# An RNS variant of fully homomorphic encryption over integers

by

Ahmed Zawia

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2017

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

**Abstract**

In 1978, the concept of privacy homomorphism was introduced by Rivest *et al.* [35]. Since then, homomorphic cryptosystems have gathered researchers' attention. Most of the early schemes were either partially homomorphic or not secure. The question then arose: was fully homomorphic encryption (FHE) scheme possible? And if so, would it have a practical worth? About thirty years later, Gentry, in his pioneering work [20], constructed the first fully homomorphic encryption scheme. The scheme's security was based on worst-case problems over ideal lattices [21] along with a sparse subset-sum problem. A conceptually simpler scheme was proposed in 2010 by Dijk, Gentry, Halevi, and Vaikuntanathan (DGHV). The scheme is over integers instead of ideal lattices, and its security is based on the hardness of the approximate great common divisor problem (A-GCD). Afterward, different techniques were proposed to reduce ciphertext noise growth and to compress the public key size in order to enhance the practicality of FHE. Moreover, Coron *et al.* [12] proposed and implemented a scale-invariant of the DGHV scheme and a number of optimization techniques including modulus switching (MS). However, FHE over integers is still far from practical. To this end, this work proposes a residue number system (RNS) variant to FHE of [12], which is also applicable to the DGHV scheme. The proposed scheme exploits properties of RNS to perform the required operations over relatively small moduli in parallel. The RNS variant enhances the timing of the original scheme. The variant scheme also improves the original scheme's security, since the former relies only on the hardness of the A-GCD problem and eliminates the need for the sparse-subset-sum problem used in the original MS procedure. Moreover, the public key elements that are required for the MS method is slightly reduced in the RNS variant. Finally, our analysis of the RNS variant reveals a different linear relationship between the noise and the multiplication depth.

**Acknowledgements**

It is my great privilege to express sincere gratitude to my supervisor Professor Anwar Hasan for his endless support and valuable guidance that have helped me succeed and provided me with the insights that I needed to progress. I would like to thank Dr. Julien Eynard who provided expert opinions and comments that greatly aided my research. I wholeheartedly express my gratitude to him for the time he spent and the advice he gave at many stages of my research that played a crucial role towards the success of my study.

## Dedication

I would like to dedicate this thesis to my parents and to my lovely wife.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Over the recent years, *cloud computing* has been rapidly developed to fulfill the demand of outsourcing services. This demand has been driven by critical users whose private data is required to be held in public and multi-tenancy computing and storage environments. These environments are subject to a great variety of attacks that compromise user's sensitive data. To address this issue, sensitive data can be encrypted to make it unreadable for secure transmission and storage. However, such unreadable data cannot be easily used for any computation without first being decrypted. This raises security concerns as users' sensitive data might be vulnerable to theft since it is publicly held by the cloud service provider, which cannot be trusted. This critical problem requires a solution that allows processing the unreadable form of the data without revealing it to the external service provider.

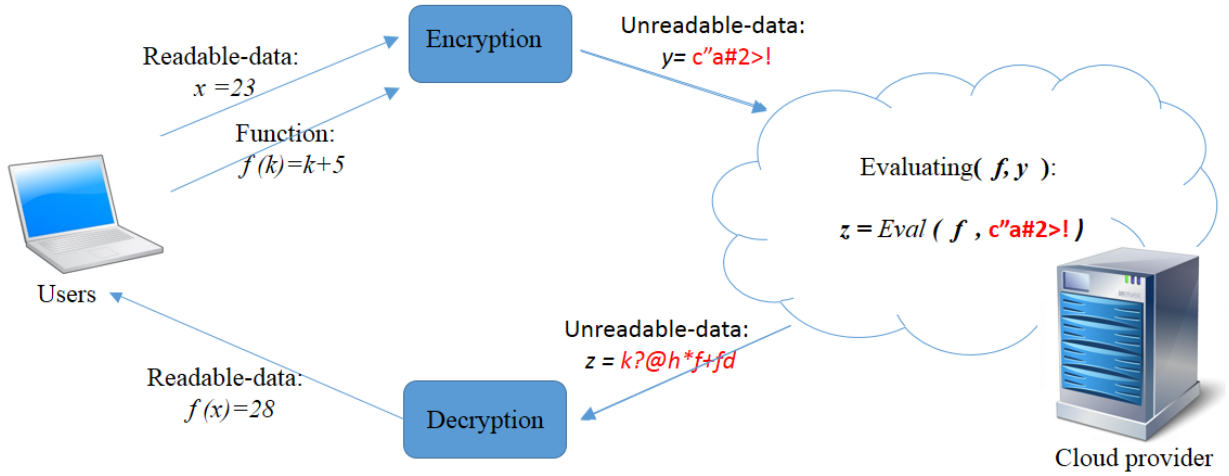


Figure 1.1: The process of homomorphically evaluating users' sensitive data over a public cloud using homomorphic encryption scheme.

## 1.1 Motivation

In the 1970s, Rivest *et al.*[35] addressed aforementioned issue by proposing the notion of *privacy homomorphism*. This concept defines the encryption scheme, which is known as homomorphic encryption that can process complicated functions on the unreadable data. Homomorphic encryption schemes allow users to manipulate their data on external service providers privately and securely as illustrated in Figure 1.1.

There is a growing body of literature that recognized the importance of the homomorphic property and several schemes and protocols were proposed in the past. Unfortunately, most of these schemes ended up to be unsecured or had the partially homomorphic property that supports only a particular class of functions. In 2009, Gentry proposed the first fully homomorphic encryption (FHE) that has the ability to evaluate any boolean function with any depth. Later on, there has been renewed interest in this field where different FHE schemes were introduced. These schemes are best classified under their security as-

sumptions. However, current homomorphic schemes are computationally so expensive that they are considered to be impractical for users whose computation resources are limited. Therefore, a different track of researchers' contributions has targeted the development of practical FHE schemes.

Together, several studies indicate that homomorphic encryption still requires more improvements to be made for practical usages. Therefore, the objective of this study is to investigate whether adopting the operations of FHE scheme over an alternative efficient number system (residue system) will improve the feasibility of our proposed FHE scheme.

## 1.2 Thesis organization

This thesis is divided into five chapters, each chapter devoted to provide significant information to build a road map toward the proposed RNS variant.

Chapter 2 reviews the basic concepts of residue number system arithmetic. Chapter 3 provides a literature review on homomorphic cryptosystems and highlights several researchers' contributions in the area. It also presents a full narrative on two main contributions to the fully homomorphic encryption over integers. Chapter 4 presents all the procedures required to achieve an RNS variant. Chapter 5 lists the contributions of the thesis and proposes several recommendations and suggestions for future research.

# Chapter 2

# Residue Number System Arithmetic

The *residue number system* (RNS) is defined over small independent divisors (known as moduli). The independence between the moduli allows RNS to be a carry-free, borrow-free system that naturally performs parallel addition, subtraction, and multiplication. This property allows RNS to reduce the computation delay. Hence, RNS has gathered considerable attention since the last century where it has been implemented in different applications such as Digital Signal Processing [34], Fast Fourier transform [43], cryptography [4], etc.

The beginning of this chapter introduces initial concepts for residue systems, followed by RNS forward conversion. Section 2.2 discusses RNS representations, system dynamic range and moduli preferences and its influence on RNS performance; this section also introduces the basic RNS arithmetic and its properties. Sections 2.3, 2.4 and 2.5 discuss briefly the mixed radix conversion, RNS base extension, and reverse conversion, respectively.

## 2.1 Preliminaries

### 2.1.1 Remainder theorem

Suppose that $a$ and $m$ are integers s.t. $a > m$. Then for an integer $q > 1$, one can write $a = q \cdot m + r$ where $r \in [0,\ m)$. I.e., $q$ and $r$ are the quotient and the remainder, respectively, resulting from the division of $a$ by $m$. The remainder can be written as.

$$r = a - \left\lfloor \frac{a}{m} \right\rfloor \cdot m \ \equiv \ |a|_m \tag{2.1}$$

where $\lfloor * \rfloor$ is the floor function and $|a|_m$ is $a$ modulo $m$ operation (or $a$ mod $m$ for short).

### 2.1.2 Linear congruence

Two numbers are said to be congruent when they have the same remainder associated with the same divisor.

Suppose $a,\ b,\ r$, and $m$ are integers s.t. $m$ divides $(a - r)$ as well as $(b - r)$, which can be written in the following way:

$$a \equiv b \equiv r \pmod{m} \tag{2.2}$$

The integer $r$ is said to be a residue of $a$ and $b$ with respect to the modulus $m$, and $a,\ b$ and $r$ are congruent modulo $m$. This implies that there are integers $q_1,\ q_2 \geq 0$ and $r \in [0,\ m)$ such that

$$a \equiv q_1 \cdot m + r$$

$$b \equiv q_2 \cdot m + r$$

**Example 2.1.1.** For modulus $m = 7$, one can say that integers 25, 11 and 4 are congruent with respect to $m$. This is because all of them have the same remainder 4 as illustrated in the following

$$25 \equiv 11 \equiv 4 \pmod 7.$$

## 2.2 Arithmetic using RNS

Unlike conventional number systems which consist of weighted numbers, a specific RNS system is defined over a set of independent co-prime moduli (also known as "base"), $\mathcal{B} \equiv \{m_1,\ m_2, \cdots,\ m_n\}$ where the system dynamic space is $M = \prod_{i=1}^{n} m_i$ . Accordingly, an integer $C$ has RNS representation defined in the following set: $C \xrightarrow{RNS} |\mathbf{c}|_{\mathcal{B}} = (c_1, c_2, \cdots, c_n)$ where:

$$C \equiv c_i \pmod{m_i}.$$

**Example 2.2.1.** Given an RNS defined over base $\mathcal{B} \equiv \{3, 5, 7\}$, the RNS representation of $C = 442$ and $X = 69$ are:

| $C = 442$ | $X = 69$ |
|---|---|
| $442 \equiv 1 \pmod 3$ | $69 \equiv 3 \pmod 3$ |
| $442 \equiv 2 \pmod 5$ | $69 \equiv 4 \pmod 5$ |
| $442 \equiv 1 \pmod 7$ | $69 \equiv 6 \pmod 7$ |

$$\text{RNS} \rightarrow \quad 442 \xrightarrow{RNS} |\mathbf{c}|_{\mathcal{B}} = (1, 2, 1) \quad 69 \xrightarrow{RNS} |\mathbf{x}|_{\mathcal{B}} = (3, 4, 6)$$

We note that it is difficult to identify which integer is bigger just based on $|\mathbf{c}|_{\mathcal{B}}$ and $|\mathbf{x}|_{\mathcal{B}}$, in the sense that $|\mathbf{x}|_{\mathcal{B}}$ has greater residue values does not indicate that $X$ is bigger.

This is because RNS is not a weighted number system where the only relationship between residues is the fact that they form a tuple of remainders to the same integer.

### 2.2.1 Uniqueness of RNS representation

The uniqueness property implies that there is only one representation associated with a unique integer $C$ over a set of moduli.

Suppose $N$ and $m_i$ are integers s.t. $N \equiv 0 \pmod{m_i}$. Then, for an arbitrary positive integer $C$ in the interval $[0,\ m_i]$, one can assume that $N + C$ has a unique representation over $m_i$ s.t. $N \equiv 0 \pmod{m_i}$, and accordingly $N + 1 \equiv 1 \pmod{m_i}$ and so on up-to $N + (m_i - 1) \equiv m_i - 1 \pmod{m_i}$. The uniqueness property will apply except in case $C = m_i$ where the two numbers $N + C$ and $N$ are congruent.

$$N + C \equiv N \pmod{m_i} \quad s.t.\ C = m_i$$

For $N = 0$, $C$ satisfies the uniqueness property when $0 \leq C \leq m_i - 1$. Likewise, for an RNS defined by a set of co-prime moduli $\mathcal{B} \equiv \{m_1, m_2, \cdots, m_n\}$, an integer $C$ has representation in each modulus s.t. $0 \leq |C|_{m_i} < m_i,\ \forall i \in [1, n]$. Thus for unique RNS representation, $C$ should satisfy $0 \leq\ C < \prod_{i=1}^{n} m_i$, so that there are no two integers that have congruent RNS representation over $\mathcal{B}$. In other words, $C$ should be in the system dynamic space $\{0, 1, 2, \cdots, M - 1\}$, where $M = \prod_{i=1}^{n} m_i$, so that the uniqueness property holds.

**Example 2.2.2.** Recalling Ex. 2.2.1, the dynamic space $M$ for the specified RNS over $\mathcal{B} \equiv \{3, 5, 7\}$ is $M = \prod_{i=1}^{n} m_i = 3 \cdot 5 \cdot 7 =\ 105$. Since $N = 442 > M$, the resultant RNS representation $|442|_{\mathcal{B}} \equiv \{1, 2, 1\}$ is not unique. Consequently, there are a number of integers less than $N$ that share the same representation. Furthermore, their number is

equal to the *quotient* of $N$ to $M$, $q = \lfloor \frac{442}{105} \rfloor = 4$. The four integers are $22, 127, 337$ and $442$ itself, all of which share the same representation over $\mathcal{B}$:

$$442 \equiv 337 \equiv 127 \equiv 22 \equiv 1 \pmod 3$$

$$442 \equiv 337 \equiv 127 \equiv 22 \equiv 2 \pmod 5$$

$$442 \equiv 337 \equiv 127 \equiv 22 \equiv 1 \pmod 7$$

$$442 \equiv 337 \equiv 127 \equiv 22 \xrightarrow{RNS} \{1, 2, 1\}$$

### 2.2.2 RNS representation over negative integers

In the above discussion, $C$ was assumed to be a positive integer. However, there is a need for some applications to represent negative integers as well. Generally, the adequate range for positive and negative integers over RNS is specified by equally dividing the RNS dynamic space $M$ as discussed below.

For an integer $C \in \mathbb{Z}$, the range of the representation over a specific RNS composed of $n$ independent moduli with dynamic space $M = \prod_{i=1}^{n} m_i$ is as follows: the range allocated for positive integer $C_p$ is defined over $0 \leqslant C_p \leqslant \left\lfloor \frac{M}{2} \right\rfloor$. Accordingly, the range allocated to negative $C_n$ is over $\left\lfloor \frac{M}{2} \right\rfloor < C_n < M$. Therefor, the allocated space of a variable $C$ is as follows:

$$\begin{aligned} -\tfrac{M-1}{2} \;&\leqslant\; C \;\leqslant\; \tfrac{M-1}{2} && \text{for M is odd} \\ -\tfrac{M}{2} \;&\leqslant\; C \;\leqslant\; \tfrac{M}{2} - 1 && \text{for M is even} \end{aligned}$$

It is worthwhile to mention that the reason for subtracting one in the even space $M$ is that zero has been assumed to be among the positive integers. Furthermore, the reason for allocating $C_n$ over $\left\lfloor \frac{M}{2} \right\rfloor < C_n < M$ is because $C_n$ considered as complementary of $M$.

### 2.2.3 Moduli preferences

Choosing a proper base for RNS is very important to enhance the performance and ensure the correctness of such a system. For instance, co-prime moduli set is the natural choice to ensure correctness. However, different aspects control appropriate (efficient) moduli set selection in different applications, such aspects as:

- The moduli set should be co-prime.
- To ensure unique representation, the product of moduli should be large enough to adequate the dynamic range and prevent overflow.
- To enhance speed performace, the moduli bit size should be as small as possible to reduce computation time [32]. Also, the different bit sizes between moduli should not be significant. This is because there are no advantages to use small moduli with large ones [1].
- To ensure efficient binary representation, conversion and arithmetic, the moduli can be formulated as $\{2^k, 2^k - 1, ...2^{kn}, 2^{kn} - 1\}$ [1, 24, 42, 17].

In addition, choosing a random moduli set usually produces an inefficient system that complicates hardware implementation. Moreover, in the forward conversion and arithmetic operations, system's delay is dominated by the worst modulus, and it is worth to mention that worst modulus might be defined in terms of hardware implementation complexity, or the largest modulus size. In contrast, reverse conversion is most complicated operation since its performance is dominated by the characteristic of all moduli.

### 2.2.4 RNS arithmetic property

RNS is also known as a residue encoding system. This implies that for a particular integer $C$, the associated RNS representation carries independent pieces of information held with each base. Furthermore, these pieces reassemble the conventional representation of that integer $C$. Consequently, the residue representation of integer $C$ also comprises the same relation that $C$ holds. Given the representations for integers $A$ and $B$ for a specified RNS over $\mathcal{B} \equiv \{m_1, m_2, \cdots, m_n\}$

$$A \xrightarrow{RNS} (a_1, a_2, \cdots, a_n)$$
$$B \xrightarrow{RNS} (b_1, b_2, \cdots, b_n)$$

there is an RNS equivalent to an integer $C$ that satisfies some relation between $A$ and $B$ s.t. $C = A \propto B$. Using the remainder theorem, the residue for both sides associated with one of the moduli (namely $m_i$) are:

$$C \equiv \ c_i \bmod m_i \tag{2.3}$$
$$A \propto B \equiv \ a \pmod{m_i} \propto b \pmod{m_i} \equiv \ c_e \bmod m_i$$

recalling linear congruence in Section 2.1.2, which implies that the integer $c_e$ is equivalent to $C$ and $c_i \pmod{m_i}$.

$$C \equiv c_e \equiv a \propto b \equiv c_i \bmod m_i$$

Therefore, the RNS representation of $C$ over $\mathcal{B}$ can be formulated as follows:

$$C = A \propto B \xrightarrow{RNS} (a_1 \propto b_1, a_2 \propto b_2, ..., a_n \propto b_n) \equiv (c_1, c_2, ..., c_n) \tag{2.4}$$

### Addition operation

Eq. (2.4) holds for $\propto$ being an addition. Thus, for given $A$, $B$ and $C$, s.t. $C = A + B$, the unique RNS representation of $C$ can be computed as follows:

$$\begin{aligned} A &\xrightarrow{RNS} (\ a_1, \quad a_2, \quad ... \quad , a_n \quad ) \\ B &\xrightarrow{RNS} (\ b_1, \quad b_2, \quad ... \quad , b_n \quad ) \\ \hline C &\xrightarrow{RNS} (|a_1 + b_1|_{m_1}, |a_2 + b_2|_{m_2}, ..., |a_n + b_n|_{m_n}\ ) \end{aligned}$$

where $0 \leqslant A, B \leqslant C < M = \prod_{i=1}^{n} m_i$.

**Example 2.2.3.** Let $A$ and $B$ be 46 and 54, respectively. The addition of these integers over RNS defined over $\mathcal{B} \equiv \{3, 5, 7\}$ can be performed as follows:

$$Base : \{\ 3, \quad 5, \quad 7 \quad \}$$

$$\begin{aligned} A &\xrightarrow{RNS} (\ 1, \quad 1, \quad 4 \quad ) \\ B &\xrightarrow{RNS} (\ 0, \quad 4, \quad 5 \quad ) \quad + \\ \hline C &\xrightarrow{RNS} (\ |1|_3 \quad , |5|_5, \quad |9|_7\ ) \end{aligned}$$

Recalling the congruent property, the resultant system of congruence must satisfy all congruent relationships which gives the following equivalent result $(\ |1|_3, \quad |0|_5, \quad |2|_7\ ) \Rightarrow C = 100$.

It is worthwhile to mention that $|C|_{\mathcal{B}}$ is unique since $A$, $B$ and $C$ are in the specified RNS dynamic space $M$ where $0 < A = 46 < B = 54 < C = 100 < 105 = 3 \cdot 5 \cdot 7$. Moreover,

the overflow in base 7 that occurred when adding $|4+5|_7$ has no effect on residues of the other moduli. In conclusion, the addition carried out in each modulus has been done independently, and the result does not depend on the intermediate overflow in the RNS moduli; For this reason, RNS is called carry-free (borrow-free) system which enables RNS to perform correct parallel computation.

### Multiplication operation

Similarly, Eq. (2.4) holds for $\propto$ being a multiplication operation with the same uniqueness condition for integers $A$, $B$ and $C$ s.t. $C = A \times B$ and $0 \leqslant A, B \leqslant C < M = \prod_{i=1}^{n} m_i$. Hence, $C$ can be computed in RNS as follows:

$$
\begin{array}{llllll}
A \xrightarrow{RNS} ( & a_1, & a_2, & \ldots & , a_n & ) \\
B \xrightarrow{RNS} ( & b_1, & b_2, & \ldots & , b_n & ) \\
\hline
C \xrightarrow{RNS} ( & |a_1 \times b_1|_{m_1}, & |a_2 \times b_2|_{m_2}, & \ldots & , |a_n \times b_n|_{m_n} & )
\end{array}
$$

**Example 2.2.4.** For RNS defined over $\mathcal{B} \equiv \{3, 5, 7\}$ and for integers $A$ and $B$ that are equal to 4 and 25, respectively, the multiplication of these two integers can be preformed as follows:

$$
\begin{array}{llll}
Bases : \{ & 3, & 5, & 7 \quad \} \\
 & & & \\
A \xrightarrow{RNS} ( & 1, & 4, & 4 \quad ) \\
B \xrightarrow{RNS} ( & 1, & 0, & 4 \quad ) \quad \times \\
\hline
C \xrightarrow{RNS} ( & |1|_3, & |0|_5, & |2|_7 \ )
\end{array}
$$

Similar to addition, the resultant RNS representation of $C$ is unique and satisfies: $0 < A = 4 < B = 25 < C = 100 < 105 = 3 \cdot 5 \cdot 7$. Furthermore, the multiplication carried out in each modulus has been done independently which enables RNS to perform parallel computation.

**Additive inverse:**

An integer $C_{in}$ is said to be the additive inverse to $C$ over base $M$, if it satisfies the following relation:

$$|C + C_{in}|_M \equiv 0 \equiv M \tag{2.5}$$

Then $C_{in}$ can be written as follows:

$$C_{in} = M - C \tag{2.6}$$

by considering the fact that the representation of negative numbers, $M/2 \leq C_n < M$, has an equal dynamic space as the positive integer, $0 \leq C_P < M/2$, as defined in Section 2.2.2. Therefor, every integer in range $0 < x < M - 1$ has a unique additive inverse.

Similar to addition and multiplication, subtraction follows Eq. (2.4) as well s.t. $A - B \xrightarrow{RNS} (\ a_1 - b_1,\ a_2 - b_2, ..., a_n - b_n)$. In addition, we can use the additive inverse property to rewrite the subtraction relation into: $C = A - B_{in} \xrightarrow{RNS} (\ a_1 + b_{1_{in}},\ a_2 + b_{2_{in}}, ..., a_n + b_{n_{in}})$.

**Example 2.2.5.** Given $A = 25$ and $B = 16$ in the RNS defined over $\mathcal{B} \equiv \{3, 5, 7\}$,

determine $A - B$.

$$\begin{array}{rcrcl} & & & \mathcal{B}: & \{3, \quad 5, \quad 7\} \\ A & = & 25 & \xrightarrow{RNS} & (1, \quad 0, \quad 4) \\ B_{in} & = & 16 & \xrightarrow{RNS} & (1, \quad 1, \quad 2) \quad - \\ \hline C & = & 9 & \xrightarrow{RNS} & (|0|_3, \quad |-1|_5, \quad |2|_7) \end{array}$$

Recalling the additive inverse property, $|-1|_5 \equiv |(5-1)|_5$, we have $C \xrightarrow{RNS} \{\ |0|_3,\ |4|_5,\ |2|_7\ \}$. This is equivalent to $A + (M - B) = 25 + (105 - 16) = 25 + 89$ and can be computed as:

$$\begin{array}{rcrcl} & & & \mathcal{B}: & \{3, \quad 5, \quad 7\} \\ A & = & 25 & \xrightarrow{RNS} & (1, \quad 0, \quad 4) \\ B_{in} & = & 89 & \xrightarrow{RNS} & (1, \quad 1, \quad 2) \quad + \\ \hline C & = & 114 & \xrightarrow{RNS} & (|3|_3, \quad |4|_5, \quad |9|_7) \end{array}$$

We note that the RNS's dynamic space has been equally divided between positive integers $\in [0,\ 53)$, and negative integers in $\in [53,\ 105)$ where they represent 53 unique additive inverses for positive integers.

### Multiplicative inverse

For non-zero integer $C$ s.t. $C$ and $M$ are co-prime, there is a unique integer $X$ that satisfies:

$$|C \cdot X|_M \equiv |1|_M \tag{2.7}$$

where $X$ is said to be the multiplicative inverse to $C$ modulo $M$, $X \equiv C^{-1} \equiv \frac{1}{C}$. Reciprocally, $C$ is the multiplicative inverse of $X$ modulo $M$. On the other hand, if $C$ and $M$ are not co-prime then there is no such integer $X$ that satisfies Eq. (2.7).

Similarly, an integer $C$ has multiplicative inverse in specified RNS, if $C$ is co-prime to all of the RNS bases. On other words, if there are one or more zero-residues in $|C|_{\mathcal{B}}$, then $C$ will not have a multiplicative inverse in the space $M$. For instance, given RNS defined over $\mathcal{B} \equiv \{3, 5, 7\}$ with $M = 105$, and $C = 15$, then $C^{-1}$ does not exist. This is because both of the bases 3 and 5 divide $C$ which is clear in the $C$ representation $|C|_{\mathcal{B}} = (0, 0, 1)$. In contrary, for $C = 13$ with $|C|_{\mathcal{B}} = (1, 3, 6)$, $C$ has inverse over $M$ ($C^{-1} = 97$). In general, given a co-prime dynamic range, it is certain that there are a multiplicative inverse for all space members as illustrated in Table 2.1.

Table 2.1: The Multiplicative inverse for different moduli.

| Prime $m$=7 | | Even $m$=8 | | Odd $m$=9 | |
|---|---|---|---|---|---|
| c | $c^{-1}$ | c | $c^{-1}$ | c | $c^{-1}$ |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 4 | 2 | None | 2 | 5 |
| 3 | 5 | 3 | 3 | 3 | None |
| 4 | 2 | 4 | None | 4 | 7 |
| 5 | 3 | 5 | 5 | 5 | 2 |
| 6 | 6 | 6 | None | 6 | None |
| | | 7 | 7 | 7 | 4 |
| | | | | 8 | 8 |

Even though there is no explicit expression to determine multiplicative inverse of an

integer modulo $M$, Fermat's theorem can be used for a prime modulus as special case. In fact, a brute-force algorithm can be used to search for a multiplicative inverse, if it exists, but this approach is inefficient for large moduli. Thus executing the extended Euclidean algorithm is considered to be a more efficient alternative.

### Division

In contrast to addition and multiplication, performing the division operation in RNS is more complex. Hence, it is suitable to classify division into categories where the first two are special cases of the last one:

- **Exact division:** The division is between two integers where the dividend is exact multiple of the divisor. For an integer $A$ and $B$ s.t. $A = q \cdot B$ or $|A|_B \equiv 0$, the division $A/B$ is defined over modulo m as follows:

$$|A|_m \equiv |q \cdot B|_m$$

Therefore, for the multiplicative inverse of $B$, we have

$$|A \cdot B^{-1}|_m \equiv |q|_m \tag{2.8}$$

and for a given RNS base $\mathcal{B}$, $A/B$ can be expressed as:

$$q = \frac{A}{B} \xrightarrow{RNS} |q|_{\mathcal{B}} \equiv (\ |a_1 \cdot b_1^{-1}|_{m_1},\ \ |a_2 \cdot b_2^{-1}|_{m_2},\ \ ..., |a_n \cdot b_n^{-1}|_{m_n}\ )$$

**Example 2.2.6.** Divide $A = 1530$ by $B = 17$, which is an exact multiple of the

dividend $A$, using RNS defined over $\mathcal{B} \equiv \{23, 29, 31\}$ and $M = 20677$:

$$\begin{array}{rcccccccc} & & & & \mathcal{B}: & \{ & 23, & 29, & 31 \quad \} \\ A & = & 1530 & \xrightarrow{RNS} & ( & 12, & 22, & 11 \quad ) & \\ |B^{-1}|_{20677} & = & 12163 & \xrightarrow{RNS} & ( & 19, & 12, & 11 \quad ) & \times \\ \hline q & = & |90|_{20677} & \xrightarrow{RNS} & ( & |19|_{23}, & |12|_{29}, & |11|_{31} \quad ) & \end{array}$$

where $|B|_{\mathcal{B}} \equiv (17, 17, 17)$, and $|17^{-1}|_{23} \equiv 19$, $|17^{-1}|_{29} \equiv 12$ and$|17^{-1}|_{31} \equiv 11$.

Indeed, $q = 90$ and it is surely true because $A$ is the exact multiple of $B$. After all, the exact division can be applied on some applications that ensure zero remainder division but what if $B$ does not exactly divide $A$ as following:

**Example 2.2.7.** To discuss the exact division, let the dividend and the divisor be 9 and 3, respectively. Then the exact division modulo 17 is:

$$q = |\ 9\ \cdot |3^{-1}|_{17}|_{17} \equiv\ |\ 9 \cdot 6\ |_{17} \equiv |3|_{17}$$

Thus Eq. (2.8) holds, but what about the division of 9 by 4 modulo 17:

$$q = |\ 9\ \cdot |4^{-1}|_{17}|_{17} \equiv\ |\ 9 \cdot 13\ |_{17} \equiv |15|_{17}$$

In this case, Eq. ( 2.8) does not hold. This is because there is a remainder included in the result as illustrated in the following: The exact answer for the above is 2.25 which consists of $= 2 + |4^{-1}|_{17} \equiv |15|_{17}$, as well as for $5/2 = 2.5 = 2 + |2^{-1}|_{17} \equiv |2 + 9|_{17}$ and for $9/5 \equiv 1 + |4 \cdot 5^{-1}|_{17} \equiv |12|_{17}$ (where $4 \equiv |9|_5$). However, to correct the result, we apply floor rounding to the division by deducting the remainder of the dividend as illustrated in scaling division.

- **Scaling division:** The divisor, here, is a known constant which consists of one modulus or the product of multiple moduli. Scaling is much easier and faster than general division. This is because dividing by a predetermined constant is fairly less complex and faster than dividing by a random integer.

  Suppose that $A$ and $B$ are two integers that satisfy the following expression:

$$A = \left\lfloor \frac{A}{B} \right\rfloor \cdot B + |A|_B$$

  Thus we have the following form of division

$$\left\lfloor \frac{A}{B} \right\rfloor = \frac{A - |A|_B}{B} \tag{2.9}$$

  Since $B$ is predetermined that has a multiplicative inverse modulo $M$ (which can be pre-computed):

$$\left\lfloor \frac{A}{B} \right\rfloor = (A - |A|_B) \cdot |B^{-1}|_M \tag{2.10}$$

  To carry out the above scaling division over specific RNS characterized by $\mathcal{B} \equiv \{m_1, m_2, \cdots, m_n\}$, $B$ has to be a co-prime to all moduli in set $\mathcal{B}$.

$$\text{RNS}(\left\lfloor \frac{A}{B} \right\rfloor)_{\mathcal{B}} \equiv (|(a_1 - ||A|_B \ |_{m_1}) \cdot |b_1^{-1}|_{m_1} \ |_{m_1}, \ |(a_2 - ||A|_B \ |_{m_2}) \cdot |b_2^{-1}|_{m_2} \ |_{m_2}, \cdots \\ \cdots \ |(a_n - ||A|_B \ |_{m_n}) \cdot |b_n^{-1}|_{m_n} \ |_{m_n})$$

  As mentioned above, it is worth to state that, if $B$ is equal to one or a product of multiple base elements (denoted as $m_x$), then Eq. (2.10) applies to all base elements except $m_x$ set, and the result will be in RNS dynamic space equal to $M/m_x$. In order to restore the original space $M$, base extension is required, and will be discussed in Section 2.4.

  In what follows, two examples will be used to illustrate scaling division by one and two RNS base elements.

**Example 2.2.8.** In RNS defined over $\mathcal{B} \equiv \{3, 5, 7, 11, 13\}$ with $M = 15015$, scale $A = 1392$ by $B = 13$.

The representation of $A$ in all base elements except $m_5$ is $(0, 2, 6, 6)$, and $|A|_B \equiv 1$, then we have:

$$Base : \{3, \quad 5, \quad 7, \quad 11 \quad \}$$

$$\begin{aligned} A = 1392 &\xrightarrow{RNS} (\ 0, \quad 2, \quad 6, \quad 6 \quad ) \\ |A|_{m_5} = 1 &\xrightarrow{RNS} (\ 1, \quad 1, \quad 1, \quad 1 \quad ) \quad - \\ \hline A - |A|_{m_5} = 1391 &\xrightarrow{RNS} (\ 2, \quad 1, \quad 5, \quad 5 \quad ) \end{aligned}$$

Now multiply by RNS$(|B^{-1}|_{M^*})_{\mathcal{B}} \equiv (1, 2, 6, 6)$, for $|B^{-1}|_{M^*} = 622$ and $M^* = \dfrac{M}{13}$.

$$\begin{aligned} A - |A|_{m_5} &\xrightarrow{RNS} (\ 2, \quad 1, \quad 5, \quad 5 \quad ) \\ |B^{-1}|_{M^*} = 622 &\xrightarrow{RNS} (\ 1, \quad 2, \quad 6, \quad 6 \quad ) \quad \times \\ \hline C = 107 &\xrightarrow{RNS} (\ |2|_3, \quad |2|_5, \quad |2|_7, \quad |8|_{11}) \end{aligned}$$

Note that the result is in the RNS dynamic space $M^* = \dfrac{M}{13}$. The final step is to preform a base extension which will be carried out in Example 2.4.1. An easy way to return to the original system is to concatenate $|107|_{13} \equiv 3$ under base 13. Then, the final result in the original RNS representation would be:

$$\left\lfloor \frac{1392}{13} \right\rfloor = 107 \xrightarrow{RNS} (\ |2|_3, \quad |2|_5, \quad |2|_7, \quad |8|_{11}, \quad |3|_{13})$$

- **General division** is one of the most difficult operations. Currently, there are different algorithms performing division that can be grouped into two categories based on their iteration operation [17] [10]. The first category is based on iterative subtraction and comparison operation, such as in [18, 25, 28, 44] and the second category is multiplicative iteration algorithms, such as [46]. The former is similar to conventional binary division where it subtracts multiples of the divisor from the dividend repeatedly until the deference becomes less that the divisor. This carries a prime disadvantage in the sense that it requires a number of costly sequential (repeated) magnitude comparisons. The second category, multiplicative division algorithms, determines the quotient by multiplying the reciprocal of the divisor with the dividend. This approach is based on division by an approximate divisor which is applicable in scaling algorithms. This procedure introduces some error that can be reduced by recursion. Unlike the first category, the latter does not require repeated magnitude comparisons; yet, the disadvantage of the multiplicative division algorithm is that it does not know when the algorithm should terminate for a given acceptable error.

## 2.3 Mixed radix conversion

The mixed radix conversion was first mentioned by Garner [19]. This operation is the conversion from a un-weighted RNS representation to a weighted mixed radix number system (MRS). The resultant MRS has the same space as the original RNS.

For an integer $A$ represented in RNS defined over $\{m_1, m_2, \ ..., m_n\}$, the corresponding MRS representation is defined as the following:

$$A = a'_1 + a'_2 \cdot m_1 \ + a'_3 \cdot m_1 \cdot m_2 \ + \ ... \ + a'_n \prod_{i=1}^{n-1} m_i \tag{2.11}$$

where $|A|_{m_1} \equiv a'_1$, and the mixed radix of $A$ denoted as MRS($A$)= $\{a'_1, a'_2, \cdots, a'_n\}$.

In order to obtain MRS($A$)'s coefficients, a sequence of operations is required. Accordingly, $a'_2$ can be computed by moving $a'_1$ to the left hand side, and then taking Eq. (2.11) mod $m_2$:

$$|A - a'_1|_{m_2} = a'_2 \cdot m_1$$
$$a'_2 = |\ |A - a'_1|_{m_2} \cdot |m_1^{-1}|_{m_2}\ |_{m_2} \equiv \left| \left\lfloor \frac{A}{m_1} \right\rfloor \right|_{m_2}$$

Similarly for $a'_3$:

$$|A - (\ a'_1 + a'_2 \cdot m_1\ )|_{m_3} = a'_3 \cdot m_1 \cdot m_2$$
$$a'_3 = |\ |A - a'_1 + a'_2 \cdot m_1|_{m_3} \cdot |(\ m_1 \cdot m_2\ )^{-1}|_{m_3}\ |_{m_3} \equiv \left| \left\lfloor \frac{A}{m_1 \cdot m_2} \right\rfloor \right|_{m_3}$$

And so on. Hence, the conversion is a sequential operations where finding $a'_n$ requires finding all $a'_i$, up to $i = n - 1$. This will slowdown the mixed redix conversion procedure when we will have a very large number of moduli.

**Example 2.3.1.** Given RNS defined over $\mathcal{B} \equiv \{3, 5, 7\}$, convert RNS$(85)_{\mathcal{B}} = \{1, 0, 1\}$ to

its equivalent representation in MRS

$$
\begin{array}{lcccccl}
Base: & \equiv & 3 & 5 & 7 & & \\
A=85 & \xrightarrow{RNS} & 1 & 0 & 1 & & \\
|A|_{m_1} & \xrightarrow{RNS} & 1 & 1 & 1 & - & \Rightarrow a_1' = |85|_{m_1} \\
\hline
 & & 0 & 4 & 0 & & \\
|\frac{1}{3}|_{m_1} & \xrightarrow{RNS} & & 2 & 6 & \times & \\
\hline
 & & & 3 & 0 & & \Rightarrow a_2' = |3|_{m_2} \\
a_2'=3 & \xrightarrow{RNS} & & 3 & 3 & - & \\
\hline
 & & & 0 & 4 & & \\
|\frac{1}{5}|_{m_2} & \xrightarrow{RNS} & & & 3 & \times & \\
\hline
 & & & & 5 & & \Rightarrow a_3' = |5|_{m_3}
\end{array}
\tag{2.12}
$$

Thus, the corresponding to RNS$(85)_{\mathcal{B}}$ $\Rightarrow$MRS$(85)_{\mathcal{B}} \equiv (1, 3, 5)$.

Since MRS is a weighted system, it eases the comparison operation relative to RNS. Moreover, it is fairly fast in residue computer to perform conversion from RNS$_{\mathcal{B}}$ to MRS$_{\mathcal{B}}$ [17]. This advantage applies for small moduli sets which can be efficiently implemented in hardware.

## 2.4 Base extension

The operation that is used to extend the original base $\mathcal{B}$ of an RNS by adding one or more moduli is known as *base extension*. Precisely, this operation determines the residue modulo $m_{new}$ from residues of an integer over $\mathcal{B}$. This is in order to expand or restore the RNS dynamic space M by a factor of the new modulus. Base extension is utilized to achieve

sign detection, comparison and overflow detection operations [33] and scaling as described in Example 2.2.8.

Szabo and Tanaka [17] proposed a procedure to find the extra residue modulo $m_{new}$. The procedure converts the un-weighted RNS to its corresponding weighted mixed radix system.

For RNS characterized by $\mathcal{B} \equiv \{m_1, m_2, ..., m_n\}$ over the interval $[0,\ M = \prod_{i=1}^{n} m_i)$, the extended RNS version, by the new base element $m_{new}$, will be in the interval $[0,\ M \cdot m_{new})$. Thus, the MRS representation, including $m_{new}$, will be:

$$A = a'_1 + a'_2 \ \cdot \ m_1 \ + \ ... \ + a'_n \prod_{i=1}^{n-1} m_i \ + \ a'_{n+1} \prod_{i=1}^{n} m_i \tag{2.13}$$

Note that the base extension procedure does not change the value of the integer $A$ thus $a'_{n+1}$ is clearly equal to zero. Therefore, the residue of $A$ mod $m_{new}$ can be obtain by determining all MRS coefficients and use the fact that $a'_{n+1}$ is zero. The following example will be used to illustrate the Szabo and Tanaka base conversion procedure.

**Example 2.4.1.** Recall Ex. 2.2.8, as last step in scaling operation is to recover the base 13 by using base extension.

First, perform the MRS conversion as follows.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $Base:$ | $\equiv$ | 3 | 5 | 7 | 11 | 13 | | |
| | $\xrightarrow{RNS}$ | 2 | 2 | 2 | 8 | $c$ | | |
| $\lvert A\rvert_{m_1}$ | $\xrightarrow{RNS}$ | 2 | 2 | 2 | 2 | 2 | $-$ | $\Rightarrow a'_1 = \lvert 107\rvert_{m_0=3}$ |
| | | 0 | 0 | 0 | 6 | $c+11$ | | |
| $\lvert\frac{1}{3}\rvert_{m_1=5}$ | $\xrightarrow{RNS}$ | | 2 | 5 | 4 | 9 | $\times$ | |
| | | | 0 | 0 | 2 | $9c+8$ | | $\Rightarrow a'_2 = \lvert 0\rvert_{m_1}$ |
| $a'_2 = 0$ | $\xrightarrow{RNS}$ | | 0 | 0 | 0 | 0 | $-$ | |
| | | | 0 | 0 | 2 | $9c+8$ | | |
| $\lvert\frac{1}{5}\rvert_{m_2=7}$ | $\xrightarrow{RNS}$ | | | 3 | 9 | 8 | $\times$ | |
| | | | | 0 | 7 | $7c+12$ | | $\Rightarrow a'_3 = \lvert 0\rvert_{m_2}$ |
| $a'_3 = 0$ | $\xrightarrow{RNS}$ | | | 0 | 0 | 0 | $-$ | |
| | | | | 0 | 7 | $7c+12$ | | |
| $\lvert\frac{1}{7}\rvert_{m_3=11}$ | $\xrightarrow{RNS}$ | | | | 8 | 2 | $\times$ | |
| | | | | | 1 | $c+11$ | | $\Rightarrow a'_4 = \lvert 1\rvert_{m_3}$ |
| $a'_4 = 1$ | $\xrightarrow{RNS}$ | | | 0 | 1 | 1 | $-$ | |
| | | | | | 0 | $c+10$ | | |
| $\lvert\frac{1}{11}\rvert_{m_4=13}$ | $\xrightarrow{RNS}$ | | | | | 6 | $\times$ | |
| | | | | | | $6c+8$ | | $\Rightarrow a'_5 = \lvert 0\rvert_{m_4}$ |

Secondly, solve the linear equation of the last MRS coefficient to find $\lvert A\rvert_{m_5=13}$, knowing

that $a'_5$ satisfies $a'_5 = 0$ and $a'_5 = 6c + 8 = 0$, as follows.

$$\begin{aligned}
a'_5 &= 6c + 8 &&= 0 && \text{To keep the same value of } A \\
6c &= |-8|_{13} && && \text{Additive inverse} \\
6c &= 5 && && \\
c &= 5 \cdot |6^{-1}|_{13} && && \text{Multiplicative inverse} \\
c &= |5 \cdot 11|_{13} && && \text{Congruence property} \\
c &= |3|_{13} &&= 3 &&
\end{aligned}$$

Indeed, this is the correct answer where, from Ex. 2.2.8, $A = 107$ and $|A|_{m_5=13} = |107|_{m_5=13} = 3$. We note that the new dynamic space has increased by the factor of $m_{new} = 13$; yet, the value of $A$ is the same.

## 2.5 Reverse conversion

The reverse conversion is the most complicated part of implementing RNS [29]. Generally, there are two main approaches to performing reverse conversion, one of which is Chinese Remainder Theorem (CRT) and the other one is Mixed Radix System (MRS). The latter will be illustrated directly in Ex. 2.5.2, where the former will be introduced below:

### 2.5.1 Chinese remainder theorem

For an integer $C$ represented in $(c_1, c_2, ..., c_n)$ over RNS base $\{m_1, m_2, ..., m_n\}$ with dynamic space $M$, the CRT provides a closed form expression that defines C in the space $M$.

$$|C|_M = \Big| \sum_{i=1}^{n} |c_i \cdot M_i^{-1}|_{m_i} \cdot M_i \Big|_M \tag{2.14}$$

where $M_i = \frac{M}{m_i}$, and $M = \prod_{i=1}^{n} m_i$.

However, the CRT can be used for a reverse conversion procedure; but first a brief explanation of Eq. (2.14) will introduced. The CRT breaks an integer $C$ into sum of the product between a constant $x_i$ and residue of $C$ associated to each modulus $m_i$. Therefore, $C$ can be expressed as follows:

$$C = c_{m_1} \cdot x_1 + c_{m_2} \cdot x_2 + \ \ldots \ + c_{m_n} \cdot x_n$$

such that $|x_j|_{m_i} \equiv 1$ only when $j = i$ and zero otherwise. Such $x_j$ can be formulated as the following:

$$x_j = (\prod_{i=1, i\neq j}^{n} m_i) \cdot \left|(\prod_{i=1, i\neq j}^{n} m_i)^{-1}\right|_{m_j}$$

$$x_j = M_j \cdot |M_j^{-1}|_{m_j}$$

Indeed, both $M_j$ and $|M_j^{-1}|_{m_j}$ depend on fixed moduli set in a specified RNS; therefore, both of them can be computed offline. Furthermore, the multiplication of each $c_{m_i} \cdot x_i$ can be done in parallel. Parallel computation should be followed by a modulus $M$ adder to complete the CRT procedure. This adder is usually large and can adversely affect the reverse conversion's hardware implementation. Below is an example for reverse conversion using CRT.

**Example 2.5.1.** Find the number whose remainders are $(2, 3, 2)$ to the divisors $\{3, 5, 7\}$.

First, we find both $M_j$ and $|M_j^{-1}|_{m_j}$ as follows

$$\begin{aligned}
[m_1 = 3] \Rightarrow M_1 &= 5 \cdot 7 = \ 35 \\
|M_1^{-1}|_3 &= |35|_3 = \ 2 \\
[m_2 = 5] \Rightarrow M_2 &= 3 \cdot 7 = \ 21 \\
|M_2^{-1}|_5 &= |21|_5 = \ 1 \\
[m_3 = 7] \Rightarrow M_3 &= 3 \cdot 5 = \ 15 \\
|M_3^{-1}|_7 &= |15|_7 = \ 1
\end{aligned}$$

Next, we perform $c_{m_i} \cdot x_i$ as follows.

$$C = 2 \cdot (35 \cdot 2) + 3 \cdot (21 \cdot 1) + \ 2 \cdot (15 \cdot 1) = \ 233$$

Lastly, we reduce $C$ mod $M$:

$$|C|_M = |105|_M \equiv 23$$

where $M = 3 \cdot 5 \cdot 7 = 105$

It is important to note that the above last step is necessary to get the desired result in the RNS dynamic range. Therefore, this might increase the complexity for large dynamic range $M$, where costly division is involved.

### 2.5.2 Reverse conversion using MRS

The advantage of reverse conversion using MRS is that it does not exceed the dynamic range $M$ as in CRT. Accordingly, this will eliminate an expensive division by $M$. However reverse conversion using MRS requires an extra step- conversion to the mixed radix equivalent as described in Section 2.3.

**Example 2.5.2.** Recalling Ex. 2.3.1, for $\text{RNS}(85)_{\mathcal{B}} \xrightarrow{RNS} (1, 0, 1)$, with $\mathcal{B} = \{3, 5, 7\}$, the MRS(85) is $(1, 3, 5)$. Thus, the reverse conversion can be performed simply as below:

$$C = 1 \ + 3 \cdot (3) + \ 5 \cdot (3 \cdot 5) = \ 85$$

We note that for a large moduli set the conversion from RNS to decimal is relatively faster using CRT than MRS as it can involve parallel computation.

# Chapter 3

# Homomorphic Encryption

These days, cryptography is being used much more widely than ever before. Cryptography has led to the development of various security functions such as encryption. For a special class of encryption, known as homomorphic encryption, the encrypted data can be directly used for complex operations without being decrypted. Homomorphic encryption is however not supported by traditional cryptosystems and has largely remained impractical to date.

This chapter presents a literature review on homomorphic encryption by highlighting several researchers' contributions to this field. Section 3.1 provides a survey of previous proposed schemes that partially fulfill the homomorphic property; this section also addresses basic homomorphic definitions and properties and presents a literature review of fully homomorphic encryption schemes. Section 3.2 provides a descriptive presentation of Gentry *et al.* in [45] scheme and its variant proposed by Coron *et al.* in [12]. Finally, Section 3.3 presents a ciphertext batch generalization by Coron *et al.*

## 3.1 Homomorphic encryption

Homomorphic encryption (HE) allows a specific operation to be performed directly on ciphertexts and generate another ciphertext that encodes the equivalent operation between plaintexts as follows:

$$Enc(m_a \ \alpha_m \ m_b) \equiv Enc(m_a) \ \alpha_c \ Enc(m_b)$$

where $\alpha_m$ and $\alpha_c$ are operations over message and ciphertext space, respectively. It is worthwhile to indicate that $\alpha_m$ and $\alpha_c$ might not be the same. We say that an encryption is an additive homomorphic encryption if $\alpha_m$ is $\oplus$, and multiplicative homomorphic if $\alpha_m$ is $\times$. In addition, HE scheme would be able to homomorphically evaluate a specific function of degree $L$ over encrypted data up to a certain level. HE is also known as a somewhat homomorphic scheme (SWHE).

Since the proposal of HE initial idea by Rivest *et al.* [35], different homomorphic schemes have been proposed, but they are either insecure or can only support one type of operations (additive or multiplicative) and are known as partially homomorphic schemes. Such an additive homomorphic scheme was proposed by Goldwasser and Silvio [23] in 1982. The Goldwasser scheme was secure, nonetheless inefficient because it encrypted one bit with ciphertext usually of size 1024-bits $[Enc(m_1) \times Enc(m_2) \equiv Enc(m_1 \oplus_m m_2)]$. Later in the probabilistic encryption version [30], Naccache and Stern proposed an additive scheme that enhanced the poor expansion rate in [23]. In 1999, similar to [23], a scheme was proposed by Paillier in [31], and its derivative was proposed by Damgrd, and Jurik in [15]. In addition, existing public key encryption schemes, namely, Rivest, Shamir, and Adleman [36], and Taher Elgamal [16] can perform homomorphic multiplication. Furthermore, schemes as in [2, 5] allowe arbitrary homomorphic additions with limited multiplication depth. It

is worthwhile mentioning that the Sanders, Young and Yung scheme [37] is able to evaluate circuits in NC1[1], with logarithmic depth, at the expense of exponentially expanding ciphertext size with circuit depth, hence, increasing their communication complexity and restricting their circuit to logarithmic depth. The rest of this section provides some basic definitions.

### Basic homomorphic definitions

The following definitions are mostly adopted from Gentry *et al.* [45].

The scheme $\zeta \equiv (KeyGen, Encrypt,\ Eval,\ Decrypt)$ is said to be a correct homomorphic scheme, if $\zeta$ can evaluate any function with arithmetic depth $d$ (namely $E(*)$), which composed of addition and multiplication operations, directly on $n$-tuple ciphertexts $C = (c_1, c_2, \cdots, c_n)$ and we have the equivalent result over a massage space as follows:

$$f(m_1, m_2, ..., m_n) \equiv Decrypt(sk, Eval(pk, E(c_1, c_2, ..., c_n))$$

where $c_i \longleftarrow Encrypt(pk, m_i)$ with a key pair $(sk, pk) \longleftarrow KeyGen(*)$ and massage $m_i$. And $f(*)$ is the equivalent function of $E(*)$ over the message space.

Furthermore, the homomorphic scheme $\zeta$ is said to be a compact scheme, if the size of $c_{out-Eval} \longleftarrow Eval(pk, E(c_1, c_2, ..., c_n))$ is within defined bounds, regardless of the size or the depth of $Eval(pk, *)$. The compactness property controls the practicality and the security of the scheme; for instance uncompact homomorphically multiplicative scheme can reveal the depth of multiplication by analyzing the size of $Eval(pk, *)$'s output.

Most likely $Eval(pk, *)$ is publicly known; thus the security proof of $\zeta$ scheme does not involve the evaluation function. However, it is essential to security requirements that the

[1]NCi is the class of decision problems, s.t. $NC1 \subseteq NC2... \subseteq NC$, computed by boolean circuits of polynomial number of gates of at most two inputs, and depth $O(\log^i n)$

output of $Eval(pk, *)$ does not reveal the inputs $(*)$ even for third party who has the secret key $sk$.

### 3.1.1 Fully homomorphic encryption (FHE)

An encryption scheme is said to be fully homomorphic when it implicitly allows evaluating an arbitrary Boolean circuit over plaintexts by only manipulating the ciphertexts.

In 2009, Gentry [20], in his pioneering work, constructed the first fully homomorphic encryption scheme. The scheme's security was based on ideal lattices worst-case problems [21] along with a sparse subset-sum problem. First, he constructed a HE scheme with limited multiplication operations namely, a somewhat homomorphic scheme (SWHE). In SWHE, the ciphertext is constructed with inner noise that grows linearly with addition and exponentially with multiplication, which limits SWHE's operation depth up to a certain bound. In order to achieve the FHE scheme, Gentry's second step was to squash the decryption circuit so it is transformed to a low degree polynomial. Finally, Gentry's breakthrough was refreshing the ciphertext noise using bootstrapping procedure. The procedure homomorphically evaluates the low degree decryption circuit on the encryption of the secret key. The result is refreshed ciphertext, which encrypts the original plaintext under different distribution with reduced noise. Therefore, the SWHE scheme with limited homomorphic operations is transformed into a fully homomorphic scheme with unlimited homomorphic operations.

Besides their aggressive analysis on the hardness assumptions of [20], Stehl and Steinfeld [40] described two optimization techniques for Gentry's scheme. Their first optimization technique was proposed to reduce the number of vectors in the sparse subset-sum problem; their second one was to reduce the degree of the decryption polynomial. The authors

of [39] suggested several optimization techniques to implement Gentry's original scheme with relatively small key and ciphertext size; yet, the implementation did not achieve the bootstrapping functionality that is needed to have a FHE scheme. However, using the optimization techniques in [39] and the first optimization in [40] along with other techniques, Gentry and Halevi in [22] successively implemented the original Gentry scheme.

Since Gentry's blueprint, fully homomorphic encryption may be divided into three main groups based on their security assumptions

- Fully homomorphic encryption based on ideal lattices proposed by Gentry in [20]: The earlier implementation was due to Smart and Vercauteren [39]. They were able to implement the SWHE of Gentry's original scheme; but unfortunately, their implementation was not bootstrappable. Employing the suggestion of [39] with technical optimization, the authors of [22] introduced the first fully implementation of [20] but without any enhancement in the performance.

- Fully homomorphic encryption based on *Learning with error* (LWE) problem proposed by Barkerski and Vaikuntananthan in [8]: The scheme security is based on the worst-case hardness of short vector problems on arbitrary lattices. The authors also presented a new dimension-modulus reduction technique that shortens the ciphertext, which allows [8] to achieve a bootstrapping functionality.

- Fully homomorphic encryption over integers proposed by Dijk, Gentry, Halevi, and Vaikuntanathan [45]: The DGHV security is based on the hardness of the approximate greatest common divisors problem (A-GCD).

## 3.2 Fully homomorphic encryption over integers

Following Gentry's footsteps in [20], DGHV first constructed SWHE using modulo algebraic arithmetic, which has limited operation depth. Then they squashed the decryption circuit and realized bootstrapping to achieve a fully homomorphic encryption scheme by just operating over integers instead of complex ideal lattices. In general, DGHV is conceptually simpler than Gentry's original scheme [20]. This simplicity comes at the expense of large public key size (around $O(\lambda^{10})$ to avoid lattice attacks for a security parameter $\lambda$), making the DGHV scheme far from practical. In order to reduce DGHV complexity, an attempt by Coron *et al.* [13] reduced the public key size to$O(\lambda^{7})$. In addition, and for similar security parameters of [13], Coron *et al.* [14] were able to obtain 10.1 MB public key instead of 802 MB for the full implementation.

However, DGHV requires a bootstrapping procedure after each homomorphic multiplication which is quite expensive. Thus, alternative techniques (so-caled modulus switching technique (MS)) were proposed in [6] and implemented for DGHV in [14]. In addition, a scale-invariant to DGHV was introduced in [12] (SI-DGHV) where the authors adopted some optimization techniques from [13], and public key compression and modulus switching technique from [14]. They also introduced an optimization to the MS scalar product to improve SI-DGHV performance; yet, SI-DGHV over integers is still far from practical, since encryption, decryption and MS take about 5min, 24s and 4.37min, respectively.

As our work focuses on FHE over integers, we provide a bit detailed overview of DGHV [45] and its variant in [12]. First, we will follow Gentry's footsteps for constructing a FHE, and provide a general description of the SWHE scheme with some examples for both [45] and [12]. Secondly and toward fully homomorphism, two techniques, namely bootstrapping and modulus switching, are introduced where the former is described in brief and the latter

in more details.

### 3.2.1 DGHV over integers

This section first provides a brief description of the somewhat homomorphic encryption DGHV (SMHE-DGHV) scheme. For this, first we introduce some notations: for a real number $x$, we denote $\lfloor x \rfloor$, $\lceil x \rceil$ and $\lceil x \rfloor$ to be the lower, upper and nearest integer to $x$, respectively; moreover, for two integers $a$ and $b$, we denote $|a|_b$ to be a residue modulo $b$ $\in (-b/2,\ b/2)$. However, for a security parameter $\lambda$ the construction of DGHV is described as follow:

**DGHV.KeyGen($\lambda$):** Set the secret key to be a random prime integer $p$ of size $\eta$, and let the distribution over $\gamma$ and $\rho$ bits integer to be:

$$\mathcal{D}_{\gamma,\rho}(p) = \Big\{\text{choose } q \in \mathbb{Z} \cap [0,\ 2^{\gamma}/p),\ r \in \mathbb{Z} \cap (-2^{\rho},\ 2^{\rho}) : \text{Outputs } x = pq + r \Big\}$$

To obtain public key elements, sample $x_i \longleftarrow \mathcal{D}_{\gamma,\rho}(p)$ for $i = 0, 1, ...,\ \tau$, then re-label them, so that $x_0$ would be the largest. Restart unless $x_0$ is odd and $[x_0]_p$ is even. Thus public key elements are $pk = \{x_0,\ x_1,\ ...,\ x_\tau\}$.

**DGHV.Enc($pk,\ m \in \{0,\ 1\}$):** Choose a random subset S $\subseteq \{1, 2, 3, ..., \tau\}$, and a random integer $r \in (-2^{\rho},\ 2^{\rho})$, output:

$$c = [\ 2 \sum_{i \in S} x_i + 2r + m\ ]_{x_0} \tag{3.1}$$

**DGHV.Evaluate($pk,\ C,\ c_1,\ c_2, ...,\ c_t$):** For a circuit $C$ with $t$ input bits and $t$ ciphertexts, apply the addition and multiplication gates of $C$ to ciphertexts, by performing all addition and multiplication over the integers, and then output the resultant integer.

**DGHV.Dec($sk,\ c$):** Output $m = (\ |c|_p\ ) \bmod 2$

SWHE scheme has limited number of homomorphic operations (known as depth) because the ciphertext's inner noise grows with respect to the depth and operation type (addition or multiplication). For illustration purposes, simpler version of Eq. (3.1) is utilized to analysis the noise growth. The simplified ciphertext would then be equivalent to the following:

$$c = p \cdot q + 2r + m$$

where the same **DGHV.Dec(*,*)** function can be used to recover the message bit $m$. Now, given two ciphertexts $c_1$ and $c_2$, the addition over the integers can be performed as follows:

$$\begin{aligned} c_1 + c_2 &= (pq_1 + 2r_1 + m_1) + (pq_2 + 2r_2 + m_2) \\ c_3 = c_1 + c_2 &= p \cdot (q_1 + q_2) + 2 \cdot (r_1 + r_2) + (m_1 + m_2) \end{aligned}$$

where $m_3 = m_1 + m_2,\ r_3 = r_1 + r_2$; hence the noise grows at most by $\rho + 1$ bits.

The multiplication operation over the integers can also be performed as follows:

$$\begin{aligned} c_1 \cdot c_2 = \ c_4 = \ & p^2 \cdot (q_1 q_2) + p \cdot (2q_1 r_2 + 2q_2 r_1 + q_1 m_2 + q_2 m_1) \\ & +4 \cdot (r_1 r_2) + 2 \cdot (r_1 m_2 + r_2 m_1) + (m_1 m_2) \\ c_4 = \ & p \cdot q_4 + 2r_4 + m_4 \end{aligned}$$

where $m_4 = m_1 m_2$, $r_4 = 2r_1 r_2 + r_1 m_2 + r_2 m_1$ and $q_4 = p \cdot (q_1 q_2) + 2q_1 r_2 + 2q_2 r_1 + q_1 m_2 + q_2 m_1$.

we note that the size of noise growth is $2\rho$ bits. However, in order to perform correct decryption the size of the noise should not exceed the modulo size (secret key $p$), thus we need:

$$\begin{aligned} -p/4 < \ r_3 = \ & r_1 + r_2 & < \ p/4 \quad & \text{for addition} \\ -p/4 < \ r_4 = \ & 2r_1 r_2 + r_1 m_2 + r_2 m_1 & < \ p/4 \quad & \text{for multiplication} \end{aligned} \tag{3.2}$$

For the addition operation, it is obvious that the noise grows linearly with the addition depth; for instance, the noise growth for the summation of $d$ fresh[2] ciphertexts would be at most of $(\rho + d + 1)$ bits. However, the multiplication dominates the noise growth, where it can say that the noise for multiply $d$ fresh ciphertexts would be $d \cdot (\rho+1)$ bits. Therefor, the SWHE scheme can perform at most $(\eta - 2)/(\rho + 1)$ successful multiplication on ciphertext.

**Example 3.2.1.** For the hypothetical parameters $\gamma$ = 18-bits, $\eta$ = 12-bits, $\rho$ = 4-bits, and $\tau = 4$, encrypt both $m_1$=1 and $m_2$=0, and determine homomorphically $m_1 \oplus m_2$ and $m_1 \otimes m_2$ using DGHV scheme.

First, we generate the private and the public keys

$$\begin{aligned}
DGHV.KeyGen(\lambda = 2) &\Rightarrow p = 4013 \\
&\Rightarrow x_0 = 256846 \longleftarrow r = 14, \quad q = 64 \\
&\Rightarrow x_1 = 232757 \longleftarrow r = 3, \quad q = 58 \\
&\Rightarrow x_2 = 248796 \longleftarrow r = -10, \quad q = 62 \\
&\Rightarrow x_3 = 224714 \longleftarrow r = -14, \quad q = 56 \\
&\Rightarrow x_4 = 240783 \longleftarrow r = 3, \quad q = 60
\end{aligned}$$

Note that, $p$ is kept secret and $pk$ is $[x_0, x_1, x_2, x_3, x_4]$=[256846,248796, 240783, 232757, 224714], with $x_0$ be the largest one s.t. $|x_0|_2 = 1$ and $||x_0|_p|_2 = 0$.

Secondly, obtain the encryption of both $m_1$ and $m_2$: $c_1 = DGHV.Enc(pk, m_1 = 1)$ and $c_2 = DGHV.Enc(pk, m_2 = 0)$

$$\begin{aligned}
c_1 &= [\ 2\ (232757 + 224714) + 2(-11) + 1\ ]_{256846} = 208667 \quad \text{for } S \in \{1, 3\} \\
c_2 &= [\ 2\ (224714 + 240783) + 2(7) + 0\ ]_{256846} = 160560 \quad \text{for } S \in \{3, 4\}
\end{aligned}$$

[2]fresh implies that cipher with noise size $\rho$ bits.

Thirdly, evaluate both $m_1 \oplus m_2$ and $m_1 \otimes m_2$: the addition and the multiplication of $c_1$ and $c_2$ are

$$\begin{aligned} c_3 &= c_1 + c_2 = 208667 + 160560 = 369227 \\ c_4 &= c_1 \cdot c_2 = 208667 \cdot 160560 = 33503573520 \end{aligned}$$

Finally, in order to check the correctness of the example scheme setting, we will decrypt both ciphertexts $c_3$ and $c_4$, but first we note that we have to use Eq. (2.1) to perform $|369227|_{sk}$

$$\begin{aligned} DGHV.Dec(sk = 4013, c_3) &\Rightarrow |(|369227|_{sk})|_2 = 1 \equiv m_1 + m_2 \\ DGHV.Dec(sk = 4013, c_4) &\Rightarrow |(|33503573520|_{sk})|_2 = 0 \equiv m_1 \cdot m_2 \end{aligned}$$

which is correct.

Indeed, the question now is how many multiplications our SWHE-DGHV can perform correctly. Let us naively analyze the noise bound to obtain the depth of multiplications. Recalling Eq. (2.1) which its result would be $\in (-p/2, p/2)$, and knowing that $p \in [2^{\eta-1}, 2^{\eta})$, thus the multiplication depth d using Eq. (3.2) is:

$$2^{(\rho+1)\cdot d} < 2^{\eta-3} < p/4 < 2^{\eta-2}$$

$$d < \frac{\eta - 2}{\rho + 1}$$

In our example $d < \frac{12-2}{4+1} = 2$. Then for correct decryption we have less than two multiplications. Actually, Gentry *et al.* [45] also considered the bootstrapping procedure in their noise bound and they had:

$$d \leqslant \frac{\eta - 4 - \log |f^{\rightharpoonup}|}{\rho + 2}$$

where $|f^{\rightharpoonup}|$ is the norm of the multivariate polynomial $f(x_1; x_2; ...; x_t)$ computed by $C^*$. $C^*$ is the associated integer circuit of a Boolean circuit $C$ with $t$ inputs (see [45]).

### Security of SWHE-DGHV

DGHV scheme's security is based on the hardness of the Approximate-GCD problem that is defined below:

**Definition 3.2.1.** (Approximate-GCD) The $(\rho, \eta, \gamma)$-Approximate-GCD problem is: given polynomially many samples from $\mathcal{D}_{\gamma,\rho}(p)$ for a random odd integer $p$ of $\eta$ bits, determine $p$, where $\mathcal{D}_{\gamma,\rho}(p)$, which is a distribution over $\eta$ bits integer, defined as

$$\mathcal{D}_{\gamma,\rho}(p) = \Big\{ choose\ q \in \mathbb{Z} \cap [0,\ 2^{\gamma}/p),\ r \in \mathbb{Z} \cap (-2^{\rho},\ 2^{\rho})\ : Outputs\ x = pq + r \Big\}$$

Thus Approximate-GCD simply tries to guess the common near divisor, namely $sk = p$, from the list of public keys $x_0, x_1, ..., x_\tau$, where $x_i = \mathcal{D}_{\gamma,\rho}(p)$ . However, this section discuss known attacks on just two public elements. Considering Approximate-GCD samples $\{x_0, x_1, ..., x_t\}$, there is a known brute-force attack that approximates the great common divisor of two samples $x_i$ and $x_j$, by guessing $r_i$ and $r_j \in (-2^{\rho}, 2^{\rho})$. The attacker in each iteration applies $p^* \leftarrow gcd(x_i, x_j)$, and considering $p^*$ as a solution if its bit-size equals to $\eta$. Accordingly, the attack running time is approximately $2^{2\rho}$. Using the Stehle-Zimmermann algorithm [41] to compute the GCD with time complexity $O(\gamma)$, the overall brute-force attack complexity would be $2^{2\rho} \cdot O(\gamma)$, and accordingly, the size of the noise $\rho$ would be $\omega(\log \lambda)$.

Moreover, there is a variant of the brute-force attack based on factoring $x_i$ to determine $\eta$ bit factor $p^*$, and then consider $p^*$ to be a solution if $p^*$ is a divisor of $x_j$. The authors of [45] show that using the Lenstras elliptic curve factoring algorithm [27], the attack complexity is approximately $2^{\rho+\sqrt{\eta}}$. Lattice-based attacks on Approximate-GCD, such as the attack proposed by Howgrave-Graham (see [26]), the attacker has an advantage when $(\rho/\gamma) < (\eta/\gamma)^2$. Therefore, $\gamma$ should satisfy $\omega(\eta^2 \log(\lambda))$.

In general, the SWHE scheme's parameters must meet some constraints to avoid known attacks such as the Howgrave-Graham attack. For a security parameter $\lambda$, the scheme's parameters are listed in the follow:

- $\rho = \omega(\log \lambda)$, to avoid brute-force attacks on the noise
- $\eta \geq \rho \cdot \Theta(\lambda \log^2 \lambda)$, to support homomorphic operation for a certain depth.
- $\gamma = \omega(\eta^2 \log \lambda)$, to thwart various lattice-based attacks on the underlying Approximate GCD problem.
- $\tau \geq \gamma + \omega(\log \lambda)$, to apply the leftover hash Lemma 3.2.1 in the reduction of Approximate-GCD.

In order to satisfy the above constraints, the SWHE scheme's parameter can be taken as $\rho = \lambda$, $\eta = O(\lambda^2)$, $\gamma = O(\lambda^5)$ and $\tau = \gamma + \lambda$ [45], which result in a scheme with complexity $O(\lambda^{10})$.

**Accomplishing fully homomorphic DGHV scheme**

In order to transform the SWHE-DGHV scheme to be fully homomorphic scheme, the authors of [45] followed Gentry's blueprint in [20] by homomorphically computing the decryption equation $m^* \Longleftarrow [c - \lfloor c/p \rceil]_2$. Unfortunately, SWHE was incapable of evaluating their decryption circuit because the decryption circuit required a Boolean circuit with depth more than SWHE can reach. Thus, squashing the decryption circuit is required to allow and make SWHE bootstrappable. Consequently, this requires adding extra elements to make the scheme bootstrappable. The extra elements assemble the secret key in the form of a large set of secret sparse subsets that sums to the original secret key ($sk = p$);

here, the security relies on a sparse subset-sum problem. The procedure is briefly described in the following:

$$m^* \Longleftarrow [c^* \ - \ \lfloor \sum_{i \in S} s_i z_i \rceil]_2$$

where $s_i \in \{0, 1\}$ and $z_i \Longleftarrow c^* \cdot y_i \pmod 2$, condition to $sk$ sparse $1/p \pmod 2 \Longleftarrow \sum_{i \in S} y_i$ such that $\lfloor \sum_{i \in S} s_i z_i \rceil \pmod 2 \Rightarrow \lfloor c/p \rceil$, (see [45] for full description).

Further, the idea of bootstrapping is to run the decryption circuit homomorphically on the noisy ciphertext using an encrypted secret key (given the public key), to obtain fresh ciphertext with less noise and different distribution that encrypts the same message $m$. However, bootstrapping is quite expensive. Thus an alternative technique was proposed in [6] and implemented for the DGHV in [14]. The new technique, a.k.a. modulus switching technique, is used to reduce the noise by transforming the ciphertext's encryption key from $sk_1$ into a smaller key $sk_2$. It is worthwhile to pointout that the noise is scaled by a factor of $sk_2/sk_1$. Consequently and for d multiplication depth, the modulus switching technique requires a storage of $d$ secret keys $(sk_i)_{1 \leq i \leq d}$, which will increase the storage requirement. Coron *et al.* [12] proposed a clever trick that is used to transform a ciphertext's encryption key from $p^2$ into a smaller key $p$, thus scale the noise by $1/p$. The new modulus switching procedure allows SWHE to be leveled FHE scheme whose noise size grows linearly with the multiplications depth. The following section presents full description of the Coron *et al.* procedure.

### 3.2.2 Homomorphic encryption scheme of Coron *et al.*

In [12], Coron *et al.* proposed a variant of the DGHV scheme (denoted as SI-DGHV) which is considered Leveled FHE scheme as described later in this section. However, the

SI-DGHV scheme works over mod $p^2$ instead of mod $p$ as in the original DGHV scheme. The scheme encrypts a message $m \in \{0, 1\}$ as the most significant bit of the ciphertext $c \pmod p$, rather than the least significant bit (as in the DGHV). Altogether, SWHE SI-DGHV ciphertext is formulated as follows:

$$c = q\ p^2 + \frac{p-1}{2}\ (m + 2r^*) + r \tag{3.3}$$

where $\log_2(c) \leq \gamma$, $q \in [0,\ 2^\gamma/p^2) \cap \mathbb{Z}$, and $r$ and $r^*$ of size $\rho$ and $\rho^*$ bits, respectively.

In addition, the ciphertext in Eq. (3.3) is denoted as a type-I ciphertext that can be decrypted using $m \longleftarrow (|2 \cdot c|_p) \bmod 2$. SWHE SI-DGHV ciphertext is homomorphic scheme, such that, for two given ciphertexts $c_1$ and $c_2$ which encrypts $m_1$ and $m_2$, respectively.

$$c_1 = q_1\ p^2 + \frac{p-1}{2}\ (m_1 + 2r_1^*) + r_1$$
$$c_2 = q_2\ p^2 + \frac{p-1}{2}\ (m_2 + 2r_2^*) + r_2$$

The addition of $c_1$ and $c_2$ is equivalent to the encryption of $|\ m_1 + m_2\ |_2$ as follows:

$$c_1 + c_2 \ = \ (q_1 + q_2)\ p^2 + \tfrac{p-1}{2}\ (\ (m_1 + m_2) + 2r_3^*) + (r_1 + r_2)$$

where $r_3$ and $r_3^*$ of size $\rho + 1$, and $\rho^* + 1$, respectively.

The multiplication of $c_1$ and $c_2$ is equivalent to the encryption of $|\ m_1 \cdot m_2\ |_2$ as follows:

$$\begin{aligned} c_p \ &= \ 2c_1c_2 \ = \ (q_p^*)\ p^2 + \tfrac{(p-1)^2}{2}\ (m_1 + 2r_1^*)(m_2 + 2r_2^*) + r_p^* \\ c_p \ &= \ (q_p^*)\ p^2 + \tfrac{p^2-1}{2}\ (m_1\ m_2) + r_p \end{aligned} \tag{3.4}$$

where $c_4$ is known as type-II ciphertext with noise $r_p$ of size $\leq \eta + \rho + \rho^* + 4$ .

In contrary to the addition operation where the noise growth linearly with addition depth; the multiplication operation transforms a ciphertext type-I to type-II whose noise is greater than modulus $p$ and less than modulus $p^2$.

### Making the SWHE scheme fully homomorphic

The author of [12] used a **Convert** procedure to convert type-II ciphertext modulo $p^2$ back to t type-I ciphertex by scaling down the product of the ciphertext $c_p$. The concept of **Convert** procedure is similar to modulus switching technique in [14] where instead of switching ciphertext with modulo $p$ to another with modulo $p^*$, and accordingly scaling the noise by $p^*/p$; the **Convert** procedure will transform a ciphertext with modulo $p^2$ to another with modulo $p$ to scale the noise by $p/p^2$.

It is important to address the fact that the factor $p/p^2$ should remain secret. Therefore, $\frac{2^\eta}{p^2}$ is hidden in sparse subset-sum problem, and $\frac{p}{2^{\eta+1}}$ encrypted in cipher type-I. The following provides a description of the **Convert** procedure:

First, secure $\frac{2^\eta}{p^2}$ by generating sparse subset-sum problem s.t.

$$\frac{2^\eta}{p^2} = \langle s, z \rangle + \varepsilon \bmod 2^\eta \quad with\ \varepsilon \leq 2^{-k} \tag{3.5}$$

where $\mathbf{s}$ is a vector of $\Theta$ bits, kept secret and encrypted in Eq. (3.6). $\mathbf{z}$ is a vector of $\Theta$ rational numbers, $\in [0, 2^\eta)$, of $k$ precision after binary point.

In addition, different ways are proposed to generate the vector $\mathbf{z}$, naive one proposed by [9], is by randomly select $z_i \in [0, 2^\eta)_{2 \leq i \leq \Theta}$, then set $z_1$ that gives $\sum_{i \in S} s_i\ z_i$ for a random set $\mathbf{s}$. Other approach proposed as practical implementation is used in [12]. However, the secret vector $\mathbf{s}$ along with $\frac{p}{2^{\eta+1}}$ have to be secured as in the following step.

Secondly, encrypt the secrets $\mathbf{s}$ and $\frac{p}{2^{\eta+1}}$ in $\Theta$ element vector $\boldsymbol{\sigma}$.

$$\boldsymbol{\sigma} = \boldsymbol{q}\ p^2 + \boldsymbol{r} + \left\lfloor \mathbf{s} \cdot \frac{p}{2^{\eta+1}} \right\rceil \tag{3.6}$$

where $\boldsymbol{q} \longleftarrow (\mathbb{Z} \cap [0,\ q_0))^\Theta$ and $\boldsymbol{r} \longleftarrow (\mathbb{Z} \cap (-2^\rho,\ 2^\rho))^\Theta$ are integers.

Then we have

$$c' = 2\langle \boldsymbol{\sigma}, \boldsymbol{c} \rangle \Longrightarrow 2 \cdot p^2 \cdot \langle \boldsymbol{q}, \boldsymbol{c} \rangle + 2\langle \boldsymbol{r}, \boldsymbol{c} \rangle + \frac{2 \cdot p}{2^{\eta+1}} \langle \boldsymbol{c}, \mathbf{s} \rangle$$

where $\boldsymbol{c} = (|\lfloor c \cdot z_i \rceil|_{2^\eta})_{1 \leq i \leq \Theta}$.

Even though $\frac{2p}{2^{\eta+1}} \cdot \langle \boldsymbol{c}, \mathbf{s} \rangle$ should scale down the ciphertext by $1/p$ which reduces the inner cipher $c$ noise, the multiplication itself adds noise formed in $2\langle \boldsymbol{r}, \boldsymbol{c} \rangle$ as well as expands the size of the ciphertext formed in $2p^2 \cdot \langle \boldsymbol{q}, \boldsymbol{c} \rangle$.

In order to reduce the noise and compact the ciphertext, the authors of [12] used $BitDecomp_\eta(*)$ and $PowerOfTwo_\eta(*)$ procedures defined in terms of $a$ and $b \in \mathbb{Z}^\Theta$ as follows

$$\begin{aligned} BitDecomp_\eta(a) &\Rightarrow (a_0, a_1, a_2, \cdots a_{\eta-1}) \in \{0,1\}^{\Theta\eta} \text{ with } a_i \in \{0,1\}^\Theta \\ PowerOfTwo_\eta(b) &\Rightarrow (b, 2b, 2^2 b, \cdots, 2^{\eta-1} b) \in \mathbb{Z}^{\Theta\eta} \end{aligned}$$

The approach squashes the multiplicand $\boldsymbol{c} = (|\lfloor c \cdot z_i \rceil|_{2^\eta})_{1 \leq i \leq \Theta}$ in the $(\Theta \times \eta)$ matrix using $\mathbf{c}^* \longleftarrow BitDecomp_\eta(\mathbf{c})$ and expands $\mathbf{s}$ using $PowerOfTwo_\eta(\mathbf{s})$.

The **convert** procedure requires extra public key elements, namely the vector $\mathbf{z}$ and $\boldsymbol{\sigma}$ that is defined as follows:

$$\boldsymbol{\sigma} = \boldsymbol{q}\ p^2 + \boldsymbol{r} + \left\lfloor PowerOfTwo_\eta(s) \cdot \frac{p}{2^{\eta+1}} \right\rceil$$

Finally, transform ciphertext type-II to type-I as follows:

$$c' = 2\langle \boldsymbol{\sigma}, \boldsymbol{c}^* \rangle = 2 \cdot p^2 \cdot \langle \boldsymbol{q}, \boldsymbol{c}^* \rangle + 2\langle \boldsymbol{r}, \boldsymbol{c}^* \rangle + \frac{p}{2^\eta} \langle \boldsymbol{c}^*, \mathbf{s} \rangle$$

The output is type-I ciphertext with noise of size $\rho' = \rho + \log_2 \Theta + 9$ and $\rho^{*\prime} = \log_2 \Theta$; thus the noise grows linearly with multiplication operation and by additive factor of $\log_2 \Theta +$ 9 bits.

To sum up, the scheme's full description for a security parameter $\lambda$ is outlined as follows:

**SI-DGHV.KeyGen($\lambda$):** Generate odd $\eta$ bit integer $p$ and $\gamma$ bits integer $x_0 = q_0\ p^2 + r_0$ with $r_0 \longleftarrow \mathbb{Z} \cap (-2^\rho,\ 2^\rho)$ and $q_0 \longleftarrow \mathbb{Z} \cap [0,\ 2^\gamma/p^2)$; thus, for the defined distribution over $\eta$ bit integers:

$$\mathcal{D}^{\rho}_{p,q_0} = \Big\{ choose\ q \in \mathbb{Z} \cap [0,\ q_0),\ r \in \mathbb{Z} \cap (-2^\rho,\ 2^\rho)\ : Outputs\ x = q\ p^2 + r \Big\}$$

- Sample $x_i \longleftarrow \mathcal{D}^{\rho}_{p,q_0}$ for $i = 0, 1, ...,\ \tau$ of size $\gamma$.
- Compute $y = y^*\ + (p-1)/2$ with $y^* \longleftarrow \mathcal{D}^{\rho}_{p,q_0}$.
- Let $\mathbf{z}$ be vector of $\Theta$ elements, with $k = 2\gamma + 2$ bits of precision after binary point, and $\mathbf{s}$ is a vector of $\Theta$ bits s.t.

  $$\frac{2^\eta}{p^2} = \langle s, z \rangle + \varepsilon \bmod 2^\eta$$

  with $\varepsilon\ \leq\ 2^{-k}$.
- Finally compute

  $$\boldsymbol{\sigma} = \boldsymbol{q}\ p^2 + \boldsymbol{r} + \Big\lfloor\ PowerOfTwo_\eta(s)\ \cdot \frac{p}{2^{\eta+1}} \Big\rceil$$

  where $\boldsymbol{q}$ and $\boldsymbol{r}$ are random vectors chosen from $(\mathbb{Z} \cap [0,\ q_0))^{\eta\Theta}$, $(\mathbb{Z} \cap (-2^\rho,\ 2^\rho))^{\eta\Theta}$, respectively.

Output the secret key $sk = \{p\}$, and the public key $pk = \{x_0, x_1, ..., x_\tau, y, \boldsymbol{\sigma}, \boldsymbol{z}\}$.

**SI-DGHV.Enc($pk,\ m \in \{0,\ 1\}$):** Choose a random subset S $\subseteq \{1, 2, 3, ..., \tau\}$ and output:

$$c \longleftarrow [\ \sum_{i \in S} x_i + m\ y]_{x_0} \tag{3.7}$$

**SI-DGHV.Evaluate($pk,\ c_1,\ c_2$):** The evaluation function performs addition, multiplication and conversion from type-II to type-I operation on ciphertext $c$.

- *SI-DGHV.Add*$(pk, c_1, c_2)$. Output $c_3 = |c_1 + c_2|_{x_0}$
- *SI-DGHV.Convert*$(pk, c)$. Output $c^* = 2\ \langle \boldsymbol{\sigma}, BitDecomp_\eta(\boldsymbol{c})\rangle$. with $\boldsymbol{c} = (|\lfloor c \cdot z_i \rceil|_{2^\eta})_{1\leq i\leq \Theta}$.
- *SI-DGHV.Mult*$(pk, c_1, c_2)$. Output $c \longleftarrow |\ SI\text{-}DGHV.Convert(pk, 2\ c_1 \cdot c_2)\ |_{x_0}$

**SI-DGHV.Dec**$(sk,\ c\})$**:** Output $m \longleftarrow (|2c|_p) \bmod 2$

**Example 3.2.2.** Let us consider a toy example of encrypting $m_1 = 1$ and $m_2 = 1$ and homomorphically evaluating $m_1 \oplus m_2$ and $m_1 \cdot m_2$ : For given security parameter $\lambda = 2$ and $\eta = 14$ $\gamma = 32,\ \rho = 5,\ \tau = 5,\ \Theta = 6$, the implementation of SIDGHV FHE can be formed as follows:

$$
\begin{array}{rcl}
SI\text{-}DGHV.KeyGen(\lambda = 2) & \Rightarrow & p = 13127 \\
q_0p^2 + r & \Rightarrow & x_0 = 4135635100 \longleftarrow r_0 = 4, \quad q_0 = 24 \\
\mathcal{D}^\rho_{p,q_0} & \Rightarrow & x_1 = 1895499415 \longleftarrow r = -4, \quad q = 11 \\
\mathcal{D}^\rho_{p,q_0} & \Rightarrow & x_2 = 861590644 \longleftarrow r = -1, \quad q = 5 \\
\mathcal{D}^\rho_{p,q_0} & \Rightarrow & x_3 = 2584771933 \longleftarrow r = -2, \quad q = 15 \\
\mathcal{D}^\rho_{p,q_0} & \Rightarrow & x_4 = 2584771937 \longleftarrow r = 2, \quad q = 15 \\
\mathcal{D}^\rho_{p,q_0} & \Rightarrow & x_5 = 1550863158 \longleftarrow r = -3, \quad q = 9 \\
\mathcal{D}^\rho_{p,q_0} + \frac{p-1}{2} & \Rightarrow & y = 3963323525 \longleftarrow r = -5, \quad q = 23
\end{array}
$$

Randomly generate the public vector $\mathbf{z}$ of size $\Theta - 1$ and then randomly choose the secret subset $S_z \subseteq \{1, 2, ..., \Theta - 1\}$. Lastly, append $z_{\Theta - i}$ that satisfies Eq. (3.5) where $i$ is selected randomly, and accordingly, set the secret vector $\mathbf{s}$. Thus we have the following:

$$
\begin{aligned}
\boldsymbol{z} &= [2.9431247067265853e^{-05}, 3.0096133401769087e^{-05}, 1.772869756397344e^{-05}, \\
&\quad 2.5342163695651898e^{-05}, 2.7395711602901107e^{-05}, 2.0524299050181383e^{-05}] \\
\boldsymbol{s} &= [1, 0, 1, 0, 1, 1]
\end{aligned}
$$

and accordingly, using Eq. (3.2.2), we get:

$$\boldsymbol{\sigma} = \begin{bmatrix} 861590641 & 1206226906 & ... & \\ 2929408194 & ... & & ... \\ ... & & ... & 2929411474 \\ & ... & 172319767 & 2929411471 \end{bmatrix}_{(\Theta,\eta)=(4,14)}$$

Thus the secret key $p = 13127$, and the public key $\longleftarrow \{x_0, x_1, x_3, x_4, x_5, y, \boldsymbol{z}, \boldsymbol{\sigma}\}$.

Using public key elements, the ciphertext $c_1$ and $c_2$ can be obtained as follows:

$$\begin{aligned} c_1 &= \textit{SI-DGHV.Enc}(pk, m_1) &&= [(861590644 + 2584771937) + 3963323525]_{x_0} \\ &&&= 3274051006 \\ c_2 &= \textit{SI-DGHV.Enc}(pk, m_2) &&= [(1895499415 + 1550863158) + 3963323525]_{x_0} \\ &&&= 3274050998 \end{aligned}$$

where $S_{c_1} = \{2, 4\}$ and $S_{c_2} = \{1, 5\}$.

The example first evolution $(m_1 \oplus m_2)$ can be homomorphically made over the integer as follows:

$$\begin{aligned} c_1 + c_2 &= 3274051006 + 3274050998 &&= 6548102004 \\ &= [6548102004]_{x_0=4135635100} &&= 2412466904 \quad \text{for compactness} \end{aligned}$$

Let us now decrepit $(c_1 + c_2)$ in order to check the correctness of the example SI-DGHV setting. We have $|\ (\ |2412466904|_{13127}\ )\ |_2 = 0$, which implies that $|\ (\ |c_1 + c_2|_{13127})\ |_2 \equiv m_1 \oplus m_2 = 0$.

The example second evaluation $(m_1 \cdot m_2)$ can also be performed over integers as follows:

$$c_4 = 2\ c_1 \times c_2 = 2 \times 3274051006 \times 3274050998 = 10719409963697203988$$

Indeed, $c_4$ is type-II ciphertext that is needed to be converted back to type-I ciphertext. Therefore, compute $\boldsymbol{c} = (|\lfloor c \times z_i \rceil|_{2^\eta})_{1 \leq i \leq \Theta} = [4740, 7232, 12236, 693, 12636, 11157]$, then $\boldsymbol{c}^* = BitDecomp_{14}(\boldsymbol{c})$ and the recovered type-I ciphertext is

$$c_p^* = \ 2\langle \boldsymbol{c}^*, \boldsymbol{\sigma} \rangle \ = \ 183346508674$$

In fact, the type-I ciphertext $c_4^*$ is 44 times $x_0$ (s.t. $c_4^* = 44x_0 + 1378564274$); thus to preserve the ciphertext compactness compute $c_p = |c_4^*|_{x_0} = 1378564274$.

*Remark.* This very last step is important at it provides compactness to the SI-DGHV scheme as well erase any trace of the SI-DGHV.Evaluate($pk,\ c_1,\ c_2$) procedure.

To check the correctness of SI-DGHV homomorphic multiplication, let us decrepit $c_p$: $||1378564274|_{13127}|_2 = 1$, which implies that $||c_1 \times c_2|_{13127}|_2 \equiv m_1 \times m_2 = 1$.

### Security of SI-DGHV scheme

Unlike the DGHV scheme, the security of the SI-DGHV scheme is based on a decisional version of Approximate-GCD problem 3.2.1. The Decisional Approximate-GCD problem is defined as follows:

**Definition 3.2.2.** (Decisional Approximate-GCD problem) The $(\rho, \eta, \gamma)$-Decisional Approximate GCD problem is: Let $p$ be a random odd integer of $\eta$ bits, $q_0$ and $r_0$ are uniformly distributed over $\mathbb{Z} \cap [0,\ 2^\gamma/p^2)$ and $\mathbb{Z} \cap (-2^\rho,\ 2^\rho)$, respectively. Given $x_0 = p^2 q_0 + r_0$ and polynomially many samples from $\mathcal{D}^\rho_{p,q_0}$ and $y \longleftarrow \mathcal{D}^\rho_{p,q_0} + \frac{p-1}{2}$, determine $b \in \{0, 1\}$ from $z = x + b \times r (\text{mod } x_0)$ where $x \longleftarrow \mathcal{D}^\rho_{p,q_0}$ and $r \longleftarrow [0, x_0) \cap \mathbb{Z}$.

Simply, try to distinguish the integer $z$ to be over the distribution $\mathcal{D}^\rho_{p,q_0}$ from being truly uniform integers $\in [0, x_0)$. The definition of Decisional Approximate-GCD that adopted

by [12], and proposed by [9], does not consider public key elements $\boldsymbol{\sigma}$ and $\boldsymbol{z}$; however, as an attacking scenario, the attacker has to guess $b^*$ with probability $Pr[b^* = 1/b = 0] - Pr[b^* = 0/b = 0] < \epsilon_{negl}$. Further, we have $Pr[b^* = 1/b = 0] = 1/2$ because $\sum_{i \in S} x_i \pmod{x_0}$ has a distribution with statistical distance from the uniform distribution at most $\varepsilon \leq \frac{1}{2}|D_{SI-DGHV} - D_{uniform}|_{\forall x \in pk}$. This can be proved using the following modified leftover hash lemma (LHL) used by [45]:

*Lemma* 3.2.1. Set $x_1, \cdots, x_\tau \longleftarrow \mathbb{Z}_{2^\gamma}$ uniformly and independently, and set $y = \sum_{i \in S} x_i \bmod 2^\gamma$ with random subset $S \subseteq \{1, 2, ..., \tau\}$. Then $(x_1, ..., x_\tau, y)$ is $\frac{1}{2}\sqrt{2^\gamma/2^\tau}$-uniform over $\mathbb{Z}_{2^\gamma}^{\tau+1}$.

Therefore, to make the distance $\epsilon = 2^{\frac{1}{2}(\gamma-\tau)}$ negligible, the constraint on the public key over $\mathcal{D}_{p,q_0}^{\rho}$ should satisfy $\tau \geqslant \gamma + 2\lambda$.

## 3.3 Batching SI-DGHV scheme

In general, the process of compacting $l$ bits $(m_0, m_1, \cdots, m_{l-1})$ into a single ciphertext and yet supporting pairwise homomorphic evaluation over each $m_i$, is known as batching a homomorphic scheme. The authors of [12] proposed generalization of batch setting in [7, 11, 8]. The batch generalization forms RNS with respect to base $\mathcal{B}_p$ of $(l)$ co-prime moduli $p_0^2, ..., p_{l-1}^2$. Consequently, the public key elements would be:

$$\begin{aligned} x_0 &= q_0 \cdot \pi^2 + CRT_{\mathcal{B}_p}(r_{0,0}, ..., r_{l-1,0}) \\ x_j &= q_j \cdot \pi^2 + CRT_{\mathcal{B}_p}(r_{0,j}, ..., r_{l-1,j}) \qquad j \in [0, \tau) \\ y_i &= q_i \cdot \pi^2 + CRT_{\mathcal{B}_p}(r_{0,i}, ..., r_{l-1,i}) + \frac{p_i - 1}{2} \cdot \prod_{j=0, j \neq i}^{l-1} p_i^2 \quad i \in [0, l) \end{aligned} \tag{3.8}$$

where $\pi = \prod_{i=0}^{l-1} p_i$ is co-prime with $q_0 \in [0, 2^\gamma/\pi^2)$, and $q_j, q_i \in \mathbb{Z} \cap [0,\ q_0)$ , $r_0, ..., r_{l-1} \in \mathbb{Z} \cap (-2^\rho,\ 2^\rho)$.

Moreover, using the encryption function of (3.7), the batched ciphertext has the following form:

$$C^{CRT} \leftarrow CRT_{q_0,\mathcal{B}_p}\left(q,\ r_0 + (2r_0^* + m_0)\cdot\frac{p_0-1}{2},\ ...,r_{l-1} + (2r_{l-1}^* + m_{l-1})\cdot\frac{p_{l-1}-1}{2}\right) \tag{3.9}$$

Because of the independence between the RNS channels, the addition of two $C_1^{CRT} + C_2^{CRT}$ yields a new ciphertext $C_3^{CRT}$ that encrypts the bitwise sum modulo 2 to decryption of both $C_1^{CRT}$ and $C_2^{CRT}$. Also, the multiplication of $C_1^{CRT}$ and $C_2^{CRT}$ yields componentwise batched ciphertext of type-II.

$$c_p^{(i)} \equiv r_p^{(i)} + (m_p^{(i)})\cdot\frac{p_{l-1}^2-1}{2} \pmod{p_i}$$

Similar to the un-batched ciphertext of type-II, the corresponding batched ciphertext can be converted using the same *SI-DGHV.Convert(∗)* procedure but for $\mathbf{z}$ be vector of $\Theta$ elements, with $k = 2\gamma + 2$ bits of precision after binary point. Also, we have $\mathbf{s}$ to be $l$ vectors of $\Theta$ bits s.t.

$$\frac{2^\eta}{p_i^2} = \langle s_i, z\rangle + \varepsilon_i \bmod 2^\eta \quad \forall i \in [0, l\ )$$

with $\varepsilon_i \ \leq\ 2^{-k}$.

Accordingly, and for $\tilde{s_i} = PowerOfTwo_\eta(s_i)$, we have $\boldsymbol{\sigma} = (\boldsymbol{\sigma}_1,\ \boldsymbol{\sigma}_2, ..., \boldsymbol{\sigma}_{\eta\Theta})$ with $\boldsymbol{\sigma}_i$ defined as follows:

$$\boldsymbol{\sigma}_i = CRT_{q_0,\ \mathcal{B}_p}\left(q_i, r_{0,i} + \left\lfloor (\tilde{s_{0,i}})\cdot\frac{p_0}{2^{\eta+1}}\right\rceil, \cdots, r_{l-1,i} + \left\lfloor (\tilde{s_{l-1,i}})\cdot\frac{p_l-1}{2^{\eta+1}}\right\rceil\right)$$

where $\boldsymbol{q}$ and $\boldsymbol{r}$ are random vectors chosen from $(\mathbb{Z}\cap[0,\ q_0))^{\eta\Theta}$, $(\mathbb{Z}\cap(-2^\rho,\ 2^\rho))^{\eta\Theta}$, respectively.

Therefore, and for $\boldsymbol{c}_p = (|\lfloor C_p^{CRT} \cdot z_i \rceil|_{2^\eta})_{1 \leq i \leq \Theta}$, the recovered type-I batched ciphertext is

$$c^* \leftarrow 2\langle \boldsymbol{\sigma}, BitDecomp_\eta(\boldsymbol{c}_p) \rangle$$

# Chapter 4

# RNS version of Fully Homomorphic Encryption over Integers

## 4.1 Problem statement

The homomorphic encryption proposed in [45] and its variant [12], are still considered to be far from practical because they deal with very large integers and require a huge amount of operations. This issue has motivated us to improve the practicality of the scheme in [12] (and [45] as well) by adopting *SI-DGHV.Mult(*)* and *SI-DGHV.Add(*)* in RNS, as well as the *SI-DGHV.Convert(*)* procedure. The RNS adoption will improve the timing performance of *SI-DGHV* which, by virtue of RNS, the *SI-DGHV.Evaluate(*)* and *SI-DGHV.Convert(*)* operate in parallel and over relatively small size moduli. The RNS version of *SI-DGHV.Convert(*)* preserves the original procedure functionality with improvement in the security hardness by eliminating the need for a sparse subset sum problem in the security requirement.

Other optimizations are available to enhance the multiplication complexity of [12]; for instance, one can use an efficient multiplication algorithm such as Karatsuba, Toom-Cook, and Schonhage-Strassen. Different optimization techniques have been already adopted by Coron *et al.* such as the one-multiplication technique [14], $\tau$ reduction technique [13], and public key compression techniques [14, 11]. These techniques are related to the expensive procedures required by the subset-sum assumption. Our variant can utilize the public key compression techniques [14, 11]; however, it does not require the use of the one-multiplication technique and the $\tau$ reduction because *RNS.Convert(*)* uses slightly different approach than *SI-DGHV.Convert(*)*.

This chapter presents the RNS variant of the SI-DGHV scheme [12]. Section 4.2 introduces the main concept of an RNS variant, and Section 4.3 presents the RNS's actual implementation. RNS base constraints, the Montgomery modulo reduction, base conversion, and ciphertext batching are introduced in Sections 4.4, 4.5, 4.6 and 4.8. Section 4.9 provides a semantic security proof of the RNS variant. Finally, Sections 4.10 and 4.11 present our complexity estimation and noise growth analysis for both schemes.

## 4.2 Preliminary

In this section a general overview of the RNS variant of [12] is presented.

Recalling the SI-DGHV scheme, the encryption of message $m_i$ with a given prime integer $p$ of size $\eta$ is:

$$c_i = q_i\, p^2 + \frac{p-1}{2}\,(m_i + 2r_i^*) + r_i \tag{4.1}$$

where $c_i$, $r_i$ and $r_i^*$ are of size $\gamma$, $\rho$ and $\rho^*$ bits, respectively, and $q \in [0, 2^\gamma/p^2)$.

As discussed in Chapter 2 and for a given proper RNS base $\mathcal{B}$ with $M = \prod_{i=0}^{n} m_i > 2^{2\gamma+2}$, the ciphertext (4.1) can be uniquely represented in the RNS base. Similar to [12], the execution of *SI-DGHV.mult(∗)* and *SI-DGHV.add(∗)* can be done correctly and in parallel over the RNS base. Furthermore, the result from *SI-DGHV.mult(∗)* is ciphertext of type-II ($c_p$ formulated in Eq. (3.4)) that has to be scaled by a factor of $1/p$ in order to restore type-I ciphertext which can be performed using *SI-DGHV.Convert*$(*)$. Unlike to [12], the RNS version of *SI-DGHV.Convert*$(*)$ (hereafter simply *RNS.Convert*$(*)$) does not use the subset-sum problem to hide the secret key.

The *RNS.Convert*$(*)$ will use the following two lemmas.

*Lemma* 4.2.1. Given integers $M > 2^{2\gamma}$ and $M^* > 0$, there is an integer $\chi = \left\lfloor \frac{M \times M^*}{p^2} \right\rfloor$ that scales a type-II ciphertext $c_p \in [0, 2^{2\gamma})$ by $M^*/p^2$ as follows:

$$\left\lfloor \frac{c}{M}\chi \right\rfloor = \left\lfloor \frac{cM^*}{p^2} \right\rfloor + \delta \tag{4.2}$$

where $\delta$ is an integer in the interval $[0, 2)$.

*Proof.* We have:

$$\frac{c}{M}\left(\frac{M \times M^*}{p^2} - 1\right) < \frac{c}{M}\chi \leq \frac{c \times M^*}{p^2}$$
$$0 \leq \frac{c \times M^*}{p^2} - \frac{c}{M}\chi < \frac{c}{M} < 1$$

which concludes the proof. □

*Lemma* 4.2.2. For an even integer $M^*$ and a given ciphertext as in Eq. (3.4) with $\log_2(r_p) < \rho_x = \eta + \rho + \rho^* + 4$, we have

$$\left\lfloor \frac{c \cdot M^*}{p^2} \right\rfloor = q_p \cdot M^* + \frac{M^*}{2}\, m_p + r \tag{4.3}$$

where $|r| < \frac{M^*}{p^2}(2^{\rho_x} + 0.5)$.

*Proof.* From Lemma 4.2.1 and Eq. (3.4), we have

$$\left\lfloor \frac{cM^*}{p^2} \right\rfloor \leq \frac{cM^*}{p^2}\big(q_p^* \cdot p^2 + \frac{p^2-1}{2}\ (m_1 \cdot m_2) + r_p\big) = q_p \cdot M^* + \frac{M^*}{2} \cdot m_p + \frac{M^*}{p^2}(r_p - m_p/2);$$

then $|r| \leq \frac{M^*}{p^2} \cdot (|r_p| + m_p/2) < \frac{M^*}{p^2}(2^{\rho_x} + 1/2)$. □

From the two previous lemmas, scaling $c_p$ can be obtained as follows:

$$\frac{p}{M^*} \cdot \left\lfloor \frac{c}{M}\chi \right\rfloor = q_p \cdot p + \frac{p-1}{2} \cdot m_p + r^{**} \tag{4.4}$$

or alternatively

$$\frac{p}{M^*} \left\lfloor \frac{c}{M}\chi \right\rfloor = \frac{p-1}{2}\ (2q_p + m_p) + r^{**}$$

where $r^{**} = \frac{p}{M^*} \cdot (\delta + r)$; hence, $|r^{**}| < \max(2 \cdot \frac{p}{M^*},\ \frac{1}{p} \cdot (2^{\rho_x} + 0.5)\ )$ and the result is type-I ciphertext with noise that is scaled by $1/p$.

## 4.3 RNS implementation

Through this chapter, we will use two RNS co-prime bases $\mathcal{B} = \{m_1, ..., m_n\}$ with $M = \prod_{i=0}^{n} m_i$ and $\mathcal{B}^* = \{m_1^*, ..., m_t^*\}$ with $M^* = \prod_{i=0}^{t} m_i^*$ s.t. $gcd(M, M^*) = 1$. Moreover, we will assume that the modulus bit size in both bases are the same and equal to $\mathcal{V}$ bits; thus, for an integer $c$, the RNS representations are denoted as $|\boldsymbol{c}|_{\mathcal{B}} \equiv (|c|_{m_1}, ..., |c|_{m_n})$ and $|\boldsymbol{c}|_{\mathcal{B}^*} \equiv (|c|_{m_1^*}, ..., |c|_{m_t^*})$ in bases $\mathcal{B}$ and $\mathcal{B}^*$, respectively.

However, the main idea of $RNS.Convert(*)$ is to perform the computation specified in Eq. (4.2) in RNS as follows:

$$\left\lfloor \frac{c}{M} \cdot \chi \right\rfloor = \frac{c_p \cdot \chi - |c_p \cdot \chi|_M}{M} \tag{4.5}$$

Accordingly, we can compute $c_p \cdot |\chi \cdot M^{-1}|_{M^*}$ in base $\mathcal{B}^*$, and $|c_p \cdot \chi|_M$ in base $\mathcal{B}$, then compute over all $|c_p \cdot \chi|_M \cdot |M^{-1}|_{M^*}$ over $\mathcal{B}^*$. Lastly, the output of Eq. (4.5) will be in base $\mathcal{B}^*$ which needs to be converted back to base $\mathcal{B}$ for further evaluation.

Eq. (4.5) requires the result of the multiplication, $c_p = 2c_1 \cdot c_2$, to be in $\mathcal{B} \cup \mathcal{B}^*$ because we need both $|\boldsymbol{c}_p|_{\mathcal{B}}$ and $|\boldsymbol{c}_p|_{\mathcal{B}^*}$. This extra multiplication over base $\mathcal{B}^*$ is of low cost with $t$ elementary multiplications over relatively small size moduli.

However, $\chi$ and the factor $p/M^*$ in Eq. (4.4) contain the secret $p$, thus it must be encrypted using *SI-DGHV* distribution $\mathcal{D}^{\rho}_{p,q_0}$. The following is complete description of the $RNS.convert(*)$ procedure which is divided into two main procedures ($c_p \cdot |\chi \cdot M^{-1}|_{M^*}$ and $|c_p \cdot \chi|_M \cdot |M^{-1}|_{M^*}$) followed by reduction modulo $x_0$.

**Computing** $c_p \cdot |\chi \cdot M^{-1}|_{M^*}$. The computation will be done in base $\mathcal{B}^*$; thus we need $c_p$ in $\mathcal{B}^*$ and $\chi \cdot M^{-1} \pmod{M^*}$. Furthermore, and in order to have correct multiplication, $c_p$ has to be assembled on the fly using the following definition of $Dec_{\mathcal{B}^*}(*)$ and $Pow_{\mathcal{B}^*}(*)$ functions:

$$\begin{aligned} Dec_{\mathcal{B}^*}(c_p) &= \left( \left| c_p \cdot \frac{m_1^*}{M^*} \right|_{m_1^*}, \left| c_p \cdot \frac{m_2^*}{M^*} \right|_{m_2^*}, \ \ldots\ , \left| c_p \cdot \frac{m_t^*}{M^*} \right|_{m_t^*} \right) \\ Pow_{\mathcal{B}^*}(|\chi \cdot M^{-1}|_{M^*}) &= \left( \left| \chi \cdot M^{-1} \frac{M^*}{m_1^*} \right|_{M^*}, \left| \chi \cdot M^{-1} \frac{M^*}{m_2^*} \right|_{M^*}, \ \ldots, \left| \chi \cdot M^{-1} \frac{M^*}{m_t^*} \right|_{M^*} \right) \end{aligned} \tag{4.6}$$

Thus we have $\langle Dec_{\mathcal{B}^*}(c_p),\ Pow_{\mathcal{B}^*}(|\chi \cdot M^{-1}|_{M^*}) \rangle \equiv c_p \cdot \chi \cdot M^{-1} \pmod{M^*}$.

As mentioned above, $\chi$, which is represented in the vector $Pow_{B^*}(|\chi \cdot M^{-1}|_{M^*})$, along with the factor $p/M^*$, must be encrypted using $\mathcal{D}^{\rho}_{p,q_0}$. For this reason, we generate the vector of encryptions $\mathcal{K}_{\mathcal{B}^*}$ as follows:

$$\mathcal{K}_{\mathcal{B}^*} = p^2\, \boldsymbol{q} + \boldsymbol{r} + \left\lfloor \frac{p}{2M^*} \cdot\ Pow_{\mathcal{B}^*}(|\chi \cdot M^{-1}|_{M^*}) \right\rceil \ \in [0,\ 2^{\gamma}\ )^t \tag{4.7}$$

where $\boldsymbol{q} \in [0, 2^{\gamma}/p^2)^t$ and $\boldsymbol{r} \in (-2^{\rho}, 2^{\rho})^t$.

**Computing** $-|c_p \cdot \chi|_M \cdot |M^{-1}|_{M^*}$. From Eq. (4.5), the computation of $|c_p \cdot \chi|$ has to be in $\mathcal{B}$ and at same time the inverse of $M$ in base $\mathcal{B}^*$; but, we need both computations in base $\mathcal{B}^*$. Therefore we will use an equivalent as follows:

$$\sum_{i=1}^{n} |c_p|_{m_i} \cdot |\chi \cdot M_i^{-1}|_{m_i} \cdot M_i = |c_p \cdot \chi|_M + \alpha \cdot M \ , \ 0 \leq \alpha < n \cdot 2^v \tag{4.8}$$

where $M_i = M/m_i$.

Similar to the previous computation, the term $|\chi \cdot M_i^{-1}|_{m_i}$ has to be encrypted as well:

$$\mathcal{K}_{\mathcal{B}} \ = \ p^2 \ \boldsymbol{q} + \boldsymbol{r} + \left( \left\lfloor \frac{p}{2M^*} \cdot \right| - |\chi \cdot M_i^{-1}|_{m_i} \ m_i^{-1} \Big|_{M^*} \right\rceil \right)_{1 \leq i \leq n} \tag{4.9}$$

where $\boldsymbol{q} \in [0, 2^{\gamma}/p^2)^n$ and $\boldsymbol{r} \in (-2^{\rho}, 2^{\rho})^n$.

**Run** $RNS.convert(\mathcal{K}_{\mathcal{B}}, \ \mathcal{K}_{\mathcal{B}^*}, \ |\boldsymbol{c}_p|_{\mathcal{B}^*}, |\boldsymbol{c}_p|_{\mathcal{B}})$**.** Let us consider both vectors $\mathcal{K}_{\mathcal{B}}$ and $\mathcal{K}_{\mathcal{B}^*}$ to be pre-computed and publicly known; thus for a given $c_p$ in both bases $(|\boldsymbol{c}_p|_{\mathcal{B}^*}, |\boldsymbol{c}_p|_{\mathcal{B}})$, we compute on-the-fly $Dec_{\mathcal{B}^*}(|\boldsymbol{c}_p|_{\mathcal{B}^*})$ and then output

$$c_{type-I} \leftarrow 2 \cdot [ \ \left\langle Dec_{\mathcal{B}^*}\left(|\boldsymbol{c}_p|_{\mathcal{B}^*}\right), \ \mathcal{K}_{\mathcal{B}^*} \right\rangle + \left\langle |\boldsymbol{c}_p|_{\mathcal{B}}, \ \mathcal{K}_{\mathcal{B}} \right\rangle \ ] \tag{4.10}$$

where $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$ is the dot product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. We note that Eq. (4.10) is computed over base $\mathcal{B}^*$.

*Lemma* 4.3.1. Let $c_1$ and $c_2$ be two type-I ciphertexts with noise size at most $(\rho_{type1}, \ \mathcal{V} + \log_2(t+n))$, and $c_p = 2c_1 \cdot c_2$ be a type-II ciphertext with noise size $\rho_{type2} \leq \eta + \rho_{type1} + \mathcal{V} + \log_2(t+n) + 4$. Using $RNS.Convert(*)$ in Eq. (4.10), the conversion of $c_p$, with $\rho_{type2} \geq \max(\eta + \rho_{type1} + \mathcal{V} + \log_2(t+n) + 4, \ 2\eta + t(1-\mathcal{V}))$, is a type-I ciphertext, with noise $(\rho_{type1}, \rho^*_{type1}) \leq (\rho_{type2} + 2 - \eta, \ \mathcal{V} + \log_2(t+n))$. The procedure yields type-I ciphertext with a noise growth at most $\mathcal{V} + \log_2(t+n) + 6$ bits.

*Proof.* For the first computation of $RNS.Convert(*)$, we have:

$$2 \cdot \left\langle Dec_{\mathcal{B}^*}\big(|\boldsymbol{c}_p|_{\mathcal{B}^*}\big), \mathcal{K}_{\mathcal{B}^*} \right\rangle = p^2\ Q^* + R^* + 2\ \left\langle Dec_{\mathcal{B}^*}\big(|\boldsymbol{c}_p|_{\mathcal{B}^*}\big), \left\lfloor \frac{p}{2M^*}\ Pow_{\mathcal{B}^*}(|\chi \cdot M^{-1}|_{M^*}) \right\rceil \right\rangle \tag{4.11}$$

where:

$$\begin{cases} Q^* = & 2\sum_{i=1}^{t} |c_p \dfrac{m_i^*}{M^*}|_{m_i^*}\ q_i & \rightarrow & 0 < Q^* < t\ 2^{\gamma+\mathcal{V}+1-2\eta} \\ R^* = & 2\sum_{i=1}^{t} |c_p \dfrac{m_i^*}{M^*}|_{m_i^*}\ r_i & \rightarrow & |R^*| < t\ 2^{\rho+\mathcal{V}+1} \end{cases}$$

moreover, there exist real numbers $v_i \in [-1, 1)\ \forall\ 1 \leq i \leq t$.

$$\begin{aligned} 2\ \left\lfloor \frac{p}{2M^*} \cdot\ Pow_{\mathcal{B}^*}(|\chi \cdot M^{-1}|_{M^*}) \right\rceil &= \frac{p}{M^*} \cdot\ Pow_{\mathcal{B}^*}(|\chi \cdot M^{-1}|_{M^*}) + (v_i)_{1\leq i\leq t} \\ &= \frac{p}{M^*} \cdot (|\chi \cdot M^{-1}\ M_i^*|_{M^*})_{1\leq i\leq t} + (v_i)_{1\leq i\leq t} \end{aligned}$$

Therefore, the last term in (4.11) can be obtained as follows :

$$\begin{aligned} 2\big\langle Dec_{\mathcal{B}^*}(c_p), \left\lfloor \frac{p}{2M^*}\ Pow_{\mathcal{B}^*}(|\chi M^{-1}|_{M^*}) \right\rceil \big\rangle &= \frac{p}{M^*}\ \sum_{i=1}^{t} |c_p M_i^{*-1}|_{m_i^*} |\chi M^{-1} M_i^*|_{M^*} \\ &\quad + \sum_{i=1}^{t} |c_p M_i^{*-1}|_{m_i^*} v_i \\ &= \frac{p}{M^*}\ (|c_p \chi M^{-1}|_{M^*} + \alpha^* M^*) + V^* \\ &= \frac{p}{M^*}\ |c_p \chi M^{-1}|_{M^*} + \alpha^* p + V^* \end{aligned} \tag{4.12}$$

where $|V^*|,\ \alpha^* < t\ 2^{\mathcal{V}}$.

Next step, computing the second term in $RNS.Convert(*)$ as follows

$$2 \cdot \left\langle |\boldsymbol{c}_p|_{\mathcal{B}},\ \mathcal{K}_{\mathcal{B}} \right\rangle = p^2\ Q + R + 2\ \left\langle |\boldsymbol{c}_p|_{\mathcal{B}}, \left( \left\lfloor \frac{p}{2M^*} \cdot \left| - |\chi \cdot M_i^{-1}|_{m_i}\ m_i^{-1} \right|_{M^*} \right\rceil \right)_{1\leq i\leq n} \right\rangle \tag{4.13}$$

where

$$\begin{cases} Q = & 2\sum_{i=1}^{n} |c_p|_{m_i} \cdot q_i & \rightarrow & 0 < Q < n\ 2^{\gamma+\mathcal{V}+1-2\eta} \\ R = & 2\sum_{i=1}^{n} |c_p|_{m_i} \cdot r_i & \rightarrow & |R| < n\ 2^{\rho+\mathcal{V}+1} \end{cases}$$

Similarly, there exist real numbers $v_i^* \in [-1, 1)\ \forall\ 1 \leq i \leq n$ .

$$2\ \left\lfloor \frac{p}{2M^*} \right| - |\chi \cdot M_i^{-1}|_{m_i}\ m_i^{-1} \Big|_{M^*} \Big\rceil \ = \ \frac{p}{M^*} \cdot \left( \left| - |\chi \cdot M_i^{-1}|_{m_i}\ m_i^{-1} \right|_{M^*} \right)_{1 \leq i \leq n} + (v_i^*)_{1 \leq i \leq n}$$

Therefore, $2\ \langle |\boldsymbol{c}_p|_{\mathcal{B}}, \left( \left\lfloor \frac{p}{2M^*} \cdot \left| - |\chi \cdot M_i^{-1}|_{m_i}\ m_i^{-1} \right|_{M^*} \right\rceil \right)_{1 \leq i \leq n} \rangle$ can be obtained as follows:

$$\begin{aligned}
&= \frac{p}{M^*} \textstyle\sum_{i=1}^{n} |c_p|_{m_i} \cdot \left| - |\chi \cdot M_i^{-1}|_{m_i}\ m_i^{-1} \right|_{M^*} + \textstyle\sum_{i=1}^{n} |c_p|_{m_i} v_i^* \\
&= \frac{p}{M^*} (| - \textstyle\sum_{i=1}^{n} |c_p|_{m_i} \cdot |\chi \cdot M_i^{-1}|_{m_i} M_i\ M^{-1}|_{M^*} + \alpha_1 M^*) + V \\
&= \frac{p}{M^*} (| - (|c_p \cdot \chi|_M\ + \alpha_2 M)\ M^{-1}|_{M^*}) + \alpha_1 p + V \\
&= \frac{p}{M^*} (| - |c_p \cdot \chi|_M\ \ M^{-1}|_{M^*}) + \alpha_3 p + V
\end{aligned} \tag{4.14}$$

where $V = \sum_{i=1}^{n} |c_p|_{m_i} v_i^*$, $\alpha_1$ and $\alpha_2 < n\ 2^{\mathcal{V}}$ and $\alpha_3 < n\ 2^{\mathcal{V}+1}$.

Finally, we can compute and analyze Eq. (4.4) by employing (4.10) as follows:

$$\begin{aligned}
2 \cdot [\ \langle Dec_{\mathcal{B}^*}(|\boldsymbol{c}_p|_{\mathcal{B}^*}),\ \mathcal{K}_{\mathcal{B}^*} \rangle + \langle |\boldsymbol{c}_p|_{\mathcal{B}},\ \mathcal{K}_{\mathcal{B}} \rangle\ ] = P^2(Q + Q^*) + (R + R^*) \\
+ 2 \langle Dec_{\mathcal{B}^*}(c_p), \left\lfloor \frac{p}{2M^*}\ Pow_{\mathcal{B}^*}(|\chi M^{-1}|_{M^*}) \right\rceil \rangle \\
+ 2 \langle |\boldsymbol{c}_p|_{\mathcal{B}}, \left( \left\lfloor \frac{p}{2M^*} \cdot \left| - |\chi \cdot M_i^{-1}|_{m_i}\ m_i^{-1} \right|_{M^*} \right\rceil \right)_{1 \leq i \leq n} \rangle
\end{aligned} \tag{4.15}$$

and use the results from Eq. (4.12) and Eq. (4.14) as follows:

$$\begin{aligned}
&2 \langle Dec_{\mathcal{B}^*}(c_p), \left\lfloor \frac{p}{2M^*}\ Pow_{\mathcal{B}^*}(|\chi M^{-1}|_{M^*}) \right\rceil \rangle + 2 \langle |\boldsymbol{c}_p|_{\mathcal{B}}, \left( \left\lfloor \frac{p}{2M^*} \right| - |\chi M_i^{-1}|_{m_i}\ m_i^{-1} \Big|_{M^*} \right\rceil \right)_{1 \leq i \leq n} \rangle \\
&= \frac{p}{M^*} \langle Dec_{\mathcal{B}^*}(c_p),\ Pow_{\mathcal{B}^*}(|\chi M^{-1}|_{M^*}) \rangle + V^* \\
&- \frac{p}{M^*} \langle |\boldsymbol{c}_p|_{\mathcal{B}}, \left( \left| - |\chi \cdot M_i^{-1}|_{m_i}\ m_i^{-1} \right|_{M^*} \right)_{1 \leq i \leq n} \rangle - V \\
&= \frac{p}{M^*} \left( \left\lfloor \frac{c}{M} \chi \right\rceil + \alpha_4\ M^* \right) + e
\end{aligned} \tag{4.16}$$

where $e = V^* - V$ and $\alpha_4$ is an integer.

Using Eq. (4.4), the right hand side of (4.16) can be written as:

$$\begin{aligned} & \frac{p}{M^*}\Big( \Big\lfloor \frac{c}{M}\chi \Big\rfloor + \alpha_4 \; M^* \Big) + e \\ = \;& p \; q_p + \frac{p-1}{2} \cdot m_p + r^{**} + \alpha_4 \cdot p + e \\ = \;& p \; q_p^* + \frac{p-1}{2} \cdot m_p + r^{***} \\ = \;& \frac{p-1}{2} \cdot (2q_p^* + m_p) + r_2^{***} \end{aligned}$$

where

$$\begin{aligned} |r^{**}| &= \max(2 \cdot \frac{p}{M^*}, \; \frac{1}{p}(2^{\rho_x} + 0.5) \;) \\ |r^{***}| &= |r^{**} + e| \; < (t+n) \; 2^{\mathcal{V}} + \max(2 \cdot \frac{p}{M^*}, \; \frac{1}{p}(2^{\rho_{type2}} + 0.5) \;) \\ q_p^* &< (t+n) \; 2^{\mathcal{V}} \\ |r_2^{***}| &= (q_p^* + r^{***}) < (t+n) \; 2^{\mathcal{V}+1} + \max(2 \cdot \frac{p}{M^*}, \; \frac{1}{p}(2^{\rho_{type2}} + 0.5) \;) \end{aligned}$$

It is worthwhile to mention that $q_p^*$ and $r_2^{***}$ are integers since $\alpha_4$ and $e$ are integers.

In conclusion, from (4.15) we obtain:

$$c_{type-I} = Q \; p^2 + R + \frac{p-1}{2}(2q_p^* + m_p) + r_2^{***} \tag{4.17}$$

with

$$\begin{aligned} Q &= (t+n) \; 2^{\gamma+\mathcal{V}+1-2\eta} \\ q_p^* &< (t+n) \; 2^{\mathcal{V}} \\ |R| &< (t+n) \; 2^{\rho_{type1}+\mathcal{V}+1} \\ |r_2^{***}| &\simeq \max(2^{\rho_x+1-\eta}, 2^{\eta+1+t(1-\mathcal{V})}) \\ c_{type-I} &< (t+n) \; 2^{\gamma+\mathcal{V}+2} \end{aligned} \tag{4.18}$$

which concludes the *proof*. □

## 4.4 Constraints on RNS bases

For base $\mathcal{B}$, $M$ has to contain the product of two ciphertext ($c_p = 2c_1 \cdot c_2$); thus, it should satisfy $M = \prod_{i=1}^{n} m_i > 2^{2\gamma+1}$. Accordingly, the dynamic space of $\mathcal{B}^*$ should satisfy $M^* > (n+t)\ 2^{\gamma+\mathcal{V}+3}$ in order to contain the recovered $c_{type-I}$ in (4.17).

Following the assumption that the size of the moduli in both bases $\mathcal{B}$ and $\mathcal{B}^*$ are the same and equal to $\mathcal{V}$ bits, and it is sufficient to have:

$$\begin{aligned} n(\mathcal{V}-1) &\geq 2\gamma+2 \\ t(\mathcal{V}-1) &\geq \log_2(t+n)+\gamma+\mathcal{V}+3 \end{aligned}$$

Moreover, it is necessity to have $2\gamma + 1 < n\mathcal{V}$, in all the above, we assumed that $m_i \in [2^{\mathcal{V}-1}, 2^{\mathcal{V}}-1]$, and $gcd(m_i, m_j) = 1$ with $(i \neq j)$.

In addition, We note that for asymptotic computational complexity, the relationship between $n$ and the ciphertext size $\gamma = O(\lambda^{\phi})$ can be written as:

$$n \in O(\frac{\lambda^{\phi}}{\beta \cdot \log_2(2\lambda)}) \tag{4.19}$$

for an integer $\beta$ satisfying $\gamma \in \Omega(\lambda^{\beta})$ and an integer $\phi$ that satisfies the scheme security requirements (i.e., $\phi = 5$ in DGHV).

*Proof.* The number of primes of size $\mathcal{V}$ bits is

$$(\#primes < 2^{\mathcal{V}}) \approx \frac{2^{\mathcal{V}}}{\mathcal{V} \cdot \ln(2)}$$

and to satisfy $M > 2^{2\gamma+5}$, we have:

$$\gamma \approx \frac{2^{\mathcal{V}}}{2\ln(2)} - \frac{5}{2}$$

Thus $\mathcal{V} \approx \log_2((2\gamma+5) \cdot ln(2))$, and for $\gamma \in \Omega(\lambda^{\beta})$ we have

$$n \in O(\frac{\lambda^{\phi}}{\beta \cdot \log_2(2\lambda) + \log_2(ln(2))}) \approx O(\frac{\lambda^{\phi}}{\beta \cdot \log_2(2\lambda)})$$

□

## 4.5 Reduce the result modulo $x_0$

From (4.18), the quantity obtained in (4.17) has at most $\log_2(t+n)+\mathcal{V}+2$ extra bits than type-I ciphertext. Thus, the ciphertext in (4.17) has to be reduced modulo $x_0$ in order to preserve the scheme's compactness.

Furthermore, the reduction has to be in the RNS base, which can be achieved including a new auxiliary modulus $\tilde{m}$ into base $\mathcal{B}^*$. This decompose a new base $\tilde{\mathcal{B}}$ as $\mathcal{B}^* \cup \tilde{m}$ (and accordingly $\tilde{M} = M^* \cdot \tilde{m}$).

Therefore, for an odd modulus $\tilde{m}$ co-prime to $x_0$ and $M^*$, we can define a reduction of the quantity obtained in (4.17) as follows:

$$\tilde{c} = \frac{c_{type-I} + x_0 \cdot |-c_{type-I}/x_0|_{\tilde{m}}}{\tilde{m}} \equiv c_{type-I} \cdot \tilde{m}^{-1} \bmod x_0 \tag{4.20}$$

From above, the suggested value for the auxiliary modulus is $\tilde{m} \geq \dfrac{2^{\gamma+k}}{x_0}$ which limits $|\tilde{c}| < x_0 + 2^{\gamma} \cdot \dfrac{2^k}{\tilde{m}} \leq 2x_0$ and we also have $k = \log_2(t+n) + \mathcal{V} + 2$ (see Eq. (4.18)).

However, the result from Eq. (4.20) will have an extra element $\tilde{m}^{-1}$. To eliminate $\tilde{m}^{-1}$, we will multiply the recovered ciphertext $c_{type-I}$ by $\tilde{m}$ and get $c^*_{type-I}$. But we want to perform this multiplication without any extra cost, thus the multiplication will be impeded in $RNS.Convert(*)$ procedure by modifying both $\mathcal{K}_{\mathcal{B}}$ and $\mathcal{K}_{\mathcal{B}^*}$ to be:

$$\tilde{\mathcal{K}}_{\tilde{\mathcal{B}}} = |(p^2 \cdot \boldsymbol{q} + \boldsymbol{r}) \cdot \tilde{m}|_{x_0} + \tilde{m} \cdot \left\lfloor \frac{p}{2M^*} \cdot \ Pow_{\tilde{\mathcal{B}}}(|\chi \cdot M^{-1}|_{M^*}) \right\rceil \tag{4.21}$$

$$\tilde{\mathcal{K}_{\mathcal{B}}} = |(p^2 \cdot \boldsymbol{q} + \boldsymbol{r}) \cdot \tilde{m}|_{x_0} + \tilde{m} \cdot \left( \left\lfloor \frac{p}{2M^*} \cdot \left| - |\chi \cdot M_i^{-1}|_{m_i} \ m_i^{-1} \right|_{M^*} \right\rceil \right)_{1 \leq i \leq n} \tag{4.22}$$

The procedure described in the following lemma.

*Lemma* 4.5.1. For a given type-II ciphertext $c_p$ in both bases $\tilde{\mathcal{B}}$ and $\mathcal{B}$ ($|\boldsymbol{c}_p|_{\tilde{\mathcal{B}}}$ and $|\boldsymbol{c}_p|_{\mathcal{B}}$), the type-I ciphertext $c^*_{type-I}$ can be obtained using both $\tilde{\mathcal{K}_{\mathcal{B}}}$ and $\tilde{\mathcal{K}_{\tilde{\mathcal{B}}}}$ as follows:

$$c^*_{type-I} \leftarrow RNS.convert(\tilde{\mathcal{K}_{\mathcal{B}}},\ \tilde{\mathcal{K}_{\tilde{\mathcal{B}}}},\ |\boldsymbol{c}_p|_{\tilde{\mathcal{B}}}, |\boldsymbol{c}_p|_{\mathcal{B}}). \tag{4.23}$$

where $\tilde{\mathcal{B}}$ as $\mathcal{B}^* \cup \tilde{m}$.

*Proof.* As a sketch proof that follows the same procedures in the proof of Lemma 4.3.1, then for a vector $\boldsymbol{y} = Dec_{\tilde{\mathcal{B}}}\big(|\boldsymbol{c}_p|_{\tilde{\mathcal{B}}}\big) \cup |\boldsymbol{c}_p|_{\mathcal{B}}$ where $y_i < 2^{\mathcal{V}}$, we have:

$$c^*_{type-I} = 2 \cdot \sum_{i=1}^{(t+1)+n} y_i \cdot |(p^2 \cdot q_i + r_i) \cdot \tilde{m}|_{x_0} + \tilde{m}(\frac{p-1}{2} \cdot (2q^*_p + m_p) + r^{***}_2\ )$$

where from (4.11) we have an equivalent $2\tilde{m} \cdot \big\langle Dec_{\tilde{\mathcal{B}}}\big(|\boldsymbol{c}_p|_{\tilde{\mathcal{B}}}\big), \left\lfloor \frac{p}{2M^*} \cdot\ Pow_{\tilde{\mathcal{B}}}(|\chi \cdot M^{-1}|_{M^*}) \right\rceil \big\rangle$, and from (4.13), we have $2\tilde{m} \cdot \big\langle |\boldsymbol{c}_p|_{\mathcal{B}}, \left( \left\lfloor \frac{p}{2M^*} \cdot \left| - |\chi \cdot M_i^{-1}|_{m_i}\ m_i^{-1} \right|_{M^*} \right\rceil \right)_{1 \le i \le n} \big\rangle$. Thus adding them both we get $\tilde{m} \cdot (\frac{p-1}{2} \cdot (2q^*_p + m_p) + r^{***}_2\ )$ which conclude the proof.

As a remark, we have:

$$\begin{aligned} \textstyle\sum_{i=1}^{(t+1)+n} y_i \cdot |(p^2 \cdot \boldsymbol{q} + \boldsymbol{r}) \cdot \tilde{m}|_{x_0} &= |(p^2 \cdot \textstyle\sum_{i=1}^{(t+1)+n} y_i \cdot q_i + \sum_{i=1}^{(t+1)+n} y_i \cdot r_i) \cdot \tilde{m}|_{x_0} + \alpha x_0 \\ &= |(p^2 \cdot Q + R) \cdot \tilde{m}|_{x_0} + \alpha x_0 \end{aligned}$$

where $\alpha \le (t+n) \cdot 2^{\mathcal{V}}$. □

Finally, we are ready to perform the reduction modulo $x_0$ as described in the following lemma.

*Lemma* 4.5.2. For a recovered ciphertext $c^*_{type-I}$ in (4.7) and an auxiliary modulus $\tilde{m} > (t+n) \cdot 2^{\mathcal{V}+1} + 2$, the reduction modulo $x_0$ of $c^*_{type-I}$ can be done using (4.20).

*Proof.* For a given $c^*_{type-I}$ in (4.7) and using (4.20), we have:

$$\begin{aligned}
\tilde{c} &= \frac{c^*_{type-I} + x_0 \cdot | - c^*_{type-I}/x_0|_{\tilde{m}}}{\tilde{m}} \\
&= \frac{|(p^2 \cdot Q + R) \cdot \tilde{m}|_{x_0} + \alpha x_0 + x_0 \cdot | - ((p^2 \cdot Q + R) \cdot \tilde{m}|_{x_0} + \alpha x_0)/x_0|_{\tilde{m}}}{\tilde{m}} \\
&\quad + \frac{p-1}{2}(2q^*_p + m_p) + r^{***}_2 \qquad (4.24) \\
&= p^2 \cdot Q + R + \alpha^* x_0 + (\frac{p-1}{2} \cdot (2q^*_p + m_p) + r^{***}_2\ ) \\
&= p^2 \cdot Q^* + R^* + (\frac{p-1}{2} \cdot (2q^*_p + m_p) + r^{***}_2\ )
\end{aligned}$$

where $\alpha^*$ is an integer, $Q^* = Q + \alpha^* q_0$ and $|R^*| \leq |R + \alpha^* r_0| < (t+n) \cdot 2^{\rho_{type1} + \mathcal{V} + 4}$ when $x_0 = p^2 q_0 + r_0$.

Moreover, from (4.24), by using central reminder theorem $| - ((p^2 \cdot Q + R) \cdot \tilde{m}|_{x_0} + \alpha x_0)/x_0|_{\tilde{m}} < \tilde{m}/2$, and $|(p^2 \cdot Q + R) \cdot \tilde{m}|_{x_0} < x_0$, then we have $|p^2 \cdot Q^* + R^*| < x_0 \cdot (0.5 + \frac{\alpha+1}{\tilde{m}})$.

Assume that $h = (0.5 + \frac{\alpha+1}{\tilde{m}})$ is a rational number; thus to reduce $c^*_{type-I}$, we should have $h < 1$. In order to have $h < 1$, and given that $\alpha \leq (t+n) \cdot 2^{\mathcal{V}}$ (see Lemma 4.5.1 proof) , we have

$$\begin{aligned}
0.5\tilde{m} &> \alpha + 1 \\
\tilde{m} &> (t+n) \cdot 2^{\mathcal{V}+1} + 2 \qquad (4.25)
\end{aligned}$$

Finally, using $\tilde{m} > (t+n) \cdot 2^{\mathcal{V}+1} + 2$, the reduced ciphertext modulo $x_0$ is given below

$$\tilde{c} = \tilde{q}\ p^2 + \tilde{r} + \frac{p-1}{2}(2q^*_p + m_p) \qquad (4.26)$$

where $\tilde{r} = R^* + r^{***}_2$, $\tilde{q} = Q^*$ and with

$$\begin{aligned}
q^*_p &= (t+n)\ 2^{\mathcal{V}} \\
|\tilde{r}| &< (t+n) \cdot 2^{\rho_{type1} + \mathcal{V} + 4} + \max(2^{\rho_x + 1 - \eta}, 2^{\eta + 1 + t(1-\mathcal{V})}) \qquad (4.27) \\
\tilde{c} &< x_0
\end{aligned}$$

which conclude the proof. □

The above implies that the RNS dynamic space M has to contain $|2c_1c_2| < 2(x_0)^2$; accordingly, the system's dynamic space should satisfy $M > 2^{2\gamma+1}$.

## 4.6 Exact base conversion

Up to now, we have recovered compact type-I ciphertext in base $\mathcal{B}^*$ (4.26). To complete the next level of evaluations, we need to perform base conversion $\mathcal{B}^* \rightarrow \mathcal{B}$. A naive way to perform a base conversion is to reconstruct $|\tilde{c}|_{\mathcal{B}^*}$ into the positional number system and then convert it back to RNS in base $\mathcal{B}$. The whole procedure is illustrated in the following:

The CRT's Eq. (2.14) can be rewritten as follows

$$\sum_{i=0}^{t-1} |c_i \cdot {M_i^*}^{-1}|_{m_i^*} \cdot M_i^* = \tilde{c} + \alpha M^* \tag{4.28}$$

where $0 < \alpha \leq t-1$.

Then, RNS forward conversion, to base $\mathcal{B}$, can be applied on (4.28) to have $|\tilde{c}^*|_{\mathcal{B}}$.

$$\tilde{c}_i^* = |\tilde{c}|_{m_i} + |\alpha M^*|_{m_i} \; \forall \; i \in [0, n) \tag{4.29}$$

Yet, the result is not an exact base conversion due to the extra term $|\alpha M^*|_{m_i}$. Therefore, to obtain the correct $|\tilde{c}|_{\mathcal{B}}$, we have to subtract $|\alpha M^*|_{\mathcal{B}}$ from $|\tilde{c}^*|_{\mathcal{B}}$ which requires finding $\alpha$. The authors in [38] proposed a technique to obtain $\alpha$ by using an extra redundant modulus $m_t^* > t$. Their procedure obtains $\alpha$ by taking (4.28) modulo $m_t^*$ as follows:

$$\sum_{i=0}^{t-1} \left| |c_i \cdot {M_i^*}^{-1}|_{m_i^*} \cdot M_i^* \right|_{m_t^*} = \tilde{c}_t + \left|\alpha M^*\right|_{m_t^*} \tag{4.30}$$

where $\tilde{c}_t = \left|\tilde{c}\right|_{m_t^*}$ which could be carried out through the computation of (4.26) or use base

extension. Finally, rearrange the above to have [38]:

$$\alpha = \left| |M^{*-1}|_{m_t^*} \cdot \Big( \sum_{i=0}^{t-1} \left| |c_i \cdot M_i^{*-1}|_{m_i^*} \cdot M_i^* \right|_{m_t^*} - \tilde{c}_t \Big) \right|_{m_t^*} \tag{4.31}$$

*Lemma* 4.6.1. For an integer $\tilde{c} < x_0$ in RNS base $\mathcal{B}^*$ with $M^* \cdot m_t^* > x_0$ and $m_t^* \geq t$, the exact base conversion $(|\tilde{c}|_{\mathcal{B}^*} \rightarrow |\tilde{c}|_{\mathcal{B}})$ is:

$$|\tilde{c}|_{\mathcal{B}} = \left( \left| \sum_{i=0}^{t-1} \left| |c_i \cdot M_i^{*-1}|_{m_i^*} \cdot M_i^* \right|_{m_j} - |\alpha M^*|_{m_j} \right|_{m_j} \right)_{0 \leq j < n} \tag{4.32}$$

where $\alpha$

$$\alpha = \left| |M^{*-1}|_{m_t^*} \cdot \Big( \sum_{i=0}^{t-1} \left| |c_i \cdot M_i^{*-1}|_{m_i^*} \cdot M_i^* \right|_{m_t^*} - \tilde{c}_t \Big) \right|_{m_t^*}$$

The work of Bajard *et al.* used further optimization to reduce the size of $M^*$ by adopting part of $m_t^*$ in the dynamic space $M^*$ as follows [3]:

*Lemma* 4.6.2. For an integer $\tilde{c} < 3 \cdot x_0$ in RNS base $\mathcal{B}^*$ with $M^* = M^{**} \cdot \beta > 3x_0$ for $\beta \geq 1$, the exact base conversion $(|\tilde{c}|_{\mathcal{B}^*} \rightarrow |\tilde{c}|_{\mathcal{B}})$ using (4.32) and (4.31) follows the assumption that $m_t^* \geq 2 \cdot (t - 1 + \lceil \beta \rceil)$ and $M^{**} \cdot \beta > 3x_0$.

## 4.7 Putting it together

Below, we sketch the new RNS variant scheme showing the sequence in which various procedures are called.

**RNS homomorphic evaluation:** The evaluation is done over both bases ($\mathcal{B}$ and $\mathcal{B}^*$) and over $m_t^*$ and $\tilde{m}$. For two type-I ciphertexts $C_1$ and $C_2$, the RNS evaluation is:

$$C_1 + C_2 \xrightarrow{RNS} (|\boldsymbol{c_1} + \boldsymbol{c_2}|_{\mathcal{B}},\ |\boldsymbol{c_1} + \boldsymbol{c_2}|_{\mathcal{B}^* \cup\, \tilde{m}\, \cup m_t^*})$$

$$C_1 \times C_2 \xrightarrow{RNS} (|\boldsymbol{c_1} \times \boldsymbol{c_2}|_{\mathcal{B}},\ |\boldsymbol{c_1} \times \boldsymbol{c_2}|_{\mathcal{B}^* \cup\, \tilde{m}\, \cup m_t^*})$$

**RNS convert procedure:** $RNS.Convert(*)$ procedure is used to transform a noisy type-II to type-I ciphertext with a scaled noise. The procedure as described in Section 4.3 is performed over both bases $\mathcal{B}$ and $\tilde{\mathcal{B}}^* = \mathcal{B}^* \cup \tilde{m} \cup m_t^*$, and the result will be in $\tilde{\mathcal{B}}^*$.

$$|c^*_{type-I}|_{\tilde{\mathcal{B}}^*} \leftarrow RNS.convert(\tilde{\mathcal{K}}_{\mathcal{B}},\ \tilde{\mathcal{K}}_{\tilde{\mathcal{B}}^*},\ |\boldsymbol{c}_p|_{\tilde{\mathcal{B}}^*}, |\boldsymbol{c}_p|_{\mathcal{B}}).$$

**Reduction modulo $x_0$:** The reduction is used to preserve the scheme's compactness and performed over $\mathcal{B}^* \cup m_t^*$.

**Exact base conversion:** This is the very last procedure in our RNS variant. The procedure is used to recover the base $\mathcal{B}$ and modulus $\tilde{m}$ for the next level of evaluation.

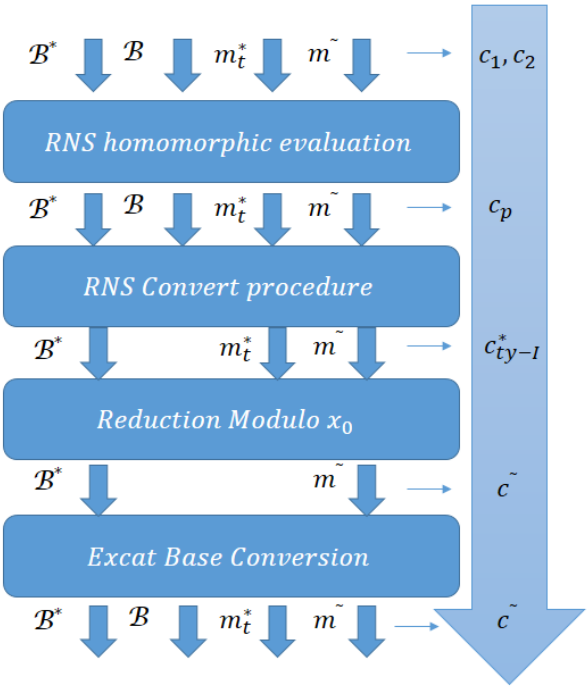


Figure 4.1: A block diagram showing the sequence of various procedures of the new RNS variant scheme.

To sum up, the previous sections show how the new RNS variant successfully recovers and reduces type-I ciphertexts and Figure 4.1 depicts how the RNS moduli change from one procedure to another.

## 4.8 Ciphertext batching

Similar to [7, 11, 8] and [12], and as described in Chapter 3, the type-I ciphertext for a given $l$ bit message $(m_i)_{0\leq i<l-1}$ encrypted using secret $p_0, ..., p_{l-1}$ of $\eta$ bits is:

$$c = q \cdot \pi^2 + CRT_{\mathcal{B}_p}\left(r_0 + (2r_0^* + m_0) \cdot \frac{p_0 - 1}{2},\ ..., r_{l-1} + (2r_{l-1}^* + m_{l-1}) \cdot \frac{p_{l-1} - 1}{2}\right)$$

where $\pi = \prod_{i=0}^{l-1} p_i$ is co-prime with $q_0 \in [0, 2^\gamma/\pi^2)$, and $\log_2(|r_i|) < \rho$, $\log_2(|r_i^*|) < \rho^*$ and $q \in \mathbb{Z} \cap [0,\ q_0)$.

Thus, the multiplication of $c_1$ and $c_2$ yields component wise batched type-II ciphertext.

$$c_p^{(i)} \equiv r_p^{(i)} + (m_p^{(i)}) \cdot \frac{p_{l-1}^2 - 1}{2} \pmod{p_i^2}$$

with similar noise $\log_2(|r_p^{(i)}|) < \rho_x = \eta + \rho + \rho^* + 4$ as un-batched ciphertext type-II.

Like the un-batched ciphertext type-II, the batched ciphertext type-II can be converted using the same $RNS.Convert(*)$ procedure. Therefore, using Lemma 4.2.1 and 4.2.2, we associate $\chi_i = \left\lfloor \frac{M \times M^*}{p_i^2} \right\rfloor$ for each secret $p_i$. Now we define the batched version of both $\mathcal{K}_{\mathcal{B}}$ and $\mathcal{K}_{\mathcal{B}^*}$ as follows:

$$\mathcal{K}_{\mathcal{B}} \ = \ \pi^2\ \boldsymbol{q} + \left(CRT_{\mathcal{B}_p}\left(\left(r_{i,j} + \left\lfloor \frac{p_i}{2M^*} \right| - |\chi_i \cdot M_j^{-1}|_{m_j}\ m_j^{-1} \right|_{M^*} \right\rceil)_{0\leq i<l-1}\right)\right)_{1\leq j\leq n} \tag{4.33}$$

and

$$\mathcal{K}_{\mathcal{B}^*} \;=\; \pi^2\ \boldsymbol{q}^* + \Big(CRT_{\mathcal{B}_p}\Big(\big(r^-_{i,j} + \Big\lfloor \frac{p_i}{2M^*}\ \Big|\chi_i \cdot M^{-1} \cdot \frac{M^*}{m^*_j}\Big|_{M^*}\ \Big\rceil\big)_{0\le i<l-1}\Big)\Big)_{1\le j<t} \tag{4.34}$$

where $\boldsymbol{q} \in [0, q_0)^n$ and $r_{i,j} \in (-2^\rho, 2^\rho)$ and $\boldsymbol{q}^* \in [0, q_0)^t$, $r^-_{i,j} \in (-2^\rho,\ 2^\rho)$.

## 4.9 Semantic security

The semantic security of the RNS variant follows the variant of the Approximate-GCD problem introduced in [9] and [12]. For given an integer $z$, it is hard to distinguish whether $z$ is from $\mathcal{D}^\rho_{p,q_0}$ or a truly uniform integer $\in [0, x_0)$, where:

$$\mathcal{D}^\rho_{p,q_0} = \Big\{ choose\ q \longleftarrow \mathbb{Z} \cap [0,\ q_0),\ r \longleftarrow \mathbb{Z} \cap (-2^\rho,\ 2^\rho)\ : Outputs\ x = q\ p^2 + r \Big\}$$

**Definition 4.9.1.** (Decisional Approximate-GCD problem) The $(\rho, \eta, \gamma)$-Decisional Approximate-GCD problem is: for a random $\eta$ bit prime $p$, given a $\gamma$ bit integer $x_0 = p^2 q_0 + r_0$ and polynomially many samples from $\mathcal{D}^\rho_{p,q_0}$ and $y \longleftarrow \mathcal{D}^\rho_{p,q_0} + \frac{p-1}{2}$, determine $b \in \{0, 1\}$ from $z = x + b \times r \pmod{x_0}$ where $x \longleftarrow \mathcal{D}^\rho_{p,q_0}$ and $r \longleftarrow [0, x_0) \cap \mathbb{Z}$.

The assumption is that the Decisional Approximate-GCD problem is hard for any polynomial time attacker which implies the hardness of determining whether $b$ is 0 or 1.

Similarly, using the reduction of the Decisional Approximate-GCD problem [9], the security of batched RNS variant is based on $(\rho, \eta, \gamma)$-$l$-Decisional Approximate-GCD problem which is as in the original scheme. Below is the problem definition:

**Definition 4.9.2.** ($l$-Decisional Approximate-GCD problem) The $(\rho, \eta, \gamma,)$-$l$-Decisional Approximate-GCD problem is: Consider $l$ random $\eta$ bit primes $p_0, p_1, ...p_{l-1}$, and a $\gamma$ bit

integer $x_0 = \pi^2 q_0 + CRT_{p_0,\ldots p_{l-1}}(r_0, \ldots r_{l-1})$ where $q_0 \in \mathbb{Z} \cap [0,\ 2^\gamma/p^2)$, and $r_i \in \mathbb{Z} \cap (-2^\rho,\ 2^\rho)$. Given polynomially many samples from $\mathcal{D}^\rho_{q_0,p_1,\ldots p_{l-1}}$ and $y_i \longleftarrow \mathcal{D}^\rho_{q_0,p_1,\ldots p_{l-1}} + \dfrac{p_i - 1}{2} \cdot \prod_{j=0,j\neq i}^{l-1} p_i^2$ for $0 \leq i < l$, determine $b \in \{0,1\}$ from $z = x + b \times c \pmod{x_0}$ where $x \longleftarrow \mathcal{D}^\rho_{q_0,p_1,\ldots p_{l-1}}$, $c \longleftarrow [0, x_0) \cap \mathbb{Z}$ and

$$\mathcal{D}^\rho_{q_0,p_1,\ldots p_{l-1}} = \Big\{\text{Outputs } CRT_{q_0,p_0,\ldots p_{l-1}}(q, r_0, \ldots r_{l-1}) : q \in \mathbb{Z} \cap [0,\ q_0),\ r_i \in \mathbb{Z} \cap (-2^\rho,\ 2^\rho)\Big\}.$$

In addition, the RNS variant scheme is semantically secure under extra public key elements $\mathcal{K}_{\mathcal{B}^*}$ and $\mathcal{K}_{\mathcal{B}}$; and since the security of both $\mathcal{K}_{\mathcal{B}^*}$ and $\mathcal{K}_{\mathcal{B}}$ only depend on Decisional Approximate-GCD problem as in the previous work [12, 9, 14], the RNS variant's security is based on Decisional Approximate-GCD assumptions. Therefore, the RNS variant enhances the security assumption of FHE over integers by eliminating the need for subset sum problem.

## 4.10 Estimation of complexity

Throughout this section, we will use the definition of elementary operations over a word that fits in $\mathcal{V}$ bits. Thus each $\mathcal{V}$ bits of positional number system is considered as a word. Now our elementary operations are defined as follows: [1]

- Elementary multiplication of two words denoted as EM and the notation for modulo reduction is M.
- Elementary modular multiplication is denoted as EMM which corresponds to an elementary multiplication modulo $m_i$ (i.e. $|2c_1 \cdot c_2|_{\mathcal{B}}$ requires $n$ EMM).

[1] The notation are mostly adopted from Bajard *et al.* [3].

- Dot modular product over $x$ moduli is denoted as $\text{DMP}_x$. This is costlier compared to EM and EMM; yet, the cost of the inner modulo reduction can be reduced by using lazy reduction (i.e. $\langle |\boldsymbol{a}|_{\mathcal{B}}, |\boldsymbol{b}|_{\mathcal{B}} \rangle$ requires $n$ $\text{DMP}_n$ where $n = \#\mathcal{B}$ )

Another remark, to simplify our complexity estimation, the estimation does not consider the two auxiliary moduli $\tilde{m}$ and $m_t^*$ since $t$, and $n >> 2$.

For bases $\mathcal{B} \equiv \{m_1, ..., m_n\}$ and $\mathcal{B}^* \equiv \{m_1^*, ..., m_t^*\}$, the multiplication between two type-I ciphertexts $(2c_1 \cdot c_2)$ has to be done in both bases $\mathcal{B}$ and $\mathcal{B}^*$; therefore, we need $n+t$ EMM.

Furthermore, to convert from type-II to type-I, we have to compute (4.10), throughout (4.11) and (4.13) requiring $t$ EMM, $t^2$ $\text{DMP}_t$ and $nt$ $\text{DMP}_n$. This cost can be reduced to $(t + t^2 + nt)$ EMM and $(t + 1)$ M using lazy reduction:

$$Dec_{\mathcal{B}^*}(c_p) = \left( \left| c_p \cdot \frac{m_1^*}{M^*} \right|_{m_1^*}, \left| c_p \cdot \frac{m_2^*}{M^*} \right|_{m_2^*}, \dots, \left| c_p \cdot \frac{m_t^*}{M^*} \right|_{m_t^*} \right) \Rightarrow t \text{ EMM}$$

and

$$c_{type-I} \leftarrow 2 \cdot [\ \langle Dec_{\mathcal{B}^*}(|\boldsymbol{c}_p|_{\mathcal{B}^*}),\ \mathcal{K}_{\mathcal{B}^*} \rangle + \langle |\boldsymbol{c}_p|_{\mathcal{B}},\ \mathcal{K}_{\mathcal{B}} \rangle\ ] \Rightarrow\ t^2 \text{ DMP}_t\ + nt \text{ DMP}_n$$

In addition, the Montgomery reduction in Section 4.5 requires $2+t$ EMM in total using (4.20) in bases $\mathcal{B}^*$:

$$\tilde{c} = \frac{c_{type-I} + x_0 \cdot |-c_{type-I}/x_0|_{\tilde{m}}}{\tilde{m}} \Rightarrow 2 + t \text{ EMM}$$

Lastly, exact base conversion, as indicated in Section4.6, requires $nt + 3t + 1$ EMM and $n$ M:

$$\sum_{i=0}^{t-1} \left| |c_i \cdot {M_i^*}^{-1}|_{m_i^*} \cdot M_i^* \right|_{m_t^*} = \tilde{c}_t + \left| \alpha M^* \right|_{m_t^*} \Rightarrow t \text{ EMM}$$

$$\alpha = \left| |M^{*-1}|_{m_t^*} \cdot \Big( \sum_{i=0}^{t-1} \left| |c_i \cdot M_i^{*-1}|_{m_i^*} \cdot M_i^* \right|_{m_t^*} - \tilde{c}_t \Big) \right|_{m_t^*} \Rightarrow 1 \text{ EMM}$$

$$|\tilde{c}|_{\mathcal{B}} = \left( \left| \sum_{i=0}^{t-1} \left| |c_i \cdot M_i^{*-1}|_{m_i^*} \cdot M_i^* \right|_{m_j} - |\alpha M^*|_{m_j} \right|_{m_j} \right)_{0 \le j < n} \Rightarrow t(n+2) \text{ EMM and } n \text{ M}$$

Therefore, the total cost for RNS multiplication and conversion is:

$$\begin{aligned} \mathcal{C} &= (t^2 + 6t + n + 2nt + 3) \text{ EMM} + (t + n + 1) \text{ M} \\ &= (\frac{5}{4}n^2 + 4n + 3) \text{ EMM} + (\frac{3}{2}n + 1) \text{ M} \end{aligned}$$

for $n \approx 2t$.

It is easy to show that the original scheme's complexity is $\Theta \cdot n^2 + t^2$ which can be rewritten as $\Theta \cdot n^2 + n^2/4$. Roughly, the complexity comparison between two schemes is:

$$(\frac{5}{4}n^2 + 4n + 3) \quad \text{EMM} \quad vs \quad (\Theta \cdot n^2 + n^2/4) \text{ EM}$$

Therefore, the asymptotic complexity comparison for the two schemes is

$$O(n^2) \qquad vs \quad O(\Theta \cdot n^2)$$

The original scheme complexity can be reduced from $O(\Theta \cdot n^2)$ to $O(\Theta \cdot n^{1+\varepsilon})$ where $\varepsilon$ depends on the underlying multiplication algorithm such as Karatsuba, Toom-Cook or Schonhage-Strassen (e.g., $\varepsilon = 0.585$ using the Karatsuba algorithm). In addition, the RNS variant of [12] can be fully parallelized, and accordingly one can have an area and a delay linearly with $n$ (i.e., $C^{delay} = O(n)$).

Therefore, using the approximation $\Theta \sim \omega(n)$, the RNS variant shows an advantage over the original scheme by almost $n$.

## 4.11 Noise growth analysis

This section discusses the noise analysis of both schemes. The analysis compares the noise added by each scheme. The compression will be in the asymptotic sense and based on the previous implementation parameters. As described in Chapter 3, using *SI-DGHV.Convert(*)* procedure, the recovered type-I ciphertext will have a noise of size:

$$\begin{aligned} \rho_p &= \rho + \log_2(\Theta) + 9 \\ \rho_p^* &= \log_2(\Theta) \end{aligned} \tag{4.35}$$

where $\rho$ is the noise size of the original ciphertext and $\Theta$ is the vector size of the public key $\mathbf{z}$. In addition, it has also been shown (in Section 4.3) that the noise growth in the RNS variant is around

$$\begin{aligned} \rho_p &= \rho + \mathcal{V} + \log_2(n+t) + 6 \\ \rho_p^* &= \mathcal{V} + \log_2(n+t) \end{aligned} \tag{4.36}$$

where $n$ and $t$ are the number of RNS mudoli of $\mathcal{V}$ bits.

Therefore, the asymptotic comparison of the RNS variant and the original scheme is:

$$O(2\lambda + \phi \cdot \log_2(3\lambda) - \log_2(2\beta \cdot \log_2(2\lambda)) \ ) \quad vs \quad O(\lambda + \log_2(L \cdot \lambda)) \tag{4.37}$$

where $\gamma = O(\lambda^{\phi})$, $\mathcal{V} < O(\lambda)$, and $L$ is the depth of homomorphic operation.

This indicates that the FHE over integers becomes *leveled* FHE in the sense that it has certain level of operation depth. Considering that for a type-I ciphertext (as in (4.17)), the noise level should not exceed:

$$\begin{aligned} 2^{\rho} + 2^{\rho^*} &< p/4 \\ |r_2^{***}| + |q_p^*| &< p/4 \end{aligned}$$

To have a fair compassion, we can compute the actual level of operations that both schemes can reach under the same security parameters.

$$d_{\text{RNS}} < \frac{\eta - 2 - \rho}{\mathcal{V} + \log_2(n+t) + 6}$$

$$d_{\text{SI-DGHV}} < \frac{\eta - 2 - \rho}{\log_2(\Theta) + 9}$$

Therefore, we have:

$$d_{\text{RNS}} < \frac{\log_2(\Theta) + 9}{\mathcal{V} + \log_2(n+t) + 6} \cdot d_{\text{SI-DGHV}}$$

For using the same security parameters $d_{\text{RNS}} \approx 0.39 \cdot d_{\text{SI-DGHV}}$. We note that the two schemes have different parameters affecting the noise growth– the RNS variant is mainly affected by ciphertext size whereas the SI-DGHV depends on the subset sum assumption. Accordingly, more optimization has to be done for the RNS variant's security parameters to improve its performance.

Lastly, the noise growth in the RNS.Convert$(*)$ procedure can be signaficantly reduced by using $BitDecomp_{\mathcal{V}}(*)$ and $PowerOfTwo_{\mathcal{V}}(*)$ techniques (as in the original scheme) where the noise growth will be around $\approx \log_2(\mathcal{V}) + \log_2(n+t) + 6$. The adoption will improve the variant multiplication depth to be $d_{\text{RNS}} \approx 0.72 \cdot d_{\text{SI-DGHV}}$.

## 4.12 Public key size

Here we briefly discuss the extra public key elements required for *SI-DGHV.Convert(∗)* and $RNS.Convert(*)$. In the original scheme [12], the *SI-DGHV.Convert(∗)* procedure requires two extra elements in the public key where $\boldsymbol{\sigma}$ is a matrix of size $\eta \times \Theta$ with each

entry being $< 2^{\gamma}$, and $\boldsymbol{z}$ is a vector of $\Theta$ elements, each of size $k = 2\gamma + 2$. Therefore, the required storage for both elements is at most $O(3\gamma + \log_2(\eta) + \log_2(\Theta))$. On the other hand, both (4.9) and (4.7) in the RNS variant require an storage of at most $O(\gamma + \log_2(n + t))$.

# Chapter 5

# Contributions and Future Work

The main goal of this thesis has been to enhance the practicality of fully homomorphic encryption over integers, by presenting an RNS variant to SI-DGHV of [12]. The work presented in this thesis can be summarized as follows:

- Presenting an overview of the mathematical definitions and properties of the residue number system that provides sufficient background to the RNS variant.
- Surveying various works on SWHE and FHE schemes, introducing a comprehensive description of DGHV and SI-DGHV, and analyzing their procedures to achieve FHE.
- Showing the basic concept of the RNS variant with analytic proof of target noise reduction, and proposing the RNS variant to *SI-DGHV.Convert*$(*)$ that does not rely on the subset-sum assumption as a security parameter.
- Providing a technical implementation of the RNS variant to [12] along with a mathematical analysis for correctness, ciphertext expansion, and noise growth. Moreover, giving a mathematical bound that restrains the number and the size of RNS moduli.

- Proposing an efficient technique to incorporate the Montgomery reduction in our RNS variant, and adopting the ciphertext batching technique for efficient utilization of the RNS variant.

- Introducing suitable Approximate-GCD assumption to our $RNS.Convert(*)$ procedure and providing a security proof of non-batched and batched RNA variants.

- Providing analytical estimation of complexity for both the RNS variant and the original scheme [12], doing the same for the $RNS.Convert(*)$ procedure's public key size, and showing the relative size reduction in comparison with $SI\text{-}DGHV.Convert(*)$ [12].

To sum up, the work in this thesis has introduced a new RNS variant to FHE of [12]. The new RNS variant scheme enhances the SI-DGHV scheme complexity and speeds up noise reduction procedures. In addition, the variant improves the original scheme's security by eliminating the need for the subset-sum assumption. However, in our variant, the noise grows linearly with the moduli size $\mathcal{V}$ which lowers the scheme's multiplication depth relative to the original scheme. In order to control the growth, the $RNS.Convert(*)$ noise management requires more optimization.

Below are some recommendations and suggestions for future research that would add to this work and contribute in general to HE:

- Adopting both $BitDecomp_{\mathcal{V}}(*)$ $PowerOfTwo_{\mathcal{V}}(*)$ procedures to squash the dot product $\langle *, * \rangle$ in Eq. (4.10) which might reduce the noise growth.

- The $RNS.Convert(*)$ procedure carries a relatively high noise growth compared to the original scheme that has to be optimized.

- Adopting an efficient public key compression as in [14].
- Adopting Gentry's bootstrapping technique in RNS.
- Further optimization for more practical implementation to speed up the $RNS.Convert(*)$ procedure.

# References


[1] Mohammad Abdallah and Alexander Skavantzos. A systematic approach for selecting practical moduli sets for residue number systems. In *System Theory, 1995., Proceedings of the Twenty-Seventh Southeastern Symposium on*, pages 445–449. IEEE, 1995.

[2] Frederik Armknecht and Ahmad-Reza Sadeghi. A New Approach for Algebraically Homomorphic Encryption. *IACR Cryptology ePrint Archive*, 2008:422, 2008.

[3] Jean-Claude Bajard, Julien Eynard, Anwar Hasan, and Vincent Zucca. A Full RNS Variant of FV like Somewhat Homomorphic Encryption Schemes. *IACR Cryptology ePrint Archive*, 2016:510, 2016.

[4] Jean-Claude Bajard and Laurent Imbert. A full RNS implementation of RSA. *IEEE Transactions on Computers*, 53(6):769–774, 2004.

[5] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF Formulas on Ciphertexts. In *Theory of Cryptography Conference, TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.

[6] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 309–325. ACM, 2012.

[7] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *Annual Cryptology Conference*, pages 505–524. Springer, 2011.

[8] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. *SIAM Journal on Computing*, 43(2):831–871, 2014.

[9] Jung Hee Cheon, Jinsu Kim, Moon Sung Lee, and Aaram Yun. CRT-based fully homomorphic encryption over the integers. *Inf. Sci.*, 310:149–162, 2015.

[10] WA Chren. A new residue number system division algorithm. *Computers & Mathematics with Applications*, 19(7):13–29, 1990.

[11] Jean-Sbastien Coron, Tancrde Lepoint, and Mehdi Tibouchi. Batch Fully Homomorphic Encryption over the Integers. *IACR Cryptology ePrint Archive*, 2013:36, 2013.

[12] Jean-Sbastien Coron, Tancrde Lepoint, and Mehdi Tibouchi. Scale-Invariant Fully Homomorphic Encryption over the Integers. *IACR Cryptology ePrint Archive*, 2014:32, 2014.

[13] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In *Annual Cryptology Conference*, pages 487–504. Springer, 2011.

[14] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *Annual*

*International Conference on the Theory and Applications of Cryptographic Techniques*, pages 446–464. Springer, 2012.

[15] Ivan Damgrd and Mads Jurik. A Length-Flexible Threshold Cryptosystem with Applications. In Reihaneh Safavi-Naini and Jennifer Seberry, editors, *ACISP*, volume 2727 of *Lecture Notes in Computer Science*, pages 350–364. Springer, 2003.

[16] Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[17] Ivan Flores. Residue Arithmetic and Its Application to Computer Technology (Nicholas S. Szabo and Richard I. Tanaka). *SIAM Review*, 11(1):103–104, 1969.

[18] Dragan Gamberger. New approach to integer division in residue number systems. In *IEEE Symposium on Computer Arithmetic*, pages 84–91. IEEE, 1991.

[19] Harvey L. Garner. The Residue Number System. *IRE Trans. Electronic Computers*, 8(2):140–147, 1959.

[20] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

[21] Craig Gentry. Toward Basing Fully Homomorphic Encryption on Worst-Case Hardness. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 116–137. Springer, 2010.

[22] Craig Gentry and Shai Halevi. Implementing Gentrys fully-homomorphic encryption scheme. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 129–148. Springer, 2011.

[23] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[24] Ahmad A Hiasat. RNS arithmetic multiplier for medium and large moduli. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 47(9):937–940, 2000.

[25] Ahmad A. Hiasat and Hoda S. Abdel-Aty-Zohdy. Design and Implementation of An RNS Division Algorithmm. In *IEEE Symposium on Computer Arithmetic*, pages 240–249. IEEE Computer Society, 1997.

[26] Nick Howgrave-Graham. Approximate Integer Common Divisors. In Joseph H. Silverman, editor, *CaLC*, volume 2146 of *Lecture Notes in Computer Science*, pages 51–66. Springer, 2001.

[27] Arjen K. Lenstra. Factoring Multivariate Polynomials over Algebraic Number Fields. *SIAM J. Comput.*, 16(3):591–598, 1987.

[28] Mi Lu and Jen-Shiun Chiang. A Novel Division Algorithm for the Residue Number System. *IEEE Trans. Computers*, 41(8):1026–1032, 1992.

[29] P.V.A. Mohan. *Residue Number Systems: Algorithms and Architectures*. The Springer International Series in Engineering and Computer Science. Springer US, 2002.

[30] D. Naccache and J. Stern. A New Public Key Cryptosystem Based on Higher Residues. *ACM Conference on Computer and Communications Security*, pages 59–66, 1998.

[31] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology - EUROCRYPT 99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, Heidelberg, May 1999. Springer.

[32] Behrooz Parhami. *Computer Arithmetic: Algorithms and Hardware Designs*. Oxford University Press, Oxford, UK, 2000.

[33] Karl C Posch and Reinhard Posch. Modulo reduction in residue number systems. *IEEE Transactions on Parallel and Distributed Systems*, 6(5):449–454, 1995.

[34] J Ramirez, A Garcia, S Lopez-Buedo, and A Lloris. RNS-enabled digital signal processor design. *Electronics Letters*, 38(6):1, 2002.

[35] Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. On Data Banks and Privacy Homomorphisms. *Foundations of secure computation*, pages 169–177, 1978.

[36] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[37] Tomas Sander, Adam L. Young, and Moti Yung. Non-Interactive CryptoComputing For NC1. In *FOCS*, pages 554–567. IEEE Computer Society, 1999.

[38] A. P. Shenoy and Ramdas Kumaresan. Fast Base Extension Using a Redundant Modulus in RNS. *IEEE Trans. Computers*, 38(2):292–297, 1989.

[39] Nigel P Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *International Workshop on Public Key Cryptography*, pages 420–443. Springer, 2010.

[40] Damien Stehl and Ron Steinfeld. Faster Fully Homomorphic Encryption. *IACR Cryptology ePrint Archive*, 2010:299, 2010.

[41] Damien Stehl and Paul Zimmermann. A Binary Recursive GCD Algorithm. In Duncan A. Buell, editor, *ANTS*, volume 3076 of *Lecture Notes in Computer Science*, pages 411–425. Springer, 2004.

[42] F Taylor. Large moduli multipliers for signal processing. *IEEE Transactions on circuits and systems*, 28(7):731–736, 1981.

[43] Ben-Dau Tseng, Graham A. Jullien, and William C. Miller. Implementation of FFT structures using the residue number system. *IEEE Transactions on Computers*, 100(11):831–845, 1979.

[44] ZD Ulman, M Czyzak, and JM Zurada. Fast division in residue arithmetic. In *Communications, Computers and Signal Processing, 1991., IEEE Pacific Rim Conference on*, pages 696–699. IEEE, 1991.

[45] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully Homomorphic Encryption over the Integers. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010.

[46] VA Vyshysky and VD Petushchak. Algorithms for Determination of the Reciprocal of a Number in a Residue Class System. *Soviet Automatic Control*, 6(3):58–61, 1973.