Using Hierarchical Approach to Speed-up RNS Base Extensions in Homomorphic Encryption Context

Morgane Vollmer[1], Karim Bigou[1], Arnaud Tisserand[2]

[1] Université de Bretagne Occidentale / Lab-STICC, UMR CNRS 6285
[2] CNRS / Lab-STICC, UMR CNRS 6285

ARITH-2023, 4–6 September 2023, Portland, Oregon, USA

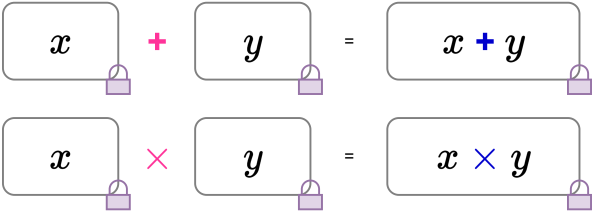
Contents

Contents

Homomorphic Encryption [Gen09]

Homomorphic encryption (HE) allows to perform computations over encrypted data without the knowledge of the secret key

Solution for privacy-concerned applications: medical/machine learning



Huge cost: requires numerous arithmetic operations on very large intermediate data

$\Rightarrow$ Arithmetic is key for HE since it requires operations on very large data

Chosen HE solution: FV scheme [FV12]

The FV scheme handles huge polynomials:

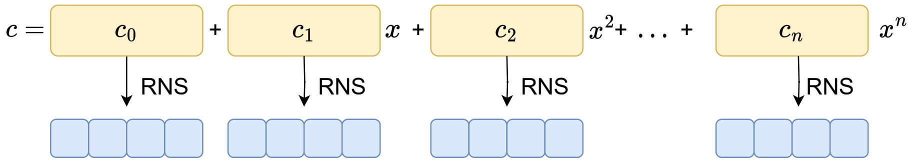
- degree $n \in [2^{11}, 2^{17}]$
- with large coefficients in $\mathbb{Z}/q\mathbb{Z}$ with $\log_2(q) \in [55, 2090]$ bits

$$c = \boxed{c_0} + \boxed{c_1}\, x + \boxed{c_2}\, x^2 + \ldots + \boxed{c_n}\, x^n$$

Chosen HE solution: FV scheme [FV12]

The FV scheme handles huge polynomials:

- degree $n \in \left[2^{11}, 2^{17}\right]$
- with large coefficients in $\mathbb{Z}/q\mathbb{Z}$ with $\log_2(q) \in [55, 2090]$ bits



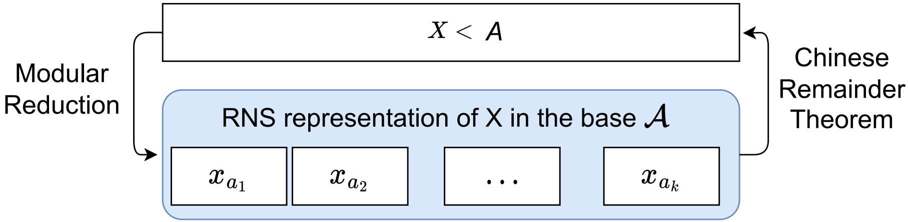
RNS is used for the arithmetic on coefficients

Residue Number System (RNS) [Gar59] [SV55]

RNS is a non-positional representation system based on the Chinese Remainder Theorem (CRT)

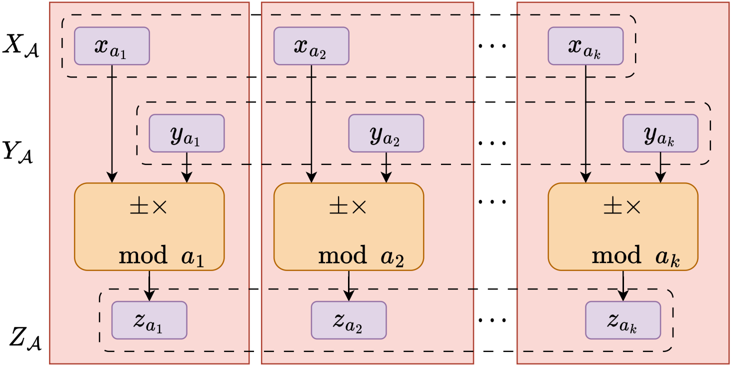
RNS Base: $\mathcal{A} = (a_1, \ldots, a_k)$ tuple of coprime integers of size w bits called **moduli**, $\prod_{i=1}^{k} a_i = A$

Representation of an integer X in the base $\mathcal{A}$:
$X_{\mathcal{A}} = (|X|_{a_1}, \ldots, |X|_{a_k}) = (x_{a_1}, \ldots, x_{a_k})$ where $|X|_{a_i} = X \bmod a_i$



Pros and cons

Pros: Operations $+, -, \times$ computed independently over the residues
$\sim$ **only k small operations over the residues**
$\Rightarrow$ fast and parallel



Cons: Division, rounding and modular reduction are difficult to compute in RNS $\Rightarrow$ intermediate operation: Base Extension

Base Extension

A base extension (BE) converts X in base $\mathcal{A}$ into X in base $\mathcal{B}$
BE costs $\boldsymbol{\sim k^2}$ operations

Most popular technique: computing the CRT in each moduli of the second base

CRT formula:

$$|X|_{b_j} = \left| \left| \sum_{i=1}^{k} \left| x_{a_i} \cdot \frac{a_i}{A} \right|_{a_i} \cdot \frac{A}{a_i} \right|_A \right|_{b_j} = \left| \left(\sum_{i=1}^{k} \left| x_{a_i} \cdot \frac{a_i}{A} \right|_{a_i} \cdot \frac{A}{a_i} \right) - hA \right|_{b_j}$$

Popular BE algorithms:

- Kawamura and al. [KKSS00] (approximate computation of h)
- Shenoy and Kumaresan [SK89] (requires specific conditions)

FV in RNS

Adaptations of FV in RNS:

- first adaptation of FV fully in RNS was proposed in [BEHZ16]
  - speed-up from 5 to 20 for the decryption
  - speed-up from 2 to 4 for the homomorphic multiplication
- variants in [HPS19] [ABVMA18] [ABPA$^+$21] [KPZ21]

Difficult operations:

- roundings
- modular reduction

In the state-of-the-art, half of the homomorphic multiplication is spent doing base extension

Fast Base Extension (FBE)

[BEHZ16] proposed to compute these BE with a Fast Base Extension algorithm (FBE)

$$\mathrm{FBE}(X, \mathcal{A}, \mathcal{B}) = \left| \left(\sum_{i=1}^{k} \left| x_{a_i} \cdot \frac{a_i}{A} \right|_{a_i} \cdot \frac{A}{a_i} \right) \right|_{b_j}$$

- Efficient because it removes the costly modular reduction by A in the CRT formula but gives an approximate result
- [BEHZ16] manages the approximation with some correction steps outside the BE

Hierarchical Base Extension (HBE)

[DBT19] proposed the hierarchical base extension: a hierarchical version of Kawamura Base Extension to speed-up ECC on FPGA

New notation: $\mathcal{A} = (a_1, \ldots, a_k) \longrightarrow \mathcal{A} = \begin{pmatrix} a_{1,1} & \ldots & a_{1,c} \\ \vdots & \ddots & \vdots \\ a_{r,1} & \ldots & a_{r,c} \end{pmatrix}$

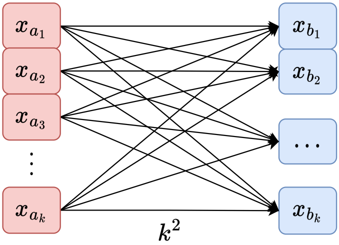
with $k = r \times c$

Idea:

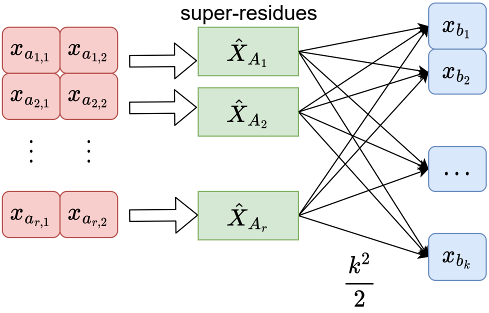
- split the base of k moduli in r sub-bases of c moduli
- compute the CRT in each sub-base to create super-residues
- compute the CRT of the super-residues of base $\mathcal{A}$ in base $\mathcal{B}$

In [DBT19], $c = 2$

Kawamura Base Extension:



HBE: $k = r \times c$ with $c = 2$

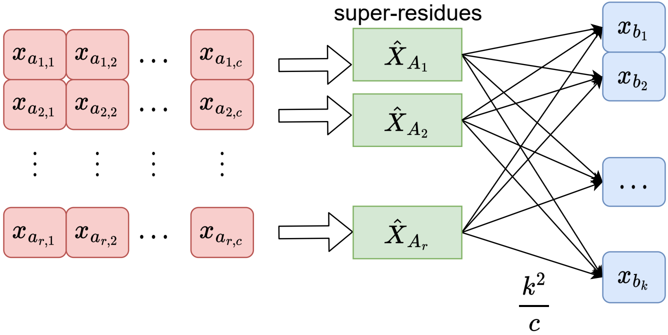


Contents

Proposed Fast Hierarchical Base Extension (FHBE)

We adapt the hierarchical approach to FBE for software implementation of FV

We study the impact of the value of c on the computation cost using prime moduli



$r = \frac{k}{c}$

Algorithm 1: FBE [BEHZ16]

Input: $X_{\mathcal{A}}$

Precomp.: $T_{a_{i,j}}$, $\forall\ i \in [1, r]$ and $\forall\ j \in [1, c]$, $\left|\frac{A}{a_i}\right|_{b_{l,j}}$ $\forall\ i \in [1, k],\ l \in [1, r],\ j \in [1, c]$

Output: $X + \alpha A$ in base $\mathcal{B}$

for i **from** 1 **to** r **parallel do**
    for j **from** 1 **to** c **parallel do**
        $\widehat{x}_{a_{i,j}} \leftarrow \left|x_{a_{i,j}} \times T_{a_{i,j}}\right|_{a_{i,j}}$
for i **from** 1 **to** k **do**
    for l **from** 1 **to** r **parallel do**
        for j **from** 1 **to** c **parallel do**
            $x_{b_{l,j}} \leftarrow \left|x_{b_{l,j}} + \widehat{x}_{a_i} \times \left|\frac{A}{a_i}\right|_{b_{l,j}}\right|_{b_{l,j}}$

Algorithm 2: Proposed Fast Hierarchical BE (FHBE)

Input: $X_{\mathcal{A}}$

Precomp.: $T_{a_{i,j}}$, $\overline{a_{i,j}}$ $\forall\ i \in [1, r]$ and $\forall\ j \in [1, c]$, $\left|\overline{A_i}\right|_{b_{l,j}}$ $\forall\ i \in [1, r]$, $\forall\ l \in [1, r]$ and $\forall\ j \in [1, c]$

Output: $X + \alpha A$ in base $\mathcal{B}$

for i **from** 1 **to** r **parallel do**
    for j **from** 1 **to** c **parallel do**
        $\widehat{x}_{a_{i,j}} \leftarrow \left|x_{a_{i,j}} \times T_{a_{i,j}}\right|_{a_{i,j}}$
for i **from** 1 **to** r **parallel do**
    $\widehat{X}_{A_i} \leftarrow 0$
    for j **from** *1* **to** c **do**
        $\widehat{X}_{A_i} \leftarrow \widehat{X}_{A_i} + \widehat{x}_{a_{i,j}} \times \overline{a_{i,j}}$ *(no reduction)*
for i **from** 1 **to** r **do**
    for l **from** 1 **to** r **parallel do**
        for j **from** 1 **to** c **parallel do**
            $\widehat{x}_{b_{l,j,i}} \leftarrow \left|\widehat{X}_{A_i}\right|_{b_{l,j}}$
            $x_{b_{l,j}} \leftarrow \left|x_{b_{l,j}} + \widehat{x}_{b_{l,j,i}} \times \left|\overline{A_i}\right|_{b_{l,j}}\right|_{b_{l,j}}$

Cost Comparisons

FBE:

Number of operations:

- $k^2 + k$ modular multiplications on w bits

Stored precomputations:

- $k^2 + k$ stored precomputations of size w bits

FHBE:

Number of operations:

- $\frac{k^2}{c} + k$ modular multiplications on w bits
- $w \times (c-1)w$ bits multiplications without reduction
- modular reductions from $cw + \lceil \log_2(c) \rceil$ to w bits

Stored precomputations:

- $\frac{k^2}{c} + k$ stored precomputations of size w bits

Contents

Implementation

Implementation of our FHBE and the FBE algorithms:

- in C language using the GMP multiple-precision arithmetic library version 6.2.0
- GCC compiler version 9.4.0
- Linux Kernel 5.15 from Ubuntu distribution

Performance and memory cost evaluations performed on a Intel Core i7-9850H processor at 2.60GHz

We analyzed:

- the computation cost using the execution time on a single thread
- the pre-computation storage requirements

Evaluation Cases

To use FHBE, k must be divisible ($k = r \times c$)

This leads to two cases:

- k from [BEHZ16] is **divisible**: we use k for both FBE and FHBE
- k from [BEHZ16] is **prime**
  - we use k for FBE
  - we choose a close k with many divisors for FHBE (with the same security in number of bits)

Results when k is divisible

k: number of moduli
w: size of the moduli in bits
$\log_2(q) = k \times w$

The parameters (k, w) come from [BEHZ16]

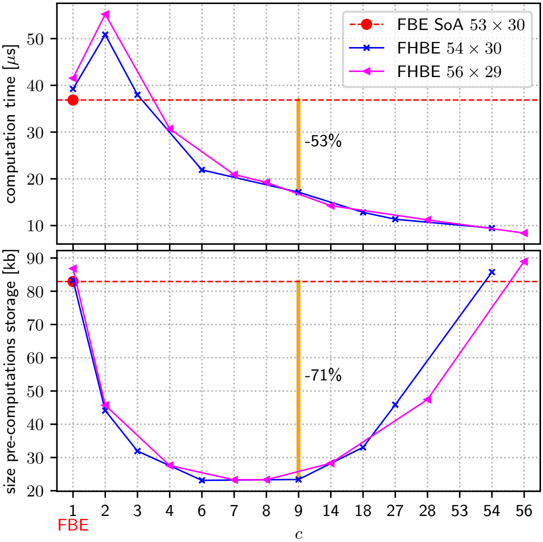
	$k \times w$	FBE	FHBE with c on line below						best gain
			2	3	4	5	6	13	
time [μs]	12×62	2.74	3.47	2.72	2.35	-	**1.96**	-	28%
	26×30	10.21	12.67	-	-	-	-	**4.26**	58%
	25×62	9.53	-	-	-	**6.61**	-	-	30%
size [kb]	12×62	9.53	5.86	**5.14**	5.15	-	5.92		46%
	26×30	20.32	**11.38**	-	-	-	-	11.6	43%
	25×62	39.65	-	-	-	**15.33**	-	-	61%

Results when k is prime

In [BEHZ16], $(k, w) = (53, 30)$ is proposed

53 is prime, then we use:

- $(54, 30)$ with $54 = 2 \times 3^3$
- $(56, 29)$ with $56 = 2^3 \times 7$



Contents

Conclusion

FHBE, a hierarchical variant of the FBE algorithm has been proposed and implemented for software RNS implementations of the FV scheme

It reduces:

- up to 58% the computation cost
- up to 71% the storage requirement

Future prospect:

- optimized multi-core implementation
- complete homomorphic library

Thank you for your attention

[ABPA+21] Ahmad Al Badawi, Yuriy Polyakov, Khin Mi Mi Aung, Bharadwaj Veeravalli et Kurt Rohloff.
Implementation and performance evaluation of RNS variants of the BFV homomorphic encryption scheme.
IEEE Transactions on Computers, 9(2):941–956, avril 2021.

[ABVMA18] Ahmad Al Badawi, Bharadwaj Veeravalli, Chan Fook Mun et Khin Mi Mi Aung.
High-performance FV somewhat homomorphic encryption on GPUs: An implementation using CUDA.
Transactions on CHES, (2):70–95, juillet 2018.

[BEHZ16] Jean-Claude Bajard, Julien Eynard, M. Anwar Hasan et Vincent Zucca.
A full RNS variant of FV like somewhat homomorphic encryption schemes.

In Proc. International Conference on Selected Areas in Cryptography (SAC), volume 10532 de *LNCS*, pages 423–442. Springer, août 2016.

[DBT19] Libey Djath, Karim Bigou et Arnaud Tisserand.
Hierarchical approach in RNS base extension for asymmetric cryptography.
In Proc. International Symposium on Computer Arithmetic (ARITH), pages 46–53. IEEE, juin 2019.

[FV12] Junfeng Fan et Frederik Vercauteren.
Somewhat practical fully homomorphic encryption, 2012.

[Gar59] Harvey L Garner.
The residue number system.
In Western joint Computer Conference, pages 146–153. ACM, 1959.

[Gen09] Craig Gentry.
Fully homomorphic encryption using ideal lattices.
In Proc. Symposium on Theory of Computing (STOC), pages 169–178. ACM, mai 2009.

[HPS19] Shai Halevi, Yuriy Polyakov et Victor Shoup.
An improved RNS variant of the BFV homomorphic encryption scheme.
In Proc. Topics in Cryptology – CT-RSA, pages 83–105. Springer, mars 2019.

[KKSS00] Shinichi Kawamura, Masanobu Koike, Fumihiko Sano et Atsushi Shimbo.
Cox-Rower architecture for fast parallel Montgomery multiplication.
In Proc. Annual International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT), volume 1807 de *LNCS*, pages 523–538. Springer, mai 2000.

[KPZ21] Andrey Kim, Yuriy Polyakov et Vincent Zucca.
Revisiting homomorphic encryption schemes for finite fields.
In Proc. International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT), pages 608–639. Springer, 2021.

[LPR10] Vadim Lyubashevsky, Chris Peikert et Oded Regev.
On ideal lattices and learning with errors over rings.
In Proc. Annual International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT), volume 6110 de *LNCS*, pages 1–23. Springer, 2010.

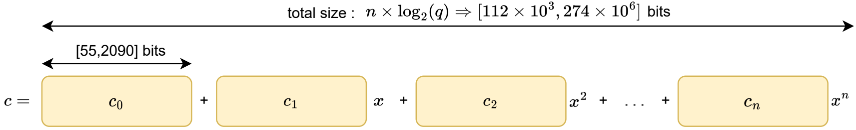
[SK89] A. P. Shenoy et R. Kumaresan.
Fast base extension using a redundant modulus in RNS.
IEEE Transactions on Computers, 38(2):292–297, février 1989.

[SV55] A Svoboda et M Valach.
Operátorové obvody (operator circuits in czech).
In Stroje na Zpracování Informací (Information Processing Machines), pages 3:247–296, 1955.

One popular HE solution: FV scheme [FV12]

FV: HE scheme based on the ring learning with error problem [LPR10]

Plain data: 1 polynomial of degree $n \in \left[2^{11}, 2^{17}\right]$ with coefficients in $\mathbb{Z}/t\mathbb{Z}$ with $t \geq 2$
Cipher data: 2 polynomials of degree $n \in \left[2^{11}, 2^{17}\right]$ with coefficients in $\mathbb{Z}/q\mathbb{Z}$ with $\log_2(q) \in [55, 2090]$ and $t < q$



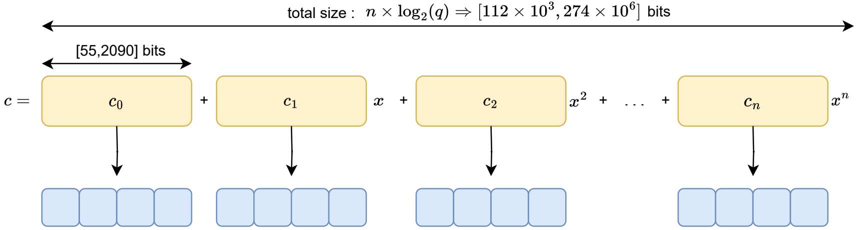
For: 256 B < message < 16 kB ⇒ 28 kB < cipher < 68 MB

One popular HE solution: FV scheme [FV12]

FV: HE scheme based on the ring learning with error problem [LPR10]

Plain data: 1 polynomial of degree $n \in \left[2^{11}, 2^{17}\right]$ with coefficients in $\mathbb{Z}/t\mathbb{Z}$ with $t \geq 2$

Cipher data: 2 polynomials of degree $n \in \left[2^{11}, 2^{17}\right]$ with coefficients in $\mathbb{Z}/q\mathbb{Z}$ with $\log_2(q) \in [55, 2090]$ and $t < q$



For: 256 B $<$ message $<$ 16 kB $\Rightarrow$ 28 kB $<$ cipher $<$ 68 MB

Fast Base Extension (FBE)

$$\mathrm{FBE}(X, \mathcal{A}, \mathcal{B}) = \left| \left(\sum_{i=1}^{k} \left| x_{a_i} \times \frac{a_i}{A} \right|_{a_i} \times \frac{A}{a_i} \right) \right|_{b_j}$$

- Efficient because it removes the costly modular reduction by A in the CRT formula but gives an approximate result
- [BEHZ16] manages the approximation with some correction steps outside the BE

Cost of the BE for an homomorphic multiplication in FV RNS (implementation results from [HPS19])