Revisiting Key Decomposition Techniques for FHE: Simpler, Faster and More Generic

M. G. Belorgey, S. Carpov, **N. Gama**, S. Guasch, D. Jetchev

FHE.org Conference - 24 Mars 2024

Outline

1. **External products:**
   The missing public-secret product.
2. **Auxiliary Gadgets and Bivariate representation**
   Gadget decomposition of the source of truth
   The return of base 2^K representation in large depth.

Part I - External products: Two halves make a whole

External products are omnipresent in FHE

Ring-LWE has never really been about Rings

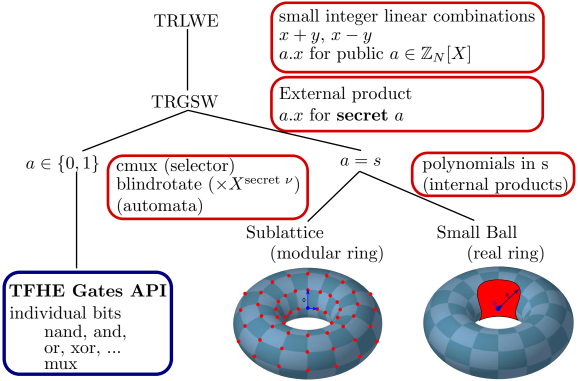
- Ring-LWE FHE was never really about rings.
- FHE arithmetic flows from **secret linear combinations**, a.k.a. **external products**:
  - with small integer polynomial coefficients
  - over some high precision space (Torus, or big integers)

$$\sum \underbrace{a_i}_{\text{small int}} \cdot \underbrace{c_i}_{\text{High Prec}}$$

.

External products are omnipresent in FHE

- Covers every "tools":
  keyswitches (2009), gadget decomposition, relinearization, products, automorphisms, bootstrappings (2024)
- Yet, proper formalization of external products appeared only from 2016.



External products are omnipresent in FHE

SANDBOXAQ™ inpher

Ring-LWE has never really been about Rings

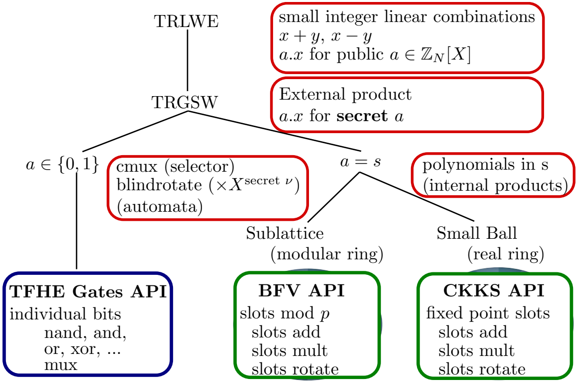
- Ring-LWE FHE was never really about rings.
- FHE arithmetic flows from **secret linear combinations**, a.k.a. **external products**:
  - with small integer polynomial coefficients
  - over some high precision space (Torus, or big integers)

$$\sum \underbrace{a_i}_{\text{small int}} \cdot \underbrace{c_i}_{\text{High Prec}}$$

.

External products are omnipresent in FHE

- Covers every "tools":
  keyswitches (2009), gadget decomposition, relinearization, products, automorphisms, bootstrappings (2024)
- Yet, proper formalization of external products appeared only from 2016.



External products vs. relinearization

	Ciphertext	Plaintext
TRLWE deg 1	(a,b)	b-S.a
TRLWE deg 1	(c,d)	d-S.c
TRLWE deg 2	(e,f,g)	g-S.f+S^2.e

BFV or CKKS products

Relinearization

Ext. product

RK=TRLWE$_S$(S^2)

C=TRGSW(S)

We just need half of the TRGSW

SANDBOXAQ inpher

Relinearization needs only half of the TRGSW material

- TRGSW was designed for **secret** × **secret** products
- Relinearization uses **secret** × **public** product

Half External Product (i.e. secret-public)

$$\begin{array}{r|ccccc} \text{Plaintext:} & \mathbb{Z}_N[X] & \cdot & \mathbb{T}_N[X] & \rightarrow & \mathbb{T}_N[X] \\ \text{Ciphertext:} & \mathtt{HTRGSW}(A) & ⊿ & b & \rightarrow & \mathsf{TRLWE}(A \cdot b) \\ \text{Noise:} & \varepsilon & & & \rightarrow & \approx N\varepsilon \end{array}$$

External product from [CGGI16]

$$\begin{array}{r|ccccc} \text{Plaintext:} & \mathbb{Z}_N[X] & \cdot & \mathbb{T}_N[X] & \rightarrow & \mathbb{T}_N[X] \\ \text{Ciphertext:} & \mathsf{TRGSW}(A) & \boxdot & \mathsf{TRLWE}(b) & \rightarrow & \mathsf{TRLWE}(A \cdot b) \\ \text{Noise:} & \varepsilon & & \alpha & \rightarrow & \approx \|A\|_2 \cdot \alpha \end{array}$$

Application of half TRGSW products

Relinearization application

- Relinearization:

$$(f, g) + \mathtt{HTRGSW}(S^2) \boxdot e$$

Original external product

- Original TRGSW:

$$\underbrace{\mathrm{TRGSW}(A)}_{C} = (\underbrace{\mathtt{HTRGSW}(A)}_{D}, \underbrace{\mathtt{HTRGSW}(SA)}_{E})$$

- 1 full = 2 halves:

$$C \boxdot (a, b) = E \boxdot a + D \boxdot b$$

Half TRGSW: It is not just syntax!

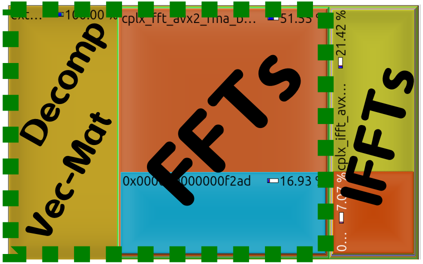
$$C \boxdot (a, b) = E \triangle a + D \triangle b$$

Approximate decomposition: Faster FHE

- Since b needs less precision than a, $D \triangle b$ is faster than $E \triangle a$
- expect 1 less FFT per external product.

Practical improvements

- 8 FFTs $\rightarrow$ 7 FFTs in the original TFHE lib.
- Possibility to improve also the Circuit bootstrapping (impacts TFHE-rs).



17% faster

overall: 15% faster

Half TRGSW: Improvements in practice!

SANDBOXAQ inpher

Table 2. Performance comparison of gate bootstrapping with a n2-standard GCP instance with 64GB of RAM and a 12-th Gen i7-1260p laptop with 64GB of RAM. All the benchmarks are single core.

Library	Instruction set	n2-standard	12-Gen i7-1260p
TFHE-lib, spqlios-fma	AVX2	22.4ms	10.4ms
TFHE-rs, `TFHE_LIB_PARAMETERS`	AVX2	18.2ms	8.6ms
	AVX512	14.4ms	*not supported*
TFHE-rs, `DEFAULT_PARAMETERS`	AVX2	14.4ms	7.6ms
	AVX512	13.7ms	*not supported*
Our work, halfTRGSW	AVX2	11.2ms	5.3ms

Part II - Auxiliary Gadgets: The return of base-2^K

First attempts on Chimera

SANDBOXAQ inpher

Idash 2018 - a pivoting moment for FHE

- Chimera: combine TFHE's blind-rotate with BFV-style arithmetic
- CKKS: first really efficient Full-RNS POC

External Product Formula

Ciphertext layer: $\text{prod} = \text{GadgetDec}(a, b) \times \text{Fixed\_GSW\_key}$

Arithmetic layer: $\underbrace{\text{vector<}\mathbb{T}_N[X]\text{>}}_{\text{Huge}} = \sum_{i=1}^{\text{level } \ell} \underbrace{\mathbb{Z}_N[X]}_{\text{small}} \cdot \underbrace{\text{vector<}\mathbb{T}_N[X]\text{>}}_{\text{Huge}}$

Chimera 2019-2020: First steps and failures

SANDBOXAQ inpher

$$\underset{\text{Huge}}{\text{vector}<\mathbb{T}_N[X]>} = \sum_{i=1}^{\text{level } \ell} \underset{\text{small}}{\mathbb{Z}_N[X]} \cdot \underset{\text{Huge}}{\text{vector}<\mathbb{T}_N[X]>}$$

$$\underset{\text{small}}{\mathbb{Z}_N[X]} \cdot \underset{\text{Huge}}{\mathbb{T}_N[X]}$$

Two big blockers

- FFT products: does not take advantage of the small coefficients.
- High precision base-2^K: carry propagation is prohibitive!

Main resolution: Give up, abort. (<2020) – Separation of concerns

- **Small depth FHE**: Use Approx-decomporition with FP.
- **Large depth FHE**: Use CRT + RNS.

Chimera 2019-2020: First steps and failures

$$\underset{\text{Huge}}{\text{vector}<\mathbb{T}_N[X]>} = \sum_{i=1}^{\text{level } \ell} \underset{\text{small}}{\mathbb{Z}_N[X]} \cdot \underset{\text{Huge}}{\text{vector}<\mathbb{T}_N[X]>}$$

$$\underset{\text{small}}{\mathbb{Z}_N[X]} \cdot \underset{\text{Huge}}{\mathbb{T}_N[X]}$$

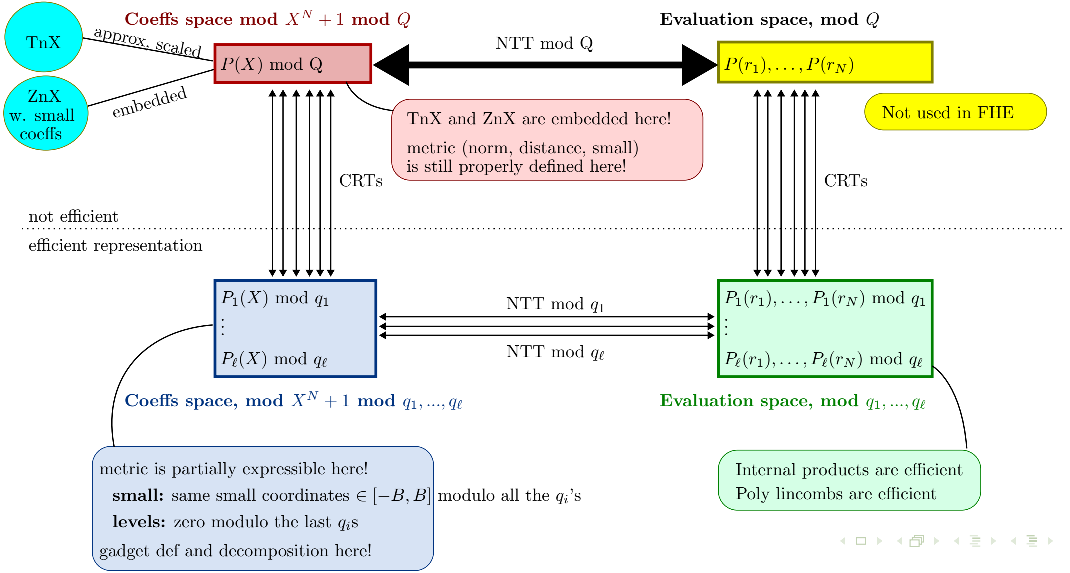
Two big blockers

- FFT products: does not take advantage of the small coefficients.
- High precision base-2^K: carry propagation is prohibitive!

Main resolution: Give up, abort. (<2020) – Separation of concerns

- **Small depth FHE**: Use Approx-decomporition with FP.
- **Large depth FHE**: Use CRT + RNS.

Full-RNS CRT-CKKS since 2018



[KLSS23]: Breaking news!! Full-RNS was not optimal!

SANDBOXAQ™ inpher

External Product Formula (CRT)

$$\text{prod} = \text{GadgetDec}(a, b)) \times \texttt{Fixed\_GSW\_key}$$

$$\begin{bmatrix} \mathbb{Z}_N[X] & \bmod q_1 \\ & \vdots \\ \mathbb{Z}_N[X] & \bmod q_\ell \end{bmatrix} = \sum_{i=1}^{\text{level } \ell} \underset{\text{small}}{\mathbb{Z}_N[X]} \cdot \begin{bmatrix} \mathbb{Z}_N[X] & \bmod q_1 \\ & \vdots \\ \mathbb{Z}_N[X] & \bmod q_\ell \end{bmatrix}$$

Observation by [KLSS23]

- Although the gadget decomposition is small, we still compute ℓ NTT's per polynomials (so ℓ^2 NTT's in total)
- This is the same as if the gadget decomposition was Huge.

The big idea (simplified scoop!)

- Forget/Delay the annoying mod $(q_1, \ldots, q_\ell)$, do the computation over $\mathbb{Z}$!!!

[KLSS23]: Breaking news!! Full-RNS was not optimal!

SANDBOXAQ inpher

External Product Formula (CRT)

$$\text{prod} = \text{GadgetDec}(a, b)) \times \texttt{Fixed\_GSW\_key}$$

$$\left(\begin{bmatrix} \mathbb{Z}_N[X] \cancel{\bmod q_1} \\ \vdots \\ \mathbb{Z}_N[X] \cancel{\bmod q_\ell} \end{bmatrix} = \sum_{i=1}^{\text{level } \ell} \underset{\text{small}}{\mathbb{Z}_N[X]} \cdot \begin{bmatrix} \mathbb{Z}_N[X] \cancel{\bmod q_1} \\ \vdots \\ \mathbb{Z}_N[X] \cancel{\bmod q_\ell} \end{bmatrix} \right) \quad \begin{matrix} \bmod q_1 \\ \vdots \\ \bmod q_\ell \end{matrix}$$

Observation by [KLSS23]

- Although the gadget decomposition is small, we still compute ℓ NTT's per polynomials (so ℓ^2 NTT's in total)
- This is the same as if the gadget decomposition was Huge.

The big idea (simplified scoop!)

- Forget/Delay the annoying mod $(q_1, \ldots, q_\ell)$, do the computation over $\mathbb{Z}$!!!

Chimera 2023: The return of base-2^K

SANDBOXAQ™ inpher

[BCGGJ23] Extends the scope of [KLSS23])

- If reductions mod $(q_1, \ldots, q_\ell)$ can be delayed and amortized.
- Carry propagation too!!!

When we consider base-2^k limbs:

$$\underset{\text{Huge}}{\mathbb{T}_N[X]} = \underset{\text{small}}{\mathbb{Z}_N[X]} \cdot \frac{1}{2^K} + \underset{\text{small}}{\mathbb{Z}_N[X]} \cdot \frac{1}{2^{2K}} + \cdots + \underset{\text{small}}{\mathbb{Z}_N[X]} \cdot \frac{1}{2^{\ell K}}$$

Linear combinations can be done per limb:

$$\begin{bmatrix} \underset{\text{bounded}}{\mathbb{Z}_N[X]} \\ \underset{\text{bounded}}{\mathbb{Z}_N[X]} \\ \vdots \\ \underset{\text{bounded}}{\mathbb{Z}_N[X]} \end{bmatrix} = \sum_{i=1}^{\text{level}\ell} \underset{\text{small}}{\mathbb{Z}_N[X]} \cdot \begin{bmatrix} \underset{\text{small}}{\mathbb{Z}_N[X]} \\ \underset{\text{small}}{\mathbb{Z}_N[X]} \\ \vdots \\ \underset{\text{small}}{\mathbb{Z}_N[X]} \end{bmatrix}$$

and carry propagation is delayed until the very end! (small DFTs, bounded integers!)

Chimera 2023: Bivariate representation

SANDBOXAQ inpher

Instead of:

$$\underbrace{\mathbb{T}_N[X]}_{\text{Huge}} = \underbrace{\mathbb{Z}_N[X]}_{\text{small}} \cdot \frac{1}{2^K} + \underbrace{\mathbb{Z}_N[X]}_{\text{small}} \cdot \frac{1}{2^{2K}} + \cdots + \underbrace{\mathbb{Z}_N[X]}_{\text{small}} \cdot \frac{1}{2^{\ell K}}$$

Write:

$$\underbrace{\mathbb{T}_N[X]}_{\text{Huge}} = \underbrace{\mathbb{Z}_N[X]}_{\text{small}} \cdot Y + \underbrace{\mathbb{Z}_N[X]}_{\text{small}} \cdot Y^2 + \cdots + \underbrace{\mathbb{Z}_N[X]}_{\text{small}} \cdot Y^\ell$$

Bivariate representation

- Carry propagation decoupling formalized by the presence of a variable Y. – hence "Bivariate" representation
- Multiplication of $\mathbb{Z}(X, Y)$ do make sense (genuine morphism between lifts over $\mathbb{R}[X]$) – leveraged in BFV and CKKS internal products.
- Fast arithmetic possible product via DFT over X and over Y !!!

Chimera 2023: Pros and Cons of the bivariate representation

SANDBOXAQ™ inpher

Wins

- External products (half and full) are fastest on base-2^K (optimal)
- Compatible with automorphisms and internal products (optimal)
- Prefix property: modulus rescaling is free (optimal)
- CKKS Noise levels: continuous! no gap, no artificial rescaling (optimal)
- Parametrization: every parameters flows from the noise level, no mysterious additional moduli (optimal)

Minor Setback

- CKKS and BFV: dominant terms improve, negligible terms are a bit larger. (good enough!)

Chimera 2023: Key takeaways 1

Cyclotomic Arithmetic (Backend)

Big-Number (Frontend)

	NTT	FFT
CRT	RNS-BFV RNS-CKKS (large depth)	🚀 Also Valid!
Bivariate (Base-2^K)	🚀 Also Valid!	TFHE (low depth) 🚀 large depth!

Frontend vs. Backend

- The big number mechanism and the cyclotomic mechanism must be decoupled.
- All combinations are as efficient (up to a factor 2)
- Switching between representations is easy! (i.e. external product to switch between representations)

SPQlios-arithmetic: a middle-ground arithmetic API for FHE

SANDBOXAQ™ inpher

Aritmetic over vector/matrices of small integer polynomials. – 3 key operations

1. **dft/idft**: via an NTT or FFT backend (whichever is faster)
2. **vmp_prepare/apply**: vector $\times$ preprocessed matrix
3. **cnv_prepare/apply**: (precomputed) vector $\times$ vector convolution

It is omnipotent!

- Can all **CRT and bivariate** frontends **at any depth**!
  - BlindRotate (CGGI bootstrapping in 6ms).
  - CKKS and BFV products (depth 30 in 0.3s)
  - Keyswitches and Automorphisms (depth 30 in 0.2s)
- Easier and sufficient **target for hardware developers**!

SPQlios-arithmetic: a middle-ground arithmetic API for FHE

SANDBOXAQ™ inpher

Aritmetic over vector/matrices of small integer polynomials. – 3 key operations

1. **dft/idft**: via an NTT or FFT backend (whichever is faster)
2. **vmp_prepare/apply**: vector $\times$ preprocessed matrix
3. **cnv_prepare/apply**: (precomputed) vector $\times$ vector convolution

It is omnipotent!

- Can all **CRT and bivariate** frontends **at any depth**!
    - BlindRotate (CGGI bootstrapping in 6ms).
    - CKKS and BFV products (depth 30 in 0.3s)
    - Keyswitches and Automorphisms (depth 30 in 0.2s)
- Easier and sufficient **target for hardware developers**!

Some benchmarks

SANDBOXAQ™ inpher

Operation	Keyswitch		Automorphism	CKKS product
Size	$N = 64k$ L=1729	$N = 32k$ L=865	$N = 64k$ L=1729	$N = 64k$ L=1729
Hardware	n2-standard VM Xeon(R) CPU @ 2.8GHz, 64GB RAM			
- Full-RNS (best r)	3.111s	0.359s	3.279s	3.311s
- [26] (best r)	0.965s	0.161s	1.134s	1.155s
- ours: biv + fft-f64 ($K = 19$)	0.589s	0.086s	0.602s	0.862s
- ours: biv + ntt120 ($K = 52$)	0.541s	0.073s	0.547s	0.777s
Hardware	Laptop with Intel Core i7-1260P @ 4.7GHz, 64GB RAM			
- Full-RNS (best r)	1.598s	0.192s	1.796s	1.759s
- [26] (best r)	0.521s	0.085s	0.578s	0.598s
- ours: biv + fft-f64 ($K = 19$)	0.228s	0.027s	0.233s	0.335s
- ours: biv + ntt120 ($K = 52$)	0.218s	0.029s	0.221s	0.314s

Conclusions and Key Takeways

SANDBOXAQ™ inpher

Key Takeway for hardware developers

- FFT or NTT are both valid and will provide efficient FHE in all levels: just pick one.
- Aim for native hardware support of the largest possible word size.
  64-bit arithmetic is good, 128-bit would be much better!

Key Takeway for FHE libraries developers

- Use gadget decompositions as primary software $\leftrightarrow$ hardware API.
- Do not expose parameters that the end-user won't be able to set!

Key Takeway for FHE compilers/transpilers developers

- CRT frontend is best for large dimension matrix or matrix/vecctor algebra over ciphertexts.
  External products can be amortized.
- Bivariate frontend is best when internal products are rare (e.g. only external products and automorphisms), or chained in sequence.
  CKKS noise propagation at any plaintext precision is easier. (no clamping to the nearest integer "level")

Thank you!