Homomorphic SIM^{2}D Operations: Single Instruction Much More Data

Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren

imec-Cosic, Dept. Electrical Engineering, KU Leuven
firstname.lastname@esat.kuleuven.be

Abstract. In 2014, Smart and Vercauteren introduced a packing technique for homomorphic encryption schemes by decomposing the plaintext space using the Chinese Remainder Theorem. This technique allows to encrypt multiple data values simultaneously into one ciphertext and execute Single Instruction Multiple Data operations homomorphically. In this paper we improve and generalize their results by introducing a flexible Laurent polynomial encoding technique and by using a more fine-grained CRT decomposition of the plaintext space. The Laurent polynomial encoding provides a convenient common framework for all conventional ways in which input data types can be represented, e.g. finite field elements, integers, rationals, floats and complex numbers. Our methods greatly increase the packing capacity of the plaintext space, as well as one's flexibility in optimizing the system parameters with respect to efficiency and/or security.

1 Introduction

Homomorphic encryption allows to perform arithmetic operations on encrypted data without decryption. The idea stems from [26] where the authors introduced so-called 'privacy homomorphisms' from plaintext space to ciphertext space. In 2009, Gentry [21] presented the first fully homomorphic encryption scheme (FHE) using ideal lattices. This breakthrough result was followed by several variants and improvements [8,9,6,7,20,23] all using the same blueprint. One first constructs a *somewhat* homomorphic encryption (SHE) scheme that can homomorphically evaluate arithmetic circuits of limited depth and then turns this into a fully homomorphic scheme using a bootstrapping procedure. The security of these schemes relies on the presence of a noise component in the ciphertexts. This noise grows during arithmetic operations and eventually reaches a threshold beyond which the ciphertext can no longer be decrypted correctly. The bootstrapping procedure basically reduces the inherent noise by executing

This work was supported by the European Commission under the ICT programme with contract H2020-ICT-2014-1 644209 HEAT, and through the European Research Council under the FP7/2007-2013 programme with ERC Grant Agreement 615722 MOTMELSUM. The first author thanks Ghent University for its hospitality. The authors also thank the anonymous referees for some helpful remarks.

the decryption circuit homomorphically. Despite considerable effort in making bootstrapping more efficient [19,13,11,2], full fledged FHE is still rather slow, so implementers typically resort to using SHE schemes for practical applications.

The efficiency of homomorphic encryption schemes can be improved significantly by a judicious choice of plaintext space and encoding techniques for the common data types such as finite field elements, integers, rationals, floats and complex numbers. Concretely, throughout this paper we assume that the plaintext space is a ring of the form

$$R_t = \mathbb{Z}_t[X]/(\overline{f}(X))$$

where $t \geqslant 2$ is an integer called the plaintext modulus, and $\overline{f}(X)$ is the reduction modulo t of a monic irreducible polynomial $f(X) \in \mathbb{Z}[X]$ of degree $d \geqslant 1$. This setting is valid for most SHE schemes whose security relies on the Ring-LWE problem.[1] The degree d together with the ciphertext modulus q and the standard deviation σ of the initial noise distribution are the main security parameters, and these are typically determined by the required security level. The noise growth is influenced by d, q, σ, but also by the plaintext modulus t. A first optimization to decrease the noise growth is therefore to use a smaller plaintext space. Several encoding techniques [25,18,14,12,3,10] have been proposed whose goal is to 'spread out' the numerical input data as evenly as possible over the whole plaintext space, allowing for a smaller value of t. A second optimization, which can be combined with the first, is to decompose the plaintext space into smaller pieces using the Chinese Remainder Theorem (CRT) and run several computations in parallel [27,4]. Smart and Vercauteren [27] described how to carry out SIMD calculations in an SHE context by viewing R_t as the CRT composition of

$$\mathbb{Z}_t[X]/(\overline{f}_1(X)) \times \mathbb{Z}_t[X]/(\overline{f}_2(X)) \times \cdots \times \mathbb{Z}_t[X]/(\overline{f}_r(X)),$$

where $\overline{f}_1(X)\overline{f}_2(X)\cdots\overline{f}_r(X)$ is a factorization of $\overline{f}(X)$ into coprime factors. In fact, they concentrate on the case $t = 2$, but the above immediate generalization is discussed in [22]. We will refer to this decomposition of R_t as a *vertical* slicing of the plaintext space.

Contributions. Our first contribution is an improvement of the above SIMD approach by utilizing a more fine-grained CRT decomposition of the plaintext space. We do this by also taking into account factorizations of the plaintext modulus t. We will refer to the CRT decomposition

$$R_t \;\cong\; \mathbb{Z}[X]/(t_1, f(X)) \times \mathbb{Z}[X]/(t_2, f(X)) \times \cdots \times \mathbb{Z}[X]/(t_s, f(X)),$$

[1] A recent adaptation of the FV scheme due to Chen et al. [10] uses as plaintext modulus a linear polynomial $x - a$ instead of an integer t. The resulting plaintext space $R_{x-a} = \mathbb{Z}[X]/(X^n + 1, x - a) \cong \mathbb{Z}/(a^n + 1)$ has various nice features, both in terms of noise growth and in terms of packing capacity. However, the algebraic structure of R_{x-a} becomes more restrictive for CRT decomposition, so rings of this type will not be considered in this paper.

corresponding to a factorization $t = t_1 t_2 \cdots t_s$ into coprime factors, as a *horizontal* slicing of the plaintext space. The flexibility of our method stems partly from the fact that factorisations modulo the various t_i do not imply a global factorisation modulo t. This alternative type of slicing for SIMD purposes is not new (see e.g. [4]). However, by *combining* horizontal and vertical slicing as explained in Section 4, the plaintext space becomes subdivided in 'bricks' as depicted in Figure 4. In our SIMD approach, which we call SIM^{2}D, each data slot corresponds to a set of such bricks (called a block) rather than one vertical or horizontal slice as considered in previous works. This results in a much more flexible but, at the same time, denser packing as described in Section 5. In Section 6 we provide several tools that can help in making an optimal choice of blocks. This includes slight alterations to t and/or $f(X)$ that lead to more fine-grained decompositions.

Our second contribution is a novel encoding technique for Laurent polynomials into a plaintext space of the form $R_t = \mathbb{Z}_t[X]/(\overline{f}(X))$ that works for general $\overline{f}$ (under the mild assumption that $\overline{f}(0)$ is an invertible element of $\mathbb{Z}_t$). Previous work [15] could only deal with the very special case of 2-power cyclotomic polynomials, due to concerns of mixing of integral and fractional parts. Our encoding technique is explained in Section 3. Encoding elements of the Laurent polynomial ring $\mathbb{Z}[X^{\pm 1}]$ serves as a convenient common framework for all customary encoding techniques: indeed, under $X \mapsto b$ the Laurent polynomials specialize to b-ary expansions for any choice of base $b \in \mathbb{C}\backslash\{0\}$. This framework allows to encode common data types such as finite field elements, integers, rationals, floats and complex numbers. Furthermore, we show that choosing different bases b for different blocks can be useful in optimizing the data packing (see Section 6).

Our algorithms for encoding, packing, unpacking and decoding are easy to implement (pseudo-code is provided) and extremely flexible to use. The overall goal is to provide a set of tools which together can be used to perform SIMD in an optimal way, given the constraints on the plaintext space imposed by security, efficiency and correctness requirements.

2 Preliminaries

2.1 Basic notation

Vectors are denoted by bold letters such as $\mathbf{a}$ and when the individual coordinates are required, we write a row vector as $(a_1, \ldots, a_k)$. For a natural number r, we denote the set $\{1, \ldots, r\}$ by $[r]$. Similarly, for any $\ell, m \in \mathbb{Z}, \ell \leqslant m$, the set $\{\ell, \ell+1, \ldots, m-1, m\}$ is denoted by $[\ell, m]$. The quotient ring of integers modulo a natural number t is denoted $\mathbb{Z}_t$.

2.2 Laurent polynomials

Most common numerical types (integers, rational, real or complex numbers) are represented as (finite) power series expansions in a certain base $b \in \mathbb{C}\backslash\{0\}$, using

digits that are taken from some given subset of $\mathbb{Z}$. These expansions naturally correspond to Laurent polynomials with integral coefficients, i.e. elements of the ring $\mathbb{Z}[X^{\pm 1}]$.

Most frequently, an integral base $b > 1$ with digit set $\{0, \ldots, b-1\}$ is used in practice, such as binary $b = 2$ or ternary $b = 3$. For use in SHE schemes, several variations [18,14,12,3] have been proposed. For the purposes of this paper we mention the non-integral base non-adjacent form (NIBNAF) from [3] which is a very sparse expansion with respect to a real base $b \in (1, 2)$ and using the digit set $\{-1, 0, 1\}$. All of these expansions can be thought of as the evaluations at $X = b$ of a Laurent polynomial with integral coefficients.

Example 1. The real number 2.3 can be approximated in base $b = 2$ using digits in $\{0, 1\}$ as

$$2.3 \simeq 1 \cdot 2 + 1 \cdot 2^{-2} + 1 \cdot 2^{-5} + 1 \cdot 2^{-6},$$

which is the evaluation of the Laurent polynomial

$$1 \cdot X + 1 \cdot X^{-2} + 1 \cdot X^{-5} + 1 \cdot X^{-6} \in \mathbb{Z}[X^{\pm 1}]$$

at $X = b = 2$.

Recall that in general, any Laurent polynomial $a(X) \in \mathbb{Z}[X^{\pm 1}]$ can be written as

$$a(X) = a_\ell X^\ell + \cdots + a_{m-1} X^{m-1} + a_m X^m \tag{1}$$

where $a_i \in \mathbb{Z}$ for every $i \in [\ell, m]$, $a_\ell, a_m \neq 0$ and $\ell \leqslant m$. For a modulus t (which will be clear from the context) we write $\overline{a}(X)$ for the Laurent polynomial in $\mathbb{Z}_t[X^{\pm 1}]$ obtained by reducing all coefficients.

Definition 1. *For an integral Laurent polynomial $a(X) \in \mathbb{Z}[X^{\pm 1}]$ represented as in Equation* (1)*, we define the* bounding box *of $a(X)$ as the tuple (w, h) with $w = m - \ell$ and $h = \log_2(\max_i a_i - \min_i a_i + 1)$ the sizes of the exponent and the coefficient ranges of $a(X)$.*

We represent the bounding box graphically with a rectangle of width w and height h.

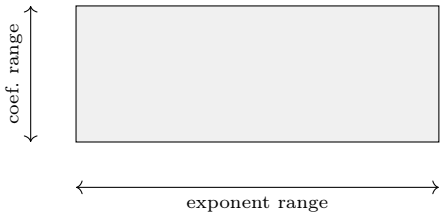

Fig. 1. The bounding box of a polynomial.

2.3 Plaintext space

Most SHE schemes utilize quotient rings of the form

$$R = \mathbb{Z}[X]/(f(X))$$

where $f(X) \in \mathbb{Z}[X]$ is a monic irreducible polynomial of degree d. The plaintext space is typically represented as a quotient ring $R_t = \mathbb{Z}_t[X]/(\overline{f}(X))$ for an integral plaintext modulus t. Similarly, the ciphertext space is defined as $R_q = \mathbb{Z}_q[X]/(\overline{f}(X))$ where $q \gg t$. Another important parameter is the standard deviation σ of the discretized Gaussian distribution from which the SHE encryption scheme samples its noise, which is embedded into the ciphertexts.

Typically, one first sets the parameters q, d and σ, primarily as functions of the security level, in order to prevent all known attacks on the underlying lattice problems [1]. Afterwards, the plaintext modulus t is selected, subject to two constraints. Firstly, it is bounded from above, which stems from the fact that the embedded noise grows during arithmetic operations up to a critical threshold above which ciphertexts can no longer be decrypted. Since the plaintext modulus directly affects the noise growth in ciphertexts, one can find a maximal t for which the decryption remains correct while evaluating a given arithmetic circuit $\mathcal{C}$. We denote this bound by $t_{\mathcal{C}}^{\max}$. If it is impossible to satisfy this bound then one can use the Chinese Remainder Theorem to split the computation into smaller parts, as explained in Remark 3; see also [4]. Secondly, as explained in the next section, the plaintext modulus t is bounded from below by some value $t_{\mathcal{C}}^{\min}$ which depends on the input data and on the way the latter is encoded, and which ensures correct decoding.

Remark 1. The values of q, d, σ are not uniquely determined by the security level. Therefore, one can try to use the remaining freedom to target a specific value of $t_{\mathcal{C}}^{\max}$. In the remainder of the paper, we will assume that $t_{\mathcal{C}}^{\max}$ is given, and our aim is to utilize the available plaintext space in an optimal way. One motivation for targeting maximal flexibility here is that it is not clear whether preselecting a precise value of $t_{\mathcal{C}}^{\max}$ is always possible in practice (e.g., for a fixed degree and security level it turns out that the value of $t_{\mathcal{C}}^{\max}$ stabilizes as $q \to \infty$). This is further impeded by the fact that concrete implementations often do not allow q and d to be picked from some continuous-like range (e.g., the `FV-NFLlib` [16] and the `SEAL` [24] libraries require that d is a power of 2 and that $\log_2 q$ is a multiple of some integer). A second motivation is that it can be desirable to use a single SHE implementation for encrypting batches of data of largely varying sizes. The plaintext space should be chosen to fit the largest data, and the methods presented below can then be used to optimize the handling of the smaller data.

The most common choice for $f(X)$ is a cyclotomic polynomial. The nth cyclotomic polynomial $\Phi_n(X) \in \mathbb{Z}[X]$ is the minimal polynomial of a primitive nth root of unity in $\mathbb{C}$

$$\Phi_n(X) = \prod_{0<k<n\,,\,(k,n)=1} (X - \zeta_n^k),$$

where $\zeta_n = e^{2\pi i/n}$. The degree of $\Phi_n(X)$ is equal to $\phi(n)$, where $\phi(n)$ is the totient function. It is always irreducible over $\mathbb{Z}$ and, additionally, $\Phi(0) = 1$ for $n \geqslant 3$.

Cyclotomic polynomials are often used by SHE implementers since they have very nice arithmetic properties such as fast modular reduction and simple Galois groups, which can be used to move data values in between data slots.

3 Plaintext encoding/decoding of Laurent polynomials

In this section we consider the problem of encoding an integral Laurent polynomial in the plaintext space and the reverse operation of decoding. We also give necessary conditions on the 'size' of the plaintext space such that a given circuit $\mathcal{C}$ can be evaluated correctly.

3.1 Encoding

Assume that the input data (integers, rationals, reals, ...) has been represented as a Laurent polynomial $a(X) \in \mathbb{Z}[X^{\pm 1}]$. Encoding such a Laurent polynomial in the plaintext space R_t has been considered in a series of recent works [18,14,12,3]. However, it was emphasized in [14] that the plaintext space should only be defined modulo a 2-power cyclotomic polynomial $f(X) = X^{2^k} + 1$ for some k. The reason for this restriction is that the authors required a small and sparse representation for X^{-1}, which in this case is given by $X^{-1} \equiv -X^{2^k-1} \bmod f(X)$.

Here we propose a very general way of encoding Laurent polynomials which works for almost all defining polynomials f. Let $\overline{f}(X)$ denote the reduction modulo t of $f(X)$ and assume that $f(0)$ is co-prime with t, so $\overline{f}(0)$ is invertible in $\mathbb{Z}_t$. Define $\overline{g}(X)$ by writing $\overline{f}(X) = \overline{g}(X)X + \overline{f}(0)$, then it is obvious that modulo $\overline{f}(X)$ we have that $X^{-1} \equiv -\overline{g}(X)\overline{f}(0)^{-1}$.

The encoding map $\mathsf{Encd}_{\overline{f}}$ is then given by the sequence of ring homomorphisms

$$\mathbb{Z}[X^{\pm 1}] \xrightarrow{\bmod t} \mathbb{Z}_t[X^{\pm 1}] \xrightarrow{\eta_{\overline{f}}} R_t$$

with

$$\eta_{\overline{f}} : \begin{array}{ll} X & \mapsto X \\ X^{-1} & \mapsto -\overline{g}(X)\overline{f}(0)^{-1}\,. \end{array}$$

Example 2. In the case of the 2-power cyclotomic polynomial $f(X) = X^{2^k} + 1$, the above map replaces negative powers X^{-j} by $-X^{d-j}$, which coincides with the approach from [14]: when expressed in terms of the basis $1, X, X^2, \ldots, X^{d-1}$ of R_t, the map $\eta_{\overline{f}}$ places the positive exponents at the low end of this range, and the negative exponents are placed at the high end.

3.2 Decoding

The crux of the construction relies on the fact that the above encoding map $\mathsf{Encd}_{\overline{f}}$ defines an isomorphism when restricted to a subset of Laurent polynomials. Indeed, if we choose a subset of $\mathbb{Z}_t[X^{\pm 1}]$ of the form

$$\mathbb{Z}_t[X^{\pm 1}]_\ell^m = \left\{ \sum_{i=\ell}^{m} \overline{a}_i X^i | \overline{a}_i \in \mathbb{Z}_t \right\}$$

with ℓ and m chosen such that $m - \ell + 1 = d$, then the restriction of $\eta_{\overline{f}}$ to $\mathbb{Z}_t[X^{\pm 1}]_\ell^m$ is an isomorphism between two free $\mathbb{Z}_t$-modules of rank d. The inverse of this map, denoted $\theta_{\overline{f},\ell,m}$, is easy to compute in practice, since it simply corresponds to a matrix inversion.

Thus, $\theta_{\overline{f},\ell,m}$ determines the decoding algorithm from R_t to Laurent polynomials over $\mathbb{Z}_t$. In the final step, one has to lift a Laurent polynomial from $\mathbb{Z}_t[X^{\pm 1}]$ to $\mathbb{Z}[X^{\pm 1}]$ by choosing a representative for each coefficient in a non-empty subset A of $\mathbb{Z}$ of size t. For simplicity we will always take $A = [z, z + t - 1]$ for some $z \in \mathbb{Z}$, common choices being $A = [-\lfloor (t-1)/2 \rfloor, \lceil (t-1)/2 \rceil]$ or $A = [0, t-1]$. But any set A of representatives would be possible, and in fact it can even depend on the coefficient under consideration. Together these two steps define the decoding map $\mathsf{Decd}_{\overline{f},\ell,m,A}$.

3.3 Correctness conditions

Since homomorphic encryption aims to perform arithmetic operations on ciphertexts, one usually deals with a ciphertext being the outcome of an arithmetic circuit involving only multiplications and additions. By the homomorphic property this ciphertext corresponds to a plaintext which is the result of the same operations in the plaintext space. Given a circuit $\mathcal{C}$, the result of its evaluation on encodings of Laurent polynomials $\mathbf{a} = (a_1(X), \ldots, a_k(X))$ is denoted by $\mathcal{C}(\mathsf{Encd}_{\overline{f}}(\mathbf{a})) \in R_t$.

To guarantee correctness of circuit evaluation, one has to make sure that there exist $\ell, m \in \mathbb{Z}$ such that $m - \ell + 1 = d$ and some non-empty set $A \subsetneq \mathbb{Z}$ of size at most t such that

$$\mathsf{Decd}_{\overline{f},\ell,m,A}(\mathcal{C}(\mathsf{Encd}_{\overline{f}}(\mathbf{a}))) = \mathcal{C}(\mathbf{a}),$$

where $\mathcal{C}(\mathbf{a})$ is the result of the same circuit evaluation in $\mathbb{Z}[X^{\pm 1}]$. This implies that the bounding box (w, h) of $\mathcal{C}(\mathbf{a})$ has to satisfy $w \leqslant m - \ell + 1 = d$ and $h \leqslant \log_2 |A| = \log_2 t$. In this case, we say that *the plaintext space covers the bounding box of* $\mathcal{C}(\mathbf{a})$ as shown on the figure below

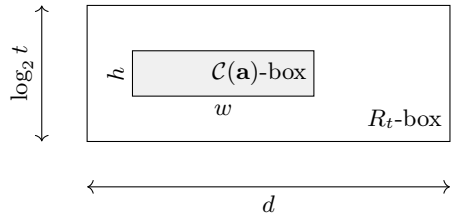

Fig. 2. The bounding box of $\mathcal{C}(\mathbf{a})$ is covered by the plaintext space R_t.

If the bounding box (w, h) of a Laurent polynomial has larger height than the plaintext space, i.e. $h > \log_2 t$, then we say that the computation *overflows modulo* t. If we end up with $w > d$ then we say that it *overflows modulo* $\overline{f}(X)$.

The parameters t and d should therefore be taken large enough to satisfy the above requirement. In practice, d is usually fixed by the security requirements of the SHE scheme. The choice for t, however, strongly depends on the arithmetic circuit $\mathcal{C}$ one is trying to evaluate. Initially, the input data of the circuit is encoded by Laurent polynomials whose bounding boxes are of height $h \leqslant \log_2(|\{b\text{-base digits}\}|)$. During arithmetic operations the height (typically) grows to the height of the bounding box of the outcome. For a given circuit $\mathcal{C}$, this defines a lower bound for t to guarantee correct decoding, which we denote $t_{\mathcal{C}}^{\min}$. Combined with the upper bound on t from Section 2.3, one obtains a range for t, namely $[t_{\mathcal{C}}^{\min}, t_{\mathcal{C}}^{\max}]$.

Example 3. To illustrate encoding and decoding, we take $R_t = \mathbb{Z}_7[X]/(\overline{f}(X))$ where $\overline{f}(X) = X^9 + 4X^7 + 1$. Thus, $\overline{g}(X) = X^8 + 4X^6$ and $\mathsf{Encd}_{\overline{f}}$ maps X^{-1} to $6X^8 + 3X^6$. Let us multiply two rational numbers, $\frac{182}{243}$ and 1476. Their base-3 expansions are as follows

$$\frac{182}{243} = 2 \cdot 3^{-5} + 2 \cdot 3^{-3} + 2 \cdot 3^{-1}, \qquad 1476 = 2 \cdot 3^2 + 2 \cdot 3^6$$

or as Laurent polynomials

$$a = 2X^{-5} + 2X^{-3} + 2X^{-1}, \qquad b = 2X^2 + 2X^6.$$

Applying $\mathsf{Encd}_{\overline{f}}$ we get encodings of a and b in R_t, namely, $\overline{a} = 6X^2 + 4X^4 + 4X^6 + 5X^8$ and $\overline{b} = 2X^2 + 2X^6$. Their product is equal to

$$\overline{c} = X + 4X^3 + 5X^4 + 4X^5 + X^6 + 3X^8.$$

We take $\ell = -3$, $m = 5$ and $A = \{4, 5, \ldots, 10\}$ in order to keep the product inside the box. Now we can define $\mathsf{Decd}_{\overline{f},-3,5,A}$. The first step is to construct a linear operator $\theta_{\overline{f},-3,5}$ using the inverse of the matrix defining the restriction of

$\eta_{\overline{f}}$ on $\mathbb{Z}_7[X^{\pm 1}]^5_{-3}$:

$$\begin{bmatrix} 0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&1&0&0 \\ 3&0&0&0&0&0&0&1&0 \\ 0&3&0&0&0&0&0&0&1 \\ 6&0&3&0&0&0&0&0&0 \\ 0&6&0&0&0&0&0&0&0 \\ 0&0&6&0&0&0&0&0&0 \end{bmatrix}^{-1} = \begin{bmatrix} 0&0&0&0&0&0&6&0&4 \\ 3&0&0&0&0&0&0&6&0 \\ 0&0&0&0&0&0&0&0&6 \\ 1&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&1&0&3&0&2 \\ 5&0&0&0&0&1&0&3&0 \end{bmatrix} \in \mathbb{Z}_7^{9\times 9}.$$

Then $\overline{c}$ is mapped to a Laurent polynomial $4X^{-3} + 4X^{-1} + X + 4X^3 + 4X^5 \in \mathbb{Z}_7[X^{\pm 1}]$. By looking for representatives of the coefficients in the set A we get $4X^{-3} + 4X^{-1} + 8X + 4X^3 + 4X^5 \in \mathbb{Z}[X^{\pm 1}]$ and evaluate it at $X = 3$

$$4 \cdot 3^{-3} + 4 \cdot 3^{-1} + 8 \cdot 3 + 4 \cdot 3^3 + 4 \cdot 3^5 = \frac{29848}{27},$$

which is the correct product of $\frac{182}{243}$ and 1476.

Remark 2. Note that the above condition for correct decoding *only* depends on the bounding box of the evaluation of the circuit $\mathcal{C}(\mathbf{a})$ and not on the bounding boxes of the individual inputs $a_i(X) \in \mathbb{Z}[X^{\pm 1}]$ nor on those of the intermediate values. Indeed, we always have

$$\mathcal{C}(\mathsf{Encd}_{\overline{f}}(a_1(X)), \ldots, \mathsf{Encd}_{\overline{f}}(a_k(X))) = \mathsf{Encd}_{\overline{f}}(\mathcal{C}(\mathbf{a})), \tag{2}$$

simply because $\mathsf{Encd}_{\overline{f}}$ is a ring homomorphism. This implies that the bounding boxes of the input or intermediate values should not necessarily be contained in the bounding box of the plaintext space, as long as the outcome of evaluation is.

4 Splitting the plaintext space

In this section we recall how the Chinese Remainder Theorem (CRT) can be used to split the plaintext space naturally along two directions: firstly, we will split horizontally for each prime power factor t_i of the plaintext modulus t and secondly, each horizontal slice will be split vertically by factoring $f(X) \bmod t_i$.

4.1 Horizontal splitting

If t is a composite that factors into distinct prime powers $t = t_1 \ldots t_s$ then the ring R_t can be mapped via the CRT to a direct product of R_{t_i}'s resulting in the following ring isomorphism

$$\begin{aligned} \mathsf{CRT}_t : R_t &\to R_{t_1} \times \cdots \times R_{t_s} \\ \overline{a}(X) &\mapsto (\overline{a}(X) \bmod t_1,\ \ldots, \overline{a}(X) \bmod t_s) \end{aligned}$$

whose inverse is easy to compute. For a given index subset $I = \{i_1, \ldots, i_c\} \subseteq [s]$ the map CRT_t induces a surjective morphism

$$\mathsf{CRT}_{t_I} : R_t \to R_{t_{i_1}} \times \cdots \times R_{t_{i_c}},$$

which is well-defined via the projection map

$$\pi_{t_I} : \prod_{i \in [s]} R_{t_i} \to \prod_{i \in I} R_{t_i}$$

so that $\mathsf{CRT}_{t_I} = \pi_{t_I} \cdot \mathsf{CRT}_t$. The CRT_t can be represented as a 'horizontal' splitting of the plaintext space according to the unique factorization of t into distinct prime powers $\{t_i\}_{i \in [s]}$. Each horizontal slice in Figure 3 corresponds to some R_{t_i}.

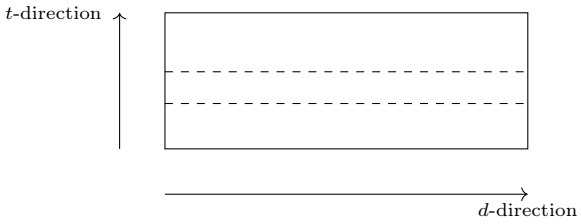

Fig. 3. CRT_t decomposition of R_t

4.2 Vertical splitting

For each factor t_i of t we define $\overline{f}_i(X) \in \mathbb{Z}_{t_i}[X]$ to be the reduction of $f(X)$ modulo t_i. Since $f(0)$ is co-prime with t, it is also co-prime with any t_i and thus, $\overline{f}_i(0)$ is invertible.

The factorization of $\overline{f}_i(X)$ into irreducible factors modulo t_i can be computed as follows: if t_i is prime, then one can simply use factorization algorithms for polynomials over finite fields; for t_i a prime power, one first computes the factorization modulo the prime and then lifts it using Hensel's lemma to a factorization modulo t_i. The result in both cases is that we can easily obtain a factorization

$$\overline{f}_i(X) \equiv \prod_{j=1}^{r_i} \overline{f}_{ij}(X)$$

for monic irreducible polynomials $\overline{f}_{ij}(X) \in \mathbb{Z}_{t_i}[X]$. Note that the constant terms $\overline{f}_{ij}(0)$ are all invertible because their product $\overline{f}_i(0)$ is invertible. Applying the CRT in the polynomial dimension gives the following map for each t_i:

$$\begin{aligned} \mathsf{CRT}_{t_i, \overline{f}_i} : R_{t_i} &\to R_{t_i,1} \times \cdots \times R_{t_i,r_i} \\ \overline{a}(X) &\mapsto (\overline{a}(X) \bmod \overline{f}_{i1}(X), \ldots, \overline{a}(X) \bmod \overline{f}_{ir_i}(X)). \end{aligned}$$

Here the $R_{t_i,j}$ denotes the ring $\mathbb{Z}_{t_i}[X]/(\overline{f}_{ij}(X))$, which corresponds to a 'brick' in Figure 4. The map $\mathsf{CRT}_{t_i,\overline{f}_i}$, whose inverse is again easy to compute, can be thought of as a 'vertical' splitting of R_{t_i}. For simplicity we will usually just write $R_{i,j}$ rather than $R_{t_i,j}$. By analogy with CRT_{t_I}, we introduce the surjective ring homomorphism $\mathsf{CRT}_{t_i,\overline{f}_J}$ from R_{t_i} to $\prod_{j\in J} R_{t_i,j}$ where $J = \{j_1, \ldots, j_c\} \subseteq [r_i]$.

5 Improved SIMD encoding

In this section we combine the results of Sections 3 and 4 to derive flexible SIMD encoding and decoding algorithms. Recall that to correctly decode the result of a circuit evaluation $\mathcal{C}(\mathbf{a})$, we require that the bounding box of the plaintext space covers the bounding box of $\mathcal{C}(\mathbf{a})$. We assume that this is indeed the case, and show how to select a minimal number of bricks of R_t to cover the bounding box of $\mathcal{C}(\mathbf{a})$, leaving the other bricks available for doing parallel computations.

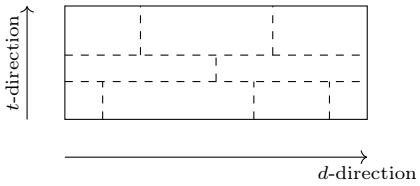

Fig. 4. Decomposition of R_t using factorization of t and $\overline{f}_i$'s

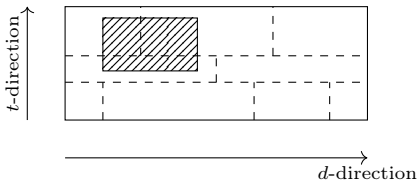

Fig. 5. Encoding of a single Laurent polynomial into R_t.

Recall that each brick corresponds to a ring $R_{i,j}$ in the decomposition

$$R_t \to R_{t_1} \times \cdots \times R_{t_s} \to (R_{1,1} \times \cdots \times R_{1,r_1}) \times \cdots \times (R_{s,1} \times \cdots \times R_{s,r_s}).$$

Each ring $R_{i,j}$ has its own bounding box of size $(d_{ij}, \log_2 t_i)$, where $d_{ij} = \deg \overline{f}_{ij}$. Assuming that the bounding box of $\mathcal{C}(\mathbf{a})$ is given by (w, h), we need to combine enough horizontal slices to cover the height h, and inside each horizontal slice, we need to select enough bricks to cover the width w as illustrated in Figure 5. Any unused bricks can be used to encode other data values, for instance to compute $\mathcal{C}(\mathbf{b})$ for some other input vector $\mathbf{b}$, immediately resulting in SIMD computations.

We formalize this approach by combining bricks into a block structure: we call a *block* a set of tuples $\mathcal{B} = \{(t_i, \overline{f}_{ij})\}_{i\in I(\mathcal{B}), j\in J(\mathcal{B},i)}$ with index sets $I(\mathcal{B}) \subseteq [s]$ and $J(\mathcal{B}, i) \subseteq [r_i]$, where we recall that r_i is the number of irreducible factors of $\overline{f}_i$. We of course think of this as corresponding to the set of $R_{i,j}$'s with $i \in I(\mathcal{B}), j \in J(\mathcal{B}, i)$. Equivalently, through an application of the CRT this corresponds to the set of quotient rings $\{R_{t_i}/(\overline{F}_{i,\mathcal{B}})\}_{i\in I(\mathcal{B})}$ where $\overline{F}_{i,\mathcal{B}} = \prod_{j\in J(\mathcal{B},i)} \overline{f}_{ij}$. Graphically we think of a block as a set of bricks of R_t, which are combined such that the

$R_{i,j}$'s with the same index i are glued column-wise and the resulting rows are placed on top of each other.

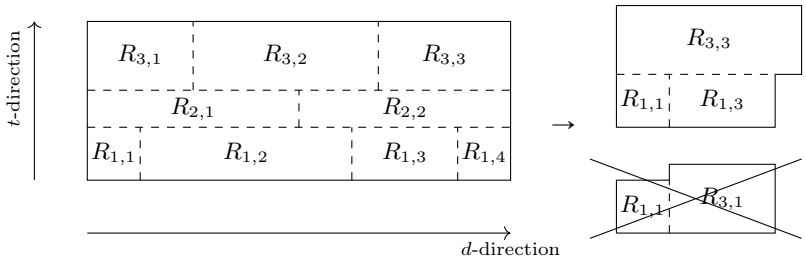

Fig. 6. Example of a block taken from the CRT decomposition of R_t. The bottom combination of 'bricks' is not a block because their first indices do not coincide.

In order for a block $\mathcal{B}$ to be suitable for computing $\mathcal{C}(\mathbf{a})$, whose bounding box we denote by (w, h), we note that the bounding box of $R_{t_i}/(\overline{F}_{i,\mathcal{B}})$ with $i \in I(\mathcal{B})$ is $(w_{i,\mathcal{B}}, \log_2 t_i)$ where

$$w_{i,\mathcal{B}} = \deg \overline{F}_{i,\mathcal{B}} = \sum_{j \in J(\mathcal{B},i)} d_{ij}.$$

If $\min_{i \in I(\mathcal{B})} w_{i,\mathcal{B}} \geqslant w$ and $\sum_{i \in I(\mathcal{B})} \log_2 t_i \geqslant h$ then we say that $\mathcal{B}$ *covers the bounding box* (w, h). As we will see $\mathcal{C}(\mathbf{a})$ will be decoded correctly as soon as an encoding block $\mathcal{B}$ is used that covers its bounding box.

Example 4. We decompose $R_t = \mathbb{Z}_{2761}[X]/(f(X))$ where $f(X) = X^{20} + X^{15} + 1$. The plaintext modulus factors into $t_1 = 11$ and $t_2 = 251$ and

$$\begin{aligned} f(X) &\equiv f_{1,1}(X) \cdot f_{1,2}(X) \\ &\equiv (X^5 + 3)(X^{15} + 9X^{10} + 6X^5 + 4) \bmod 11, \\ f(X) &\equiv f_{2,1}(X) \cdot f_{2,2}(X) \cdot f_{2,3}(X) \\ &\equiv (X^5 + 18)(X^5 + 120)(X^{10} + 114X^5 + 180) \bmod 251. \end{aligned}$$

Accordingly, R_t splits into $(R_{1,1} \times R_{1,2}) \times (R_{2,1} \times R_{2,2} \times R_{2,3})$. Overall we have 5 'bricks' that can be combined into 31 different blocks. For example, one can take a block $\{(11, X^{15}+9X^{10}+6X^5+4), (251, X^5+18), (251, X^5+120)\}$ corresponding to the combination of $R_{1,2}, R_{2,1}$ and $R_{2,2}$ or $\{(11, X^5 + 3), (11, X^{15} + 9X^{10} + 6X^5 + 4)\}$ which simply corresponds to $R_{11} = R_t/(11)$ (see Figure 7).

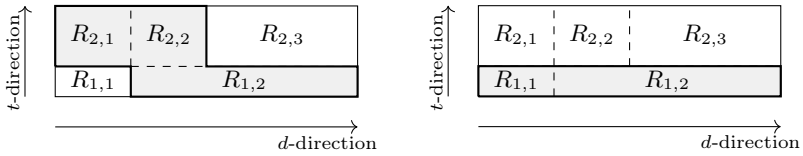

Fig. 7. The block structure of $R_t = \mathbb{Z}[X]/(2651, X^{20}+X^{15}+1)$ with two blocks colored in gray.

The whole plaintext space can be represented by a block as well

$$\mathcal{P} = \bigcup_{i\in[s]} \bigcup_{j\in[r_i]} \{(t_i, \overline{f}_{ij})\}.$$

Therefore, the SIMD packing problem consists in finding a set of disjoint blocks $S = \{\mathcal{B}_1, \ldots, \mathcal{B}_u\}$ such that $\bigcup_{\mathcal{B}\in S} \mathcal{B} = \mathcal{P}$ and every block covers the maximal bounding box among the corresponding output values.

To a partition S of $\mathcal{P}$ there naturally corresponds a factorization of $\overline{f}_i$ for every $i \in [s]$:

$$\overline{f}_i(X) = \prod_{\mathcal{B}\in S, i\in I(\mathcal{B})} \overline{F}_{i,\mathcal{B}}(X).$$

This induces a family of CRT isomorphisms

$$\mathsf{CRT}_{t_i,\overline{f}_i,S} : R_{t_i} \to \prod_{\mathcal{B}\in S, i\in I(\mathcal{B})} R_{t_i}/(\overline{F}_{i,\mathcal{B}}).$$

Now we have all the ingredients to pack a number of data values into one plaintext as described in Algorithm 1.

Algorithm 1: Plaintext packing.

Input : a set of disjoint blocks $S = \{\mathcal{B}_1, \ldots, \mathcal{B}_u\}$ with corresponding data values $a_1, \ldots, a_u \in \mathbb{Z}[X^{\pm 1}]$ such that $\bigcup_{k=1}^{u} \mathcal{B}_k = \mathcal{P}$.

Output: $b \in R_t$

```
for k ← 1 to u do
    for i ∈ I(B_k) do
        a_{t_i, F̄_{i,B_k}} ← Encd_{F̄_{i,B_k}}(a_k)
for i ← 1 to s do
    b_i ← CRT^{-1}_{t_i, f̄_i, S}({a_{t_i, F̄_{i,B}}}_{B, i∈I(B)})
b ← CRT_t^{-1}(b_1, ..., b_s)
```

After packing one can encrypt the output and feed it to an arithmetic circuit (together with other packings in case the circuit takes more than one argument). The resulting plaintext contains multiple evaluations corresponding to each block that can be decoded using Algorithm 2.

Algorithm 2: Plaintext decoding for one block.

Input : a plaintext $\overline{c} \in R_t$, a block $\mathcal{B}$, an exponent range $[\ell, m]$ and a coefficient set $A \in \mathbb{Z}$

Output: a Laurent polynomial $a \in \mathbb{Z}[X^{\pm 1}]$

$t_I \leftarrow 1$
for $i \in I(\mathcal{B})$ **do**
  $t_I \leftarrow t_I \cdot t_i$
  $\overline{c}_i \leftarrow \overline{c} \bmod t_i$
  $\overline{c}_i \leftarrow \overline{c}_i \bmod \overline{F}_{i,\mathcal{B}}$
  $m_i \leftarrow \ell + w_{i,\mathcal{B}} - 1$
  $c_i \leftarrow \theta_{\overline{F}_{i,\mathcal{B}},\ell,m_i}(\overline{c}_i)$
$a \leftarrow$ coefficient-wise CRT^{-1} of $\{c_i\}_{i \in I(\mathcal{B})}$ to $\mathbb{Z}_{t_I}[X^{\pm 1}]$
$a \leftarrow$ selecting coefficient representatives of a from the set A

Algorithm 2 produces correct circuit evaluations for all blocks occurring in Algorithm 1 that satisfy the properties outlined in the next theorem.

Theorem 1. *Let S be a set of disjoint blocks such that $\bigcup_{\mathcal{B} \in S} \mathcal{B} = \mathcal{P}$. Let $\mathcal{C}$ be an arithmetic circuit taking v arguments and for each block $\mathcal{B}$ let $\mathbf{a}_\mathcal{B} = (a_{\mathcal{B},1}, \ldots, a_{\mathcal{B},v})$ be a vector of Laurent polynomials. For each $k = 1, \ldots, v$ let b_k denote the output of Algorithm 1 upon input of $(a_{\mathcal{B},k})_{\mathcal{B} \in S}$. Let $\overline{c} = \mathcal{C}(b_1, \ldots, b_v)$. Then for each block $\mathcal{B}$ we have that if it covers the bounding box of $\mathcal{C}(\mathbf{a}_\mathcal{B})$, then upon input of $\overline{c}$ Algorithm 2 produces $\mathcal{C}(\mathbf{a}_\mathcal{B})$, for an appropriate choice of ℓ, m and A.*

Proof. By our assumption there are ℓ, m such that $\mathcal{C}(\mathbf{a}_\mathcal{B}) = \sum_{i=\ell}^{m} \alpha_i X^i$ where

$$\min_{i \in I(\mathcal{B})} w_{i,\mathcal{B}} \geqslant m - \ell + 1 \qquad \text{and} \qquad \prod_{i \in I(\mathcal{B})} t_i \geqslant |A|, \tag{3}$$

with $A = \{\min_i \alpha_i, \ldots, \max_i \alpha_i\}$. Let a denote the output of Algorithm 2 upon input of $\overline{c}$ using these ℓ, m, and A. Since this is a Laurent polynomial having coefficients in A, by (3) it suffices to prove that the reductions of a and $\mathcal{C}(\mathbf{a}_\mathcal{B})$ modulo t_i are the same for each $i \in I(\mathcal{B})$. Again by (3) these reductions are contained in $\mathbb{Z}_{t_i}[X^{\pm 1}]_\ell^{m_i}$ where $m_i = \ell + w_{i,\mathcal{B}} - 1$, so by injectivity of $\eta_{\overline{F}_{i,\mathcal{B}}}$ it suffices to prove that

$$\mathsf{Encd}_{\overline{F}_{i,\mathcal{B}}}(a) = \mathsf{Encd}_{\overline{F}_{i,\mathcal{B}}}(\mathcal{C}(\mathbf{a}_\mathcal{B})).$$

From Algorithm 2 we see that the left-hand side is just the reduction of $\overline{c}$ into $R_{t_i}/(\overline{F}_{i,\mathcal{B}})$, while the right hand side is

$$\mathcal{C}(\mathsf{Encd}_{\overline{F}_{i,\mathcal{B}}}(a_{\mathcal{B},1}), \ldots, \mathsf{Encd}_{\overline{F}_{i,\mathcal{B}}}(a_{\mathcal{B},v}))$$

because of the homomorphic properties of the encoding map. From Algorithm 1 we clearly see that $\mathsf{Encd}_{\overline{F}_{i,\mathcal{B}}}(a_{\mathcal{B},k})$ is the reduction of b_k into $R_{t_i}/(\overline{F}_{i,\mathcal{B}})$, for all $k = 1, \ldots, v$, so the theorem follows. ▫

Example 5. Using the CRT decomposition of R_t from Example 4 we cube two Laurent polynomials simultaneously using SIMD, namely $u(X) = 7X^3 + 7X^2$ and $v(X) = 8X^5 + 7X$. To encode u^3 we take the block $\mathcal{B}_1$ with rings $R_{1,1}, R_{2,1}$ and the remaining bricks to build the block $\mathcal{B}_2$ to hold the result v^3.

Since only positive exponents are present in the data, all encoding functions $\mathsf{Encd}_{\overline{F}_{i,\mathcal{B}_1}}$ and $\mathsf{Encd}_{\overline{F}_{i,\mathcal{B}_2}}$ map $u(X)$ and $v(X)$ identically to the corresponding $R_{i,j}$'s. Then we get

$$\begin{aligned}
&a_{11,\overline{F}_{1,\mathcal{B}_1}}(X) = 7X^3 + 7X^2 \ \in R_{1,1} = R_{11}/(X^5+3),\\
&a_{251,\overline{F}_{2,\mathcal{B}_1}}(X) = 7X^3 + 7X^2 \in R_{2,1} = R_{251}/(X^5+18),\\
&a_{11,\overline{F}_{1,\mathcal{B}_2}}(X) = 8X^5 + 7X \quad \in R_{1,2} = R_{11}/(X^{15}+9X^{10}+6X^5+4),\\
&a_{251,\overline{F}_{2,\mathcal{B}_2}}(X) = 8X^5 + 7X \ \in R_{2,2} \times R_{2,3} \cong R_{251}/(X^{15}+234X^{10}+55X^5+14).
\end{aligned}$$

Applying $\mathsf{CRT}^{-1}_{t_i,\overline{f}_i,\{\mathcal{B}_1,\mathcal{B}_2\}}$ for each t_i we find

$$\begin{aligned}
b_1 = {} & X^{18} + X^{17} + 10X^{16} + 5X^{15} + 9X^{13} + 9X^{12}\\
& +2X^{11} + X^{10} + 6X^8 + 6X^7 + 5X^6 + 5X^5 + 4X^3 + 4X^2 + 3X + 9 \in R_{11},\\
b_2 = {} & 162X^{18} + 162X^{17} + 89X^{16} + 213X^{15} + 7X^{13} + 7X^{12}\\
& +244X^{11} + 144X^{10} + 125X^8 + 125X^7 + 126X^6 + 177X^5\\
& +9X^3 + 9X^2 + 249X + 221 \in R_{251},
\end{aligned}$$

which finally leads to the following plaintext via CRT_t^{-1}

$$\begin{aligned}
b = {} & 2421X^{18} + 2421X^{17} + 340X^{16} + 1468X^{15} + 2517X^{13} + 2517X^{12}\\
& +244X^{11} + 144X^{10} + 2635X^8 + 2635X^7 + 126X^6 + 2436X^5\\
& +2017X^3 + 2017X^2 + 751X + 1978 \in R_{2761}.
\end{aligned}$$

Now we evaluate an arithmetic circuit $z \mapsto z^3$ in b and obtain

$$\begin{aligned}
\overline{c} = {} & 1943X^{19} + 401X^{18} + 745X^{17} + 391X^{16} + 433X^{15}\\
& +2109X^{14} + 1717X^{13} + 2646X^{12} + 2729X^{11} + 2347X^{10}\\
& +2198X^9 + 1724X^8 + 234X^7 + 421X^6 + 2683X^5 + 94X^4\\
& +1188X^3 + 1143X^2 + 1960X + 1906 \in R_{2761},
\end{aligned}$$

which simultaneously encodes u^3 and v^3.

In order to decode the data we apply Algorithm 2 starting with the block $\mathcal{B}_1$ equipped with the exponent range $[6, 9]$ and the coefficient set $A_{\mathcal{B}_1} = [0, 2760]$. At first, we should reduce $\overline{c}$ modulo $\overline{F}_{i,\mathcal{B}_1}$ and t_i for each $i \in I(\mathcal{B}_1)$. As a result, we find

$$\begin{aligned}
&\overline{c}_{1,\mathcal{B}_1} = 5X^4 + 4X^3 + 4X^2 + 5X \in R_{11}/(X^5+3),\\
&\overline{c}_{2,\mathcal{B}_1} = 101X^4 + 52X^3 + 52X^2 + 101X \in R_{251}/(X^5+18).
\end{aligned}$$

To decode into Laurent polynomials we set $\ell_i = 6$ and $m_i = 10$ for every $i \in I(\mathcal{B}_1)$ because $\deg \overline{F}_{1,\mathcal{B}_1} = \deg \overline{F}_{2,\mathcal{B}_1} = 5$. Then we follow the same procedure as in Example 3 to define $\theta_{\overline{F}_1,6,10}$ and $\theta_{\overline{F}_2,6,10}$ via matrices $M_1 = 7 \cdot M$ and

$M_2 = 237 \cdot M$ where

$$M = \begin{bmatrix} 0\,1\,0\,0\,0 \\ 0\,0\,1\,0\,0 \\ 0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,1 \\ 1\,0\,0\,0\,0 \end{bmatrix}.$$

These linear transformations give us two Laurent polynomials modulo 11 and 251, respectively

$$\begin{aligned} c_{1,\mathcal{B}_1} &= 2X^9 + 6X^8 + 6X^7 + 2X^6 \in \mathbb{Z}_{11}[X^{\pm 1}], \\ c_{2,\mathcal{B}_1} &= 92X^9 + 25X^8 + 25X^7 + 92X^6 \in \mathbb{Z}_{251}[X^{\pm 1}]. \end{aligned}$$

Using the coefficient-wise CRT and lifting coefficients in $A_{\mathcal{B}_1}$ we recover the Laurent polynomial

$$a_{\mathcal{B}_1} = 343X^9 + 1029X^8 + 1029X^7 + 343X^6 \in \mathbb{Z}[X^{\pm 1}],$$

which is equal to u^3.

We repeat the same steps for the block $\mathcal{B}_2$ with the exponent range $[3, 15]$ and the same coefficient set A. This block has again the polynomials $\overline{F}_{i,\mathcal{B}_2}$ of the same degree and thus every $m_i = 17$ and $\ell_i = 3$. Executing Algorithm 2 we get the following sequence of calculations

$$\begin{aligned} &\overline{c}_{1,\mathcal{B}_2} = 2X^{11} + X^{10} + 10X^7 + 8X^5 + 2X^3 + 9, \\ &\overline{c}_{2,\mathcal{B}_2} = 89X^{11} + 170X^{10} + 172X^7 + 203X^5 + 92X^3 + 111, \\ &\downarrow \\ &c_{1,\mathcal{B}_2} = 6X^{15} + 2X^{11} + 10X^7 + 2X^3, \\ &c_{2,\mathcal{B}_2} = 10X^{15} + 89X^{11} + 172X^7 + 92X^3, \\ &\downarrow \\ &a_{\mathcal{B}_2} = 512X^{15} + 1344X^{11} + 1176X^7 + 343X^3. \end{aligned}$$

The last polynomial is exactly v^3 so we correctly cubed two Laurent polynomials.

Remark 3. The CRT factorization can also be exploited when a homomorphic algorithm needs a bigger plaintext modulus than the upper bound $t_{\mathcal{C}}^{\max}$ discussed above. Let us denote this modulus with a capital T to emphasize direct incompatibility of this parameter with other SHE parameters, namely, $T > t_{\mathcal{C}}^{\max}$. However, one can find a set of natural numbers $\{T_i \leqslant t_{\mathcal{C}}^{\max}\}$ such that $T \leqslant T' = \prod_i T_i$. Then $R_{T'}$ splits into smaller quotient rings R_{T_i}. A plaintext $a \in R_{T'}$ then maps to a vector whose ith component lies in R_{T_i}. In that case the plaintext space splits into quotient rings with smaller moduli via CRT such that each ring fits the SHE settings according to the following diagram

$$R_{T'} \xrightarrow{\mathsf{CRT}} \begin{cases} R_{T_1} \xrightarrow{\mathsf{CRT}} \prod_{t'|T_1} \prod_{f'|\overline{f} \bmod t'} R_{t',f'} \xrightarrow{\text{Alg 1}} R_{T_1} \\ \dots \\ R_{T_s} \xrightarrow{\mathsf{CRT}} \prod_{t'|T_s} \prod_{f'|\overline{f} \bmod t'} R_{t',f'} \xrightarrow{\text{Alg 1}} R_{T_s} \end{cases}$$

A homomorphic circuit evaluation must then be repeated over each CRT factor T_i. Nevertheless, this gives some freedom of choice for T_i's so as to find $R_{T'}$ with a nice CRT decomposition.

6 Parameter choice

In this section we discuss a set of tools that will allow implementers to benefit from our enhanced SIMD approach as much as possible. There are three parameters that directly affect the packing capacity. We list them below in an order that seems natural for solving any packing problem. Nevertheless, all parameters depend on each other.

Plaintext modulus. Earlier we defined the range $[t_{\mathcal{C}}^{\min}, t_{\mathcal{C}}^{\max}]$ from which the plaintext modulus t is allowed to be chosen. Additionally, at the end of Section 4 we discussed the CRT trick that allows to handle plaintext moduli that are bigger than $t_{\mathcal{C}}^{\max}$. Altogether this gives a designer some freedom to choose t such that it splits into many 'advantageous' t_i's. An 'advantageous' t_i means that the factorization of $\overline{f}_i$ is such that the resulting CRT decomposition can embed as many plaintexts as possible, which is usually facilitated by a finer brick structure as in Figure 8.

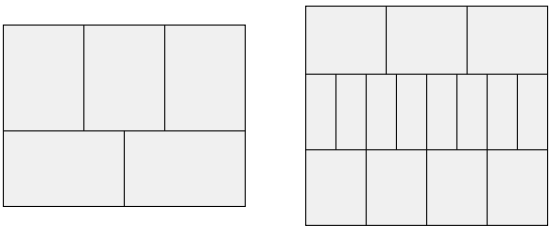

Fig. 8. The CRT decompositions of plaintext spaces corresponding to different t's.

This brick structure is defined by the t_i's and by the degrees of the $\overline{f}_{ij}$'s, namely $d_{i1}, \ldots, d_{ir_i}$ which constitute *a decomposition type* of f modulo t_i. Let G be the Galois group of the splitting field of f over $\mathbb{Q}$. It can be considered as a subgroup of the group S_d of permutations of d elements. Every automorphism σ can be represented as a product of cycle permutations with a corresponding pattern of cycle lengths. Additionally, we say that a set P of prime numbers has density δ if

$$\lim_{x \to \infty} \frac{|\{p \leqslant x : p \in P\}|}{|\{p \leqslant x : p \text{ prime}\}|} = \delta.$$

Then the probability that a desired decomposition type occurs for some random t_i is estimated by the following classical theorem.

Theorem 2 (Frobenius). *The density of the set P of primes modulo which f has a given decomposition type $d_1, d_2, \ldots, d_r$ exists, and it is equal to $1/|G|$ times the number of automorphisms $\sigma \in G$ with cycle pattern $d_1, d_2, \ldots, d_r$.*

An interesting case is where $\overline{f}_i$ splits into linear factors since it gives maximal flexibility to combine blocks. There exists only one $\sigma \in G$ corresponding to such a decomposition which is the identity permutation, so the corresponding probability is $1/|G|$.

Example 6. If $f(X)$ is the nth cyclotomic polynomial then its Galois group G has $d = \phi(n)$ elements and it always splits into irreducible factors of the same degree, i.e. its decomposition type modulo t_i is always $(d', \ldots, d')$ where d' is the order of t_i modulo n; here we implicitly assume that $\gcd(t_i, n) = 1$. Let us take $f(X) = X^{2^k} + 1$. Its Galois group is isomorphic to $\mathbb{Z}^{\times}_{2^{k+1}}$ or to the direct product of two cyclic groups $C_2 \times C_{2^{k-1}}$. It contains 2^k elements with orders shown in the following table:

ord	1	2	4	...	2^{k-1}
$\#\{a \in \mathbb{Z}^{\times}_{2^{k+1}}\}$	1	3	4	...	2^{k-1}

This implies that f splits into $2^{k'}$ irreducible factors of degree $2^{k-k'}$ modulo a random t_i with probability $2^{-k'}$, for any $k' \in \{1, \ldots, k-2, k-1\}$.

In the classical example of a homomorphic application a client encrypts his data and sends it to a third party to perform calculations. Since encryption and decryption are done only on the client side, he therefore has the possibility to tweak the plaintext modulus without re-generation of keys as long as the evaluation (or linearization) key does not depend on t. It is important to note that the plaintext modulus does not affect the security level of an SHE scheme but it does affect the decryption correctness. Hence, t should fit the upper bound $t_{\mathcal{C}}^{\max}$ introduced by the noise growth inside ciphertexts. As a result, one can exploit the same technique as above to find R_t with the most useful decomposition.

Block set. Recall that the plaintext space can be thought of as a set of bricks $\mathcal{P}$. Every block is then a subset of $\mathcal{P}$. The packing problem consists in finding a partition of $\mathcal{P}$ with the maximal number of blocks where each one satisfies Theorem 1. It is clear that the partition search is highly dependent on the data values and the arithmetic operations being performed homomorphically. Therefore the same plaintext space can be used differently for various applications as shown in Figure 9. If $r = \sum_{i=1}^{s} r_i$ is the cardinality of $\mathcal{P}$ then the total number of partitions is equal to the r-th Bell number B_r. That number grows exponentially (see [17]) while r is increasing according to

$$\frac{\ln B_r}{r} \simeq \ln r.$$

As a result a system designer has a lot of flexibility to play with the plaintext space partitions to fit data into some block structure. Obviously, the maximal

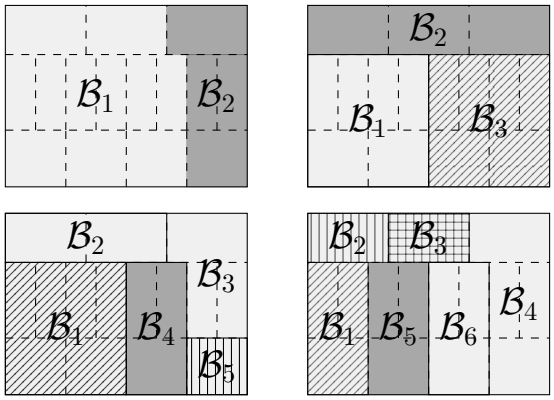

Fig. 9. Different partitions of $\mathcal{P}$.

number of blocks cannot be bigger than r, in which case the blocks are just the singletons $\{R_{i,j}\}$. A plaintext space with many CRT factors is usually easier to handle because it is more flexible for block constructions.

If one does not find a satisfying partition of all of $\mathcal{P}$, it is of course also possible to leave a couple of bricks unused by packing zeros in them (or even random values).

Encoding base. Representing data using Laurent polynomials requires a numerical base b which can be a real or a complex number. The size of b affects the length of a representation as well as the size of its coefficients.

In [3] it was shown that non-integral bases taken from the interval $(1, 2)$ have a simple greedy algorithm that, given a real number, produces a base-b expansion with a ternary set of coefficients. This procedure has the property that smaller bases lead to sparser representations and thus smaller coefficient growth but longer expansions. To illustrate this we resort again to the box representation of a Laurent polynomial.

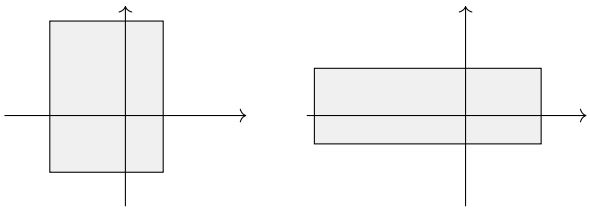

Fig. 10. The examples of bounding boxes corresponding to different encoding bases.

As a result, by changing the encoding base one could play a trade-off game between degree and coefficient size such that the number of plaintexts fitting a block structure is maximal. Furthermore, each block allows to encode data in a different base because neither Algorithm 1 nor Algorithm 2 depends on the choice of b.

Example 7. To illustrate the aforementioned techniques we revisit a medical application of the YASHE homomorphic encryption scheme [4] given in [5]. In this paper the standard logistic function is homomorphically computed to predict the probability of having a heart attack. The algorithm is divided into two steps.
Step 1. One computes the following weighted sum of six encrypted predictive variables

$$z = 0.072 \cdot z_1 + 0.013 \cdot z_2 - 0.029 \cdot z_3 + 0.008 \cdot z_4 - 0.053 \cdot z_5 + 0.021 \cdot z_6,$$

where each $z_i \in [0, 400]$. The multiplicative depth of the corresponding circuit is 1. For this step we take the same YASHE parameters as in [5], i.e. $q \simeq 2^{128}$ and $f(X) = X^{4096} + 1$. Given these parameters we derive $t^{\mathcal{C}}_{\max} = 2097152 \simeq 2^{21}$ using [4, Lem. 9]. Running over all primes less than $t^{\mathcal{C}}_{\max}$ we find that modulo $t_1 = 257$ and modulo $t_2 = 3583$ our polynomial $f(X)$ can be written as a product of 128 coprime factors of degree 32. With $t = t_1$ the conventional SIMD technique allows then to pack at most 128 values into one plaintext. This capacity can be achieved with base-3 balanced ternary expansions that result in an output bounding box of size $(29, \log_2 53)$. However, our approach supports $t = t_1 \cdot t_2$ so one can pack 256 values using the same encoding method.
Step 2. The output of Step 1 is decrypted, decoded to a real number and encoded again to a plaintext. This 'refreshed' encoding is then encrypted and given as input to the following approximation of the logistic function

$$P(x) = \frac{1}{2} + \frac{1}{4}x - \frac{1}{48}x^3 + \frac{1}{480}x^5 - \frac{17}{80640}x^7.$$

In this step the multiplicative depth is 3, $q \simeq 2^{512}$ and $f(X) = X^{16384} + 1$. These parameters lead to $t^{\mathcal{C}}_{\max} \simeq 2^{50}$. Using the previous SIMD technique the maximal plaintext capacity can be achieved with the plaintext modulus $t \simeq 2^{30.54}$ and base-3 balanced ternary encoding. In this case $f(X)$ splits into 8192 quadratic factors and the output bounding box is of size $(229, 29.54)$. We can thus compose 71 blocks with 115 slots and one block with the remaining slots. As a result, one plaintext can contain at most 71 values.

This capacity can be increased with our SIM^{2}D technique. In particular, one can notice that the ratio between $t^{\mathcal{C}}_{\max}$ and the previously mentioned modulus t is around $2^{19.46}$, which implies some part of the plaintext space remains unfilled. We can fill that space setting the plaintext modulus to $t_1 \cdot t_2$ with $t_1 \simeq 2^{30.54}$ and $t_2 = 675071 \simeq 2^{19.36}$. The polynomial $f(X)$ splits into 128 factors of degree 128 modulo t_2. To fit the modulus t_2 we encoded real values with the non-integral base $b = 1.16391$ and obtained the output bounding box $(1684, 19.36)$. Therefore one block should consist of 14 slots, and we can construct 9 such blocks. As a result, we can combine these blocks with the 71 blocks given by the old SIMD technique, which results in a total plaintext capacity of 80 values.

7 Conclusion

In this paper we presented two techniques that make SIMD operations in the setting of homomorphic encryption more flexible and efficient. Our first technique showed how data values that are naturally represented as Laurent polynomials can be encoded into a plaintext space of the form $\mathbb{Z}_t[X]/(\overline{f}(X))$. Furthermore, we also provided sufficient conditions for correct decoding after evaluation of an arithmetic circuit. Our second technique relied on a fine-grained CRT decomposition of the plaintext space resulting in a much denser and thus more efficient data packing compared to the state of the art. Finally, we provided guidelines on how to choose system parameters in order to find the most efficient packing strategy for a particular task.

References

1. Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.
2. Fabrice Benhamouda, Tancrède Lepoint, Claire Mathieu, and Hang Zhou. Optimization of bootstrapping in circuits. In Philip N. Klein, editor, *28th SODA*, pages 2423–2433. ACM-SIAM, 2017.
3. Charlotte Bonte, Carl Bootland, Joppe W. Bos, Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. Faster homomorphic function evaluation using non-integral base encoding. In Wieland Fischer and Naofumi Homma, editors, *CHES 2017*, volume 10529 of *LNCS*, pages 579–600. Springer, Heidelberg, 2017.
4. Joppe W. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In Martijn Stam, editor, *14th IMA International Conference on Cryptography and Coding*, volume 8308 of *LNCS*, pages 45–64. Springer, Heidelberg, 2013.
5. Joppe W. Bos, Kristin E. Lauter, and Michael Naehrig. Private predictive analysis on encrypted medical data. *Journal of Biomedical Informatics*, 50:234–243, 2014.
6. Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886. Springer, Heidelberg, 2012.
7. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, 2012.
8. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, 2011.
9. Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524. Springer, Heidelberg, 2011.
10. Hao Chen, Kim Laine, Rachel Player, and Yuhou Xia. High-precision arithmetic in homomorphic encryption. In Nigel P. Smart, editor, *CT-RSA 2018*, volume 10808 of *LNCS*. Springer, Heidelberg, 2018. To appear.
11. Jung Hee Cheon, Kyoohyung Han, and Duhyeong Kim. Faster bootstrapping of FHE over the integers. Cryptology ePrint Archive, Report 2017/079, 2017. `http://eprint.iacr.org/2017/079`.

12. Jung Hee Cheon, Jinhyuck Jeong, Joohee Lee, and Keewoo Lee. Privacy-preserving computations of predictive medical models with minimax approximation and non-adjacent form. In Michael Brenner, Kurt Rohloff, Joseph Bonneau, Andrew Miller, Peter Y. A. Ryan, Vanessa Teague, Andrea Bracciali, Massimiliano Sala, Federico Pintore, and Markus Jakobsson, editors, *FC 2017*, volume 10323, pages 53–74. Springer, Heidelberg, 2017.
13. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 3–33. Springer, Heidelberg, 2016.
14. Anamaria Costache, Nigel P. Smart, and Srinivas Vivek. Faster homomorphic evaluation of discrete Fourier transforms. In Aggelos Kiayias, editor, *FC 2017*, volume 10322 of *LNCS*, pages 517–529, 2017.
15. Anamaria Costache, Nigel P. Smart, Srinivas Vivek, and Adrian Waller. Fixed-point arithmetic in SHE schemes. In Roberto Avanzi and Howard M. Heys, editors, *SAC 2016*, volume 10532 of *LNCS*, pages 401–422. Springer, Heidelberg, 2016.
16. CryptoExperts. FV-NFLlib. `https://github.com/CryptoExperts/FV-NFLlib`, 2016.
17. Nicolaas Govert De Bruijn. *Asymptotic methods in analysis*. Dover, New York, NY, 1958.
18. Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing. Manual for using homomorphic encryption for bioinformatics. *Proceedings of the IEEE*, 105(3):552–567, 2017.
19. Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 617–640. Springer, Heidelberg, 2015.
20. Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. `http://eprint.iacr.org/2012/144`.
21. Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, 2009.
22. Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 465–482. Springer, Heidelberg, 2012.
23. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, 2013.
24. Zhicong Huang Amir Jalali Hao Chen, Kyoohyung Han and Kim Laine. Simple encrypted arithmetic library — SEAL (v2.3). Technical report, Technical report, Microsoft Research, 2017.
25. Michael Naehrig, Kristin E. Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In Christian Cachin and Thomas Ristenpart, editors, *Proceedings of the 3rd ACM Cloud Computing Security Workshop, CCSW 2011*, pages 113–124. ACM, 2011.
26. Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.
27. Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *Des. Codes Cryptography*, 71(1):57–81, 2014.