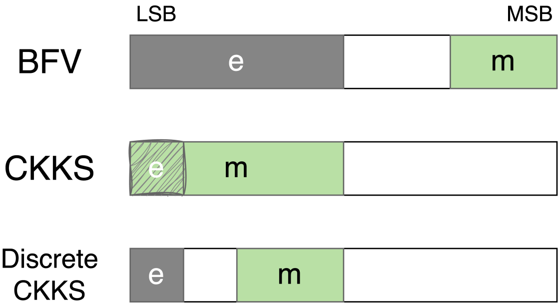
	SIMD	Plaintext Space
BGV/BFV	✓	$\mathbb{Z}_p$
CGGI/DM	✗	$\{0, 1\}$
CKKS	✓	$\mathbb{C}$

Problem of BGV/BFV

The noise growth is proportional the plaintext modulus p.[a]

[a]One may consider using the generalized BFV [GV25, CHM+25]. They only support cyclotomic moduli, not arbitrary moduli.

Discrete CKKS[1]



[1] [DMPS24, CKKL24, BC**K**S24, B**K**SS24, AKP25, **K**N25]

Discrete CKKS

Supports the following homomorphic operations:

1. **Arithmetic Operations** [DMPS24]: $+$ and $\times$ over $\mathbb{Z}$.
2. **Look-up Table** [B**K**SS24, AKP25]: Any function $f : \mathbb{Z}_t \to \mathbb{Z}_t$
3. **Modular Reduction** [**K**N25]: $[\cdot]_t : \mathbb{Z} \to \mathbb{Z}_t$.

Our Construction: Ingredients

A **homomorphic computer** with $+$, $\times$, and $[\cdot]_t$ over $\mathbb{Z}$ and $\mathbb{R}$. The computer is equipped with SIMD for a large dimension n (e.g. 2^{15}).

7	3	-3		$\cdots$		-2	8	-1

$$\times$$

1	-2	6		$\cdots$		4	-9	3

The computer only supports small integers (e.g. up to 8 bits).

Step 1: Asymmetric Modular Reduction

[**K**N25] evaluates a polynomial interpolation to modular reduce. We evaluate different polynomials for each slot to allow different modular reductions across the slots.

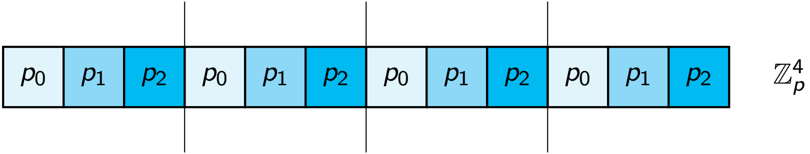
10	7	4		$\cdots$		-1	-3	3
%7	%5	%3		$\cdots$		%11	%13	%17
3	2	1		$\cdots$		10	10	3

Key Idea

Leverage CRT to store a large integer within a single ciphertext.

First Layer CRT Encoding

In the slots, we assign moduli as follows:



The first layer CRT system, where $(n, k) = (12, 3)$.

In particular, a ciphertext can store n/k integers of modulus $p = \prod_{i=0}^{k} p_i$.

Checklist

- ✓ Homomorphic $\mathbb{Z}_p$ computer for smooth $p = \prod_i p_i$.
- ✓ Homomorphic $\mathbb{Z}_{p_i}$ computer $(0 \le i < k)$.

Step 2: Homomorphic Base Conversion

To support a modular reduction by an arbitrary integer $r \gg \max_i(p_i)$, we rely on the fast base conversion from [HPS19].

[HPS19] converts an integer x represented under CRT moduli $\{p_i\}_{0 \le i < k}$ to a modulo r representation as follows:

$$[x]_r = \left[\sum_{i=0}^{k-1} y_i \cdot [\hat{p}_i]_r - v \cdot [p]_r \right]_r$$

where

$$\begin{aligned} p &:= \textstyle\prod_{i=0}^{k-1} p_i, \quad \hat{p}_i := p/p_i \\ y_i &:= \left[[x]_{p_i} \cdot \hat{p}_i^{-1} \right]_{p_i} \\ v &:= \lfloor \textstyle\sum_{i=0}^{k-1} y_i / p_i \rceil \end{aligned} .$$

Step 2: Homomorphic Base Conversion

As the last $[\cdot]_r$ cannot be evaluated easily, we instead compute

$$\sum_{i=0}^{k-1} y_i \cdot [\hat{p}_i]_r - v \cdot [p]_r = [x]_r + re$$

for some small e. Since we cannot directly store this big integer, we keep our CRT representation. In terms of modulo p_i computation, we compute

$$\sum_{j=0}^{k-1} y_j \cdot [[\hat{p}_j]_r]_{p_i} - v \cdot [[p]_r]_{p_i}.$$

Step 2: Homomorphic Base Conversion

$$\sum_{j=0}^{k-1} y_j \cdot [[\hat{p}_j]_r]_{p_i} - v \cdot [[p]_r]_{p_i}.$$

This can be written as[2]

1. Arithmetic over p_i $(0 \leq i < k)$.
2. Real number computation (to compute $\sum_{i=0}^{k-1} y_i/p_i$)
3. Rounding (to compute v)

Interestingly, the rounding is **free** due to the nature of discrete CKKS.

Checklist

✓ Homomorphic $\mathbb{Z}_r$ computer $(r < \sqrt{p})$.

[2] Recall that $v = \lfloor \sum_{i=0}^{k-1} y_i/p_i \rceil$.

Problem: Not enough small primes

Now we have modulo r arithmetic for a large integer r.
This seems to solve our initial goal, but...

Not enough small primes

The CRT moduli p_i for $0 \leq i < k$ need to be coprime to each other.
However, there is only a limited number of mutually coprime moduli.

For instance, there are 31 primes less than 128 which can represent at most $2^7 \times 3^4 \times 5^3 \times \cdots < (2^7)^{31} = 2^{217}$.

Step 3: Second Layer CRT Encoding

We may use different r across the $(\mathbb{Z}_p)$ slots, providing a second layer CRT system. Suppose that we use $r_0, r_1, \ldots, r_{\ell-1}$.

r_0			r_1			r_0			r_1			$\mathbb{Z}_r^2$
p_0	p_1	p_2	p_0	p_1	p_2	p_0	p_1	p_2	p_0	p_1	p_2	$\mathbb{Z}_p^4$

The second layer CRT system, where $(n, k, \ell) = (12, 3, 2)$.

Then we have $\frac{n}{k\ell}$ many $\mathbb{Z}_r$ slots where $r = \prod_{i=0}^{\ell-1} r_i$.

Checklist

- ✓ Homomorphic $\mathbb{Z}_r$ computer ($r = \prod_i r_i$).
- ✓ Homomorphic $\mathbb{Z}_{r_i}$ computer ($0 \leq i < \ell$).

Step 4: Second Layer Base Conversion

To take a larger modular reduction by $s \in \mathbb{Z}_{>0}$, we simulate the homomorphic base conversion in Step 2. To do this, we need

- **Arithmetic Operations over $\mathbb{Z}_{r_i}$**: ✓

Problems

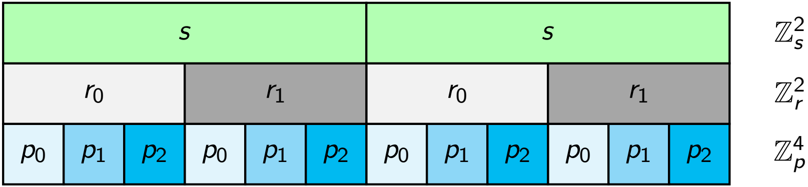
- **Real Number Arithmetic**: We no longer have a baseline homomorphic real number computer.
- **Rounding**: We no longer can rely on the nature of discrete CKKS.

$\Rightarrow$ we refer to our paper for details.

Checklist

✓ Homomorphic $\mathbb{Z}_s$ computer ($s < \sqrt{r}$, $r = \prod_i r_i$).

Summary



A visualization of the nested CRT system.

Observe that

$$\log s \approx \frac{1}{2}\sum_j \log r_j \approx \frac{1}{4}\sum_j \sum_i \log p_i \leq \frac{n}{4}\log t = O(n)$$

where t is the maximum plaintext modulus that supports modular reduction from [**K**N25].

Experiments

All experiments in single threaded CPU (Apple M4 Max), satisfying 128 bits of security according to [BTPH22].

$\log(r)$	$\#$ slots	$\mathbb{Z}_r$ mult time	
		latency	amortized time
960	32	18.3 sec	572 ms
7679	4	18.4 sec	4.60 sec

Smooth ($\mathbb{Z}_r$) Modular Multiplication.

$\log(s)$	$\#$ slots	$\mathbb{Z}_s$ mult time	
		latency	amortized time
255	32	150 sec	4.67 sec
384	32	149 sec	4.66 sec
2048	4	190 sec	47.5 sec

Arbitrary ($\mathbb{Z}_s \subset \mathbb{Z}_r$) Modular Multiplication.

Experiments

	$\log(t)$	# slots	latency	throughput
TFHE-rs [Zam22]	128	1	101 sec	101 sec
	256		403 sec	403 sec
This paper	128	256	18.3 sec	0.0715 sec
	256	128	18.3 sec	0.143 sec

Comparison with the state-of-the-art integer (bootstrapped) multiplications. Here t denotes the plaintext modulus.

Takeaways

- Instead of directly supporting a large modulus, we show how to build a large integer computer from small integer computers via CRT.
- Sacrificing the number of slots gives you better latency.
  - Q: Is there an analogue in BGV/BFV?
A: The generalized BFV [GV25, CHM+25], for cyclotomic rings.
  - Q: Can we do better for power-of-two?
A: Use partial DFT encoding [**Kim**25]

Thank you!

ePrint 2025/346

jaehyungkim0/CRT-FHE

Bibliography I

A. Alexandru, A. Kim, and Y. Polyakov.
General functional bootstrapping using CKKS.
In *CRYPTO*, 2025.

Y. Bae, J. H. Cheon, J. **K**im, and D. Stehlé.
Bootstrapping bits with CKKS.
In *EUROCRYPT*, 2024.

D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. R. Rasmussen, and A. Sahai.
Threshold cryptosystems from threshold fully homomorphic encryption.
In *CRYPTO*, 2018.

Y. Bae, J. **K**im, D. Stehlé, and E. Suvanto.
Bootstrapping small integers with CKKS.
In *ASIACRYPT*, 2024.

Bibliography II

J.-P. Bossuat, J. Troncoso-Pastoriza, and J.-P. Hubaux.
Bootstrapping for approximate homomorphic encryption with negligible failure-probability by using sparse-secret encapsulation.
In *ACNS*, 2022.

H. Cha, I. Hwang, S. Min, J. Seo, and Y. Song.
MatriGear: Accelerating Authenticated Matrix Triple Generation with Scalable Prime Fields via Optimized HE Packing .
In *IEEE S&P*, 2025.

H. Chung, H. Kim, Y.-S. Kim, and Y. Lee.
Amortized large look-up table evaluation with multivariate polynomials for homomorphic encryption.
IACR eprint 2024/274, 2024.

Bibliography III

N. Drucker, G. Moshkowich, T. Pelleg, and H. Shaul.
BLEACH: Cleaning errors in discrete computations over CKKS.
J. Cryptol., 2024.

R. Geelen and F. Vercauteren.
Fully homomorphic encryption for cyclotomic prime moduli.
In *EUROCRYPT*, 2025.

S. Halevi, Y. Polyakov, and V. Shoup.
An improved rns variant of the bfv homomorphic encryption scheme.
In *CT-RSA*, 2019.

J. **Kim**.
Faster homomorphic integer computer.
Cryptology ePrint Archive, Paper 2025/1440, 2025.

Bibliography IV

J. **K**im and T. Noh.
Modular reduction in CKKS.
Communication in Cryptology, 2025.

Zama.
TFHE-rs: A Pure Rust Implementation of the TFHE Scheme for Boolean and Integer Arithmetics Over Encrypted Data, 2022.
https://github.com/zama-ai/tfhe-rs.