Bootstrapping with RMFE for Fully Homomorphic Encryption*

Khin Mi Mi Aung[1][0000−0002−5652−3455], Enhui Lim[2][0000−0003−2702−6537], Jun Jie Sim[0000−0002−3709−6929], Benjamin Hong Meng Tan[1][0000−0002−8629−9052], and Huaxiong Wang[2][0000−0002−7669−8922]

[1] Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore
{mi_mi_aung, benjamin_tan}@i2r.a-star.edu.sg
[2] School of Physical & Mathematical Sciences, Nanyang Technological University, Singapore
{enhui001@e.ntu.edu.sg,hxwang@ntu.edu.sg}

Abstract. There is a heavy preference towards instantiating BGV and BFV homomorphic encryption schemes where the cyclotomic order m is a power of two, as this admits highly efficient fast Fourier transformations. Field Instruction Multiple Data (FIMD) was introduced to increase packing capacity in the case of small primes and improve amortised performance, using reverse multiplication-friendly embeddings (RMFEs) to encode more data into each SIMD slot. However, FIMD currently does not admit bootstrapping.

In this work, we achieve bootstrapping for RMFE-packed ciphertexts with low capacity loss. We first adapt the digit extraction algorithm to work over RMFE-packed ciphertexts, by applying the recode map after every evaluation of the lifting polynomial. This allows us to follow the blueprint of thin bootstrapping, performing digit extraction on a single ciphertext. To achieve the low capacity loss, we introduce *correction maps* to the Halevi-Shoup digit extraction algorithm, to remove all but the final recode of RMFE digit extraction.

We implement several workflows for bootstrapping RMFE-packed ciphertexts in HElib, and benchmark them against thin bootstrapping for $m = 32768$. Our experiments show that the basic strategy of recoding multiple times in digit extraction yield better data packing, but result in very low remaining capacity and latencies of up to hundreds of seconds. On the other hand, using correction maps gives up to 6 additional

* This research is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Trust Tech Funding Initiative (DTC-RGC-01). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority. Huaxiong Wang is supported by Singapore Ministry of Education Academic Research Fund Tier 2 Grant MOE-000623-00. Work was done while Jun Jie Sim was affiliated with Nanyang Technological University and the Institute for Infocomm Research.

multiplicative depth and brings latencies often below 10 seconds, at the cost of lower packing capacity.

Keywords: Fully Homomorphic Encryption · Bootstrapping · Galois Rings · Reverse Multiplication Friendly Embeddings

1 Introduction

Homomorphic encryption (HE) allows computation on ciphertexts, without decrypting to the underlying data. In particular, fully homomorphic encryption (FHE) enables arbitrary computation on encrypted data. Since the first instantiation of FHE by Gentry [17], significant progress has been made to speed up FHE computation but it remains costly compared to working in plaintext.

Currently, four main FHE schemes, Brakerski-Gentry-Vaikunathan (BGV) [6], Brakerski-Fan-Vercauteren (BFV) [5,15], Cheon-Kim-Kim-Song (CKKS) [9], and FHEW/TFHE [10, 11, 13] are commonly used. Of these four, the BGV and BFV schemes support exact arithmetic over integer rings modulo a prime power (more precisely, they enable arithmetic over Galois rings).

In the first three schemes, ciphertexts can only be applied to a finite number of multiplications, after which they must undergo *bootstrapping* for use in further multiplications. Bootstrapping, broadly speaking, refers to homomorphically evaluating the decryption of a ciphertext using an encrypted secret key. This effectively removes the noise that is in the input ciphertext, leaving a result that can be used in several multiplications. The bootstrapping protocol is very costly, and much research attention has been devoted to making it more efficient. Bootstrapping broadly consists of a linear phase and a non-linear modular reduction or rounding step, though the exact strategy used differs among the various FHE schemes. Bootstrapping in the BGV and BFV schemes are nearly identical, using a digit extraction procedure to evaluate the non-linear rounding step [16]. CKKS on the other hand is an approximate homomorphic encryption scheme, and its bootstrapping procedure approximates the modular reduction step of the decryption formula. Most works do this by evaluating a polynomial approximation of the modular reduction operation, usually via a periodic function such as sine or cosine. FHEW/TFHE uses a different computation paradigm but also requires bootstrapping to achieve fully homomorphic encryption. These schemes use a homomorphic accumulator for the linear phase and a variation of sample extraction to achieve the non-linear rounding step, which was extended to functional bootstrapping to simultaneously round and compute a lookup table on the bootstrapping input.

Two fundamental parameters for instantiating the BGV/BFV schemes are the prime modulus p and the order m of the cyclotomic polynomial modulus. To fully exploit the available plaintext space in BGV/BFV, Single Instruction Multiple Data (SIMD), first proposed by Smart-Vercauteren [31], can be used to pack more data in slots that fit within a single ciphertext. Choosing m to be a power of two is heavily favoured in BGV/BFV implementations [22, 24, 26,

30] for efficient HE operations. A side effect of this choice is that using small odd primes for p (e.g. $3, 5, 7, \dots$) yields very few SIMD slots, each of which is an algebra of a very high degree. This results in inefficient packing, rendering popular instantiations of BGV/BFV impractical for small-prime arithmetic.

Addressing this limitation, Aung et al. [2] introduced the technique *Field Instruction Multiple Data* (FIMD) to support small-prime finite field arithmetic. The core idea is to leverage reverse multiplication-friendly embeddings (RMFEs) to embed vectors of base field elements into each SIMD slot. This enables computation on the plaintext space to carry over component-wise to each embedded vector, hence exploiting the high extension degree of SIMD slots caused by small primes to increase ciphertext packing capacity. A significant amortised speedup was achieved, alleviating the low-capacity issue brought on by small primes. However, this technique did not support bootstrapping.

The goal of this present work is to unlock bootstrapping for RMFE-packed ciphertexts. While BGV/BFV can be instantiated with a plaintext modulus p^r where $r \geq 1$, our work focuses on the case where $r = 1$, following the context where FIMD is applied. Bootstrapping with the FIMD setting enables the evaluation of deeper circuits that leverage small-prime arithmetic, such as digit representation-based integer comparisons [23]. It may also be more useful for evaluating quantised neural networks and transformers on encrypted data [25].

1.1 Related work

BGV/BFV Bootstrapping. The two schemes essentially follow the same workflow for bootstrapping [16]. Halevi and Shoup [19] achieved bootstrapping in `HElib` [22] for BGV with plaintext spaces $\mathbb{Z}_{p^r}$, $r \geq 1$, by generalising the procedure of Gentry et al. [18]. They proposed a digit extraction algorithm using a lifting polynomial of degree p. Their bootstrapping procedure takes in a "thickly packed" ciphertext in which each SIMD slot contains an arbitrary element of the slot algebra $E = \mathbb{Z}_{p^r}[X]/(f(X))$ for some irreducible polynomial of d, and requires d-many runs of the digit extraction algorithm.

Chen and Han [8] developed a more efficient bootstrapping framework for "thinly packed" ciphertexts, that is, when data in each SIMD slot is restricted to the subring $\mathbb{Z}_{p^r} \subseteq E$. Their bootstrapping procedure only requires one run of the digit extraction algorithm and can be applied to both the BGV and BFV schemes. EC:CheHan18 bootstrapping is called *thin bootstrapping*, and the earlier framework by Halevi and Shoup is called *thick bootstrapping*. Chen and Han included further optimisations for plaintext modulus p^r where $r \geq 2$, such as a modified digit extraction algorithm using their *lowest digit retain* polynomial.

Field Instruction Multiple Data. Small values for p with the popular choice of powers of 2 for m yield few SIMD slots of very high extension degrees, which are not practical for computation involving small-prime arithmetic. To improve the amortised performance of such instantiations, Aung et al. [2] introduced a second layer of packing, on top of SIMD offered by the Chinese Remainder Theorem. Finite field RMFEs were used to encode vectors of base field elements as

field extension elements lying in each SIMD slot. The operations of addition and multiplication on ciphertexts thus act component-wise on the vectors encoded by RMFE, though a recoding map must be evaluated after some multiplications. This packing method was called Field Instruction Multiple Data (FIMD).

For optimisation, Aung et al. [2] further leverage the high extension degree of the SIMD slot algebra by introducing r-fold RMFEs and a three-stage recoding method. These techniques were implemented with the BGV scheme in `HElib` with significant amortised speedups observed. However, their work was confined to the finite field case, hence FIMD could not be readily applied to bootstrapping where the slot algebra is no longer a finite field, but a Galois ring.

Reverse Multiplication-Friendly Embeddings. RMFEs are an algebraic tool that allows component-wise multiplication and addition of vectors over a ring to be carried over to multiplication and addition over an extension ring. Introduced by Cascudo et al. [7] and concurrently studied by Block et al. [3], RMFEs were originally constructed to support multiplication between two elements over the finite field. Cramer et al. [12] then provided a Hensel-like lifting technique that lifted RMFEs from finite fields to Galois rings.

Aung et al. [2] extended finite field RMFEs to r-fold RMFEs, which supported the multiplication of 2^r-many elements. Most recently, Escudero et al. [14] generalised RMFEs to degree-η RMFEs which support the multiplication of η-many elements over Galois rings, where $\eta \geq 2$ is an integer. They also found that for any finite field RMFE supporting some number of multiplications, there exists a lift to an equivalent RMFE over Galois rings.

1.2 Our contributions

In this work, we enable bootstrapping for RMFE-packed BGV/BFV ciphertexts, where the plaintext modulus is p. The main contributions of this work are:

- We adapt the Halevi-Shoup digit extraction method for RMFE-packed ciphertexts to achieve bootstrapping for RMFE-packed ciphertexts. The digit extraction algorithm as proposed by Halevi and Shoup does not work for general Galois ring elements because the lifting polynomials developed in [19, Corollary 5.4] do not apply to Galois ring extension elements (only integers). We first overcome this by applying RMFE recoding after each evaluation of the lifting polynomial, at the cost of increased noise growth. The bootstrapping process for RMFE-packed ciphertexts differs from traditional thin and thick bootstrapping by taking in a thick RMFE-packed ciphertext as an input (unlike thin bootstrapping) and performing digit extraction on only one ciphertext (unlike thick bootstrapping, which requires d-many runs).
- We propose a less noisy alternative to digit extraction for RMFE-packed ciphertexts by introducing new *correction maps*. These maps remove the need for RMFE recoding after every evaluation of the lifting polynomial, requiring only a single evaluation of the RMFE recode map at the end of the bootstrapping process. Furthermore, our technique leaves the multiplicative degree of digit extraction unchanged. Empirically, this method performs best

for the smallest primes $p = 3, 5$ where there are multiple digit extraction steps and recovers up to 6 additional multiplicative depth.

Our work also includes the following auxiliary contribution.

– We construct high-degree RMFEs directly over Galois rings by naturally extending a construction technique by Cascudo et al. [7, Lemma 4]. This approach is very practical for high values of η and Galois rings of large extension degree. The construction avoids the need to lift a finite field RMFE unlike existing methods in the literature.

2 Preliminaries

Notation. The theory on the BGV/BFV plaintext space and RMFEs over Galois rings are presented in full generality assuming a prime power modulus denoted by p^r. The parameter m denotes the cyclotomic order of the plaintext ring modulus, and ℓ denotes the number of SIMD slots in a ciphertext. A general Galois ring is denoted as $GR(p^r, s)$, where p^r is the characteristic of the ring, and $s \geq 1$ is the extension degree of the ring with respect to $\mathbb{Z}_{p^r}$. In bootstrapping, the plaintext is temporarily raised from the regular modulus p^r to a higher modulus p^e. Later in our work, we assume the regular plaintext modulus is p in line with the FIMD setting. When combining RMFEs with bootstrapping, we consider RMFEs with modulus p^e. Modular reduction by p is denoted by $[\cdot]_p$. If z is an integer, $z\langle i \rangle$ denotes the i-th p-ary digit of z. In this notation, $[z]_p = z\langle 0 \rangle$ is the least significant digit of z.

2.1 Fully Homomorphic Encryption.

A fully homomorphic encryption (FHE) scheme is an encryption scheme that supports evaluating arbitrary functions on encrypted data. Constructions of FHE schemes are mostly based on the Ring Learning with Errors (RLWE) problem with ciphertexts being noisy encryptions of the message. As computation is performed on ciphertexts, the noise within will grow and eventually flood the message, rendering the ciphertext useless. Bootstrapping [17] is a procedure that refreshes the ciphertexts and reduces the noise in it, such that there is room again for noise to grow in subsequent computations.

Let a ciphertext of a message m is denoted as $\overline{m}$ and $\mathcal{P}$ denote the message space of the FHE scheme. An FHE scheme is defined as follows:

– $(\mathsf{pk}, \mathsf{sk}, \mathsf{evk}, \mathsf{bk}) \leftarrow \mathtt{KeyGen}(1^\lambda)$: Given security parameter λ, return a public key pk, secret key sk, evaluation key evk, and bootstrapping key bk.
– $c \leftarrow \mathtt{Encrypt}(m, \mathsf{pk})$: Given a message $m \in \mathcal{P}$ and public key pk, return ciphertext $c = \overline{m}$.
– $m' \leftarrow \mathtt{Decrypt}(c, \mathsf{sk})$: Given a ciphertext $c = \overline{m'}$ and secret key sk, return message $m' \in \mathcal{P}$.
– $c^+ \leftarrow \mathtt{EvalAdd}(c_1, c_2, \mathsf{evk})$: Given two ciphertexts $c_1 = \overline{m_1}$, $c_2 = \overline{m_2}$, and evaluation key evk, return a ciphertext $c^+ = \overline{m_1 + m_2}$.

- $c^{\times} \leftarrow \texttt{EvalMult}(c_1, c_2, \mathsf{evk})$: Given two ciphertexts $c_1 = \overline{m_1}$, $c_2 = \overline{m_2}$, and evaluation key evk, return a ciphertext $c^{\times} = \overline{m_1 \times m_2}$.
- $c' \leftarrow \texttt{Bootstrap}(c, \mathsf{bk})$: Given a ciphertext $c = \overline{m}$ and bootstrapping key bk, return a ciphertext $c' = \overline{m}$.

Single Instruction Multiple Data. Let $\Phi_m(X)$ to be the m-th cyclotomic polynomial and define $R = \mathbb{Z}[X]/\Phi_m(X)$ and $R_q = R/qR$. In second generation FHE schemes [6, 15], the plaintext space is defined as $\mathcal{P} = R_{p^r}$, for some prime power p^r. An observation by Smart and Vercauteren [31] states that for coprime m and p^r, the cyclotomic polynomial can be split into ℓ irreducible polynomials by factoring modulo p and applying Hensel lifting. Each factor $F_i(X)$ has degree $d = \phi(m)/\ell$. Applying the Chinese Remainder Theorem to the plaintext space ring R_{p^r} yields the following

$$\mathcal{P} = R_{p^r} = \mathbb{Z}_{p^r}[X]\big/\Phi_m(X) = \prod_{i=1}^{\ell} \mathbb{Z}_{p^r}[X]\big/F_i(X).$$

Let ζ be a residue class X in $\mathbb{Z}_{p^r}[X]/F_i(X)$ and define $E = \mathbb{Z}_{p^r}[\zeta]$. This gives rise to the isomorphism $R_{p^r} \to E^{\ell}$, $\alpha(X) \mapsto \{\alpha(\zeta^h)\}_{h \in S}$ where $S \subset \mathbb{Z}$ is a set of representatives for $\mathbb{Z}_m^*/\langle p \rangle$. By leveraging on this isomorphism, a SIMD ciphertext can encrypt a vector $(m_1, \ldots, m_\ell) \in E^{\ell}$. Ciphertext addition and multiplication translate to component-wise ring addition and multiplication. Elements of the encrypted vector can be shuffled by applying ring automorphisms $\theta_\kappa : X \mapsto X^{\kappa}$ for $\kappa \in S$, where $\theta_\kappa(\alpha(X)) \mapsto \{\alpha(\zeta^{\kappa h})\}_{h \in S}$. A detailed exposition of the plaintext structure and effect of ring automorphisms can be found in [20, 21]. By using suitable masking vectors and different automorphisms, it is possible to achieve the effect of shifting or rotating the slots.

Frobenius Maps and Linear Maps Evaluation. The automorphism map $\theta_p : X \mapsto X^p$ is the Frobenius map on R_{p^r}. Based on the following relation, the Frobenius map on R_{p^r} translates to slotwise Frobenius maps on E, fixing elements of the subring $\mathbb{Z}_{p^r}$

$$\theta_p(\alpha(X)) \mapsto \{\alpha(\zeta^{ph})\}_{h \in S} = \{\theta_p(\alpha(\zeta^h))\}_{h \in S}.$$

Frobenius maps are useful when evaluating $\mathbb{Z}_p$ linear transformations. Any $\mathbb{Z}_p$-linear transformation M, on E can be written as a unique linearized polynomial $f_M(X) = \sum_{i=1}^{d} c_i X^{p^i}$, with $c_i \in E$ [29]. Equivalently, $f_M(\alpha) = M(\alpha)$ for $\alpha \in E$. The evaluation of the linearized polynomial on α is given as $f(\alpha) = \sum_{i=1}^{d} c_i \theta_p^i(\alpha)$.

2.2 Galois rings and the plaintext slot algebra

In this subsection, we review the theory of general Galois rings, after which we treat the structure of the plaintext slot algebra.

Let $f(X) \in \mathbb{Z}_{p^r}[X]$ be a degree-s monic irreducible polynomial that is also irreducible modulo p. The ring $GR(p^r, s) := \mathbb{Z}_{p^r}[X]/(f(X))$ is called a Galois ring. The value p^r is the characteristic of the ring, and the parameter s is called the degree of the ring. The cardinality of $GR(p^r, s)$ is p^{rs}. In the case where $r = 1$, we recover the construction of the finite field $\mathbb{F}_{p^s}$. When $s = 1$, we get $\mathbb{Z}_{p^r}$. Two Galois rings are isomorphic if they have the same characteristic and cardinality [32, Corollary 14.7], thus we may speak of *the* Galois ring $GR(p^r, s)$.

Letting ζ denote a root of $f(X)$, we may represent $GR(p^r, s)$ as $\mathbb{Z}_{p^r}[\zeta]$. The irreducibility of $f(X)$ then implies $GR(p^r, s)$ is a free $\mathbb{Z}_{p^r}$-module of rank s with basis $\{1, \zeta, \ldots, \zeta^{s-1}\}$.

Given an element $\beta \in GR(p^r, s)$ where $r \geq 2$, we may define $\beta \bmod p^{r'}$ for $1 \leq r' \leq r$ by sending β to its residue class in $GR(p^r, s)/(p^{r'}) \simeq GR(p^{r'}, s)$. Moreover, if $x = py$ for some $y \in GR(p^r, s)$, we define division by p by mapping x to $y \bmod p^{r-1}$. This division is well-defined, since for any y, y' such that $py = py' = x$, we have that $y = y' + p^{r-1}z$ for some $z \in GR(p^r, s)$, so division of x by p yields $y \bmod p^{r-1}$ which is equal to $y' \bmod p^{r-1}$ in $GR(p^{r-1}, s)$.

Tower structure. The following propositions describe the relationship between a Galois ring and its subrings. We will exploit this structure later on in our work.

Proposition 1. *[28, Proposition 1] Every subring of $GR(p^r, s)$ is of the form $GR(p^r, s')$ for some divisor s' of s. Conversely, for every positive divisor s' of s, there is a unique subring of $GR(p^r, s)$ that is isomorphic to $GR(p^r, s')$.*

Proposition 2. *Let $\mathcal{G} = GR(p^r, s)$ be contained in $\mathcal{G}' = GR(p^r, sn)$. Then $\mathcal{G}'$ is a free $\mathcal{G}$-module of rank n.*

Proof. By [32, Theorem 14.27(i)], $\mathcal{G}' = \mathcal{G}[\xi]$ for some $\xi \in \mathcal{G}'$ such that ξ is a root of a degree-n polynomial $h(X)$ that is irreducible in $\mathcal{G}[X]$ modulo p. Furthermore, $h(X)$ is the unique polynomial of degree not more than n that has ξ as a root. By the degree of $h(X)$, ξ^n can be rewritten as a $\mathcal{G}$-linear combination of lower powers of ξ. Thus the set $B = \{1, \xi, \ldots, \xi^{n-1}\}$ is a generating set of $\mathcal{G}'$ over $\mathcal{G}$.

We now claim that B is linearly independent over $\mathcal{G}$. Suppose not. Then $\sum_{i=0}^{n-1} a_i \xi^i = 0$ for some $a_0, \ldots, a_{n-1} \in \mathcal{G}$, and so $g(X) = \sum_{i=0}^{n-1} a_i X$ is a polynomial over $\mathcal{G}$ of degree less than n having ξ as a root. However, this contradicts the minimal degree property of $h(X)$. Hence B must be $\mathcal{G}$-linearly independent, completing the proof. □

Linear Maps over Galois Rings. In BGV/BFV, the plaintext slot algebra E is $GR(p^r, d)$, which we denote as $\mathbb{Z}_{p^r}[\zeta]$, where ζ is a root of an irreducible factor of $\Phi_m(X)$. Hence, ζ is an m-th root of unity. By Proposition 1, E contains a unique subring isomorphic to $GR(p^r, s)$ for each divisor $s > 0$ of d.

The Frobenius map $\sigma : \zeta \mapsto \zeta^p$ of E is an automorphism on E that fixes $\mathbb{Z}_{p^r}$ only, per [19]. More generally, for every positive divisor s of d, the map σ^s is an automorphism on E fixing the subring $GR(p^r, s) \subseteq E$ by [32, Corollary 14.33]. When $r = 1$, σ is the usual Frobenius automorphism of $\mathbb{F}_{p^d}$ defined as $\sigma : a \mapsto a^p$ for all $a \in \mathbb{F}_{p^d}$. We record the following results which are crucial to leveraging linear maps over Galois rings in homomorphic encryption.

Proposition 3. *Let $\mathcal{G} = GR(p^r, s)$ and let $\mathcal{G}'$ be an extension $\mathcal{G}$. Let $f : V \to \mathcal{G}'$ be a $\mathcal{G}$-linear map where V is a free $\mathcal{G}$-submodule of $\mathcal{G}'$ with rank k. Then there exist unique elements $\theta_0, \ldots, \theta_{k-1} \in \mathcal{G}'$ such that for all $x \in V$,*

$$f(x) = \sum_{i=0}^{k-1} \theta_i \sigma^{si}(x)$$

where σ is the Frobenius map of $\mathcal{G}'$.

Proof. Let $\{u_0, \ldots, u_{k-1}\}$ be a $\mathcal{G}$-basis of V. Consider the Moore matrix

$$\mathcal{M} = \begin{bmatrix} u_0 & \cdots & u_{k-1} \\ \sigma^s(u_0) & & \sigma^s(u_{k-1}) \\ \vdots & & \vdots \\ \sigma^{s(k-1)}(u_0) & \cdots & \sigma^{s(k-1)}(u_{k-1}). \end{bmatrix}$$

We want to find $\theta_0, \ldots, \theta_{k-1}$ satisfying

$$\begin{bmatrix} \theta_0 \cdots \theta_{k-1} \end{bmatrix} \mathcal{M} = \begin{bmatrix} f(u_0) \cdots f(u_{k-1}) \end{bmatrix},$$

which we may obtain by computing

$$\begin{bmatrix} \theta_0 \cdots \theta_{k-1} \end{bmatrix} = \begin{bmatrix} f(u_0) \cdots f(u_{k-1}) \end{bmatrix} \mathcal{M}^{-1}.$$

The uniqueness of $\theta_0, \ldots, \theta_{k-1}$ follows. □

Proposition 4. *Let $\mathcal{G}'$ be a degree-n extension of $\mathcal{G} = GR(p^r, s)$ and let $f : \mathcal{G}' \to \mathcal{G}$ be a $\mathcal{G}$-linear map. Then there exists $\theta \in \mathcal{G}'$ such that*

$$f(x) = \sum_{i=0}^{n-1} \sigma^{si}(\theta x).$$

Proof. By Proposition 3, there exist unique $\theta_0, \ldots, \theta_{n-1}$ such that for all $x \in \mathcal{G}'$, $f(x) = \sum_{i=0}^{n-1} \sigma^s(x)$. Since $\operatorname{im} f \subseteq \mathcal{G}$, $f(x)$ is fixed by σ^s. And so

$$\begin{aligned} f(x) &= \sigma^s(f(x)) \\ &= \sum_{i=0}^{n-1} \sigma^s(\theta_i)\sigma^{s(i+1)}(x) \\ &= \sigma^s(\theta_{n-1})x + \sigma^s(\theta_0)\sigma^s(x) + \sigma^s(\theta_1)\sigma^{2s}(x) + \cdots + \sigma^s(\theta_{n-2})\sigma^{(n-1)s}(x). \end{aligned}$$

The uniqueness of $\theta_0, \ldots, \theta_{n-1}$ then implies $\theta_i = \sigma^{si}(\theta_0)$ for $1 \leq i \leq n-1$. Thus we may write $f(x) = \sum_{i=0}^{n-1} \sigma^{si}(\theta_0 x)$, completing the proof. □

3 BGV Bootstrapping

Bootstrapping involves "refreshing" a noisy ciphertext into one with less noise encrypting the same message. First proposed by Gentry in his seminal work [17], the key idea is to evaluate the decryption circuit homomorphically. The inputs to this process are the noisy ciphertext and a bootstrapping key, which is an encryption of the secret key under itself (assuming circular security). It is helpful to regard the bootstrapping key as a ciphertext input and ciphertext to be "refreshed" as a plaintext input to the homomorphic decryption circuit

$$\texttt{Bootstrap}\big(\overline{m}, \overline{\mathsf{sk}}\big) = \overline{\texttt{Decrypt}(\overline{m}, \mathsf{sk})} = \overline{m}.$$

Now, we recall the process of bootstrapping for the BGV [6] scheme. Modulo reduction by p is denoted by $[\cdot]_p$ and $z\langle i\rangle$ denotes the i-th digit of $z \in \mathbb{Z}$.

3.1 An Alternative Decryption Formula

Decrypting a BGV ciphertext consists of two steps; an inner product followed by a modulo reduction. For a ciphertext $c = (c_0, c_1)$, encrypted under secret key $\mathsf{sk} = (1, \mathfrak{s})$,

$$\texttt{Decrypt}(c, \mathsf{sk}) = [[c_0 + c_1\mathfrak{s}]_q]_p = [m + pe]_p,$$

for some $e < \frac{q}{p}$. Modulo reduction removes the noise (pe) that is crucial to the security of the BGV scheme. As BGV supports linear functions, the inner product $c_0 + c_1\mathfrak{s}$ can be computed easily. The non-linear modulo reduction turns out to be the bottleneck of bootstrapping.

It was observed in [18] that modulo reduction $[[\cdot]_{\tilde{q}}]_2$ for $\tilde{q} = 2^e + 1$ can be achieved by computing the sum of certain bits. More precisely, for $u = c_0 + c_1\mathfrak{s}$, we have $[[u]_{\tilde{q}}]_2 = u\langle e\rangle \oplus u\langle e-1\rangle \oplus u\langle 0\rangle$. This was further generalized by [19, Lemma 5.1] to p^r into

$$[[u]_{\tilde{q}}]_{p^r} = u\langle r-1, \cdots, 0\rangle - u\langle e+r-1, \cdots, e\rangle.$$

The cost of bootstrapping increases as e grows because the algorithm to homomorphically extract digits requires the evaluation of polynomials up to degree 2^e. An optimization by [18] reduced the depth of digit extraction. Let $e' < e$ and $\tilde{c}$ denote the ciphertext encrypting the same message over $\tilde{q}$, i.e. $[[\langle \tilde{c}, \mathsf{sk}\rangle]_{\tilde{q}}]_2 = [[\langle c, \mathsf{sk}\rangle]_q]_2$. Add multiples of $\tilde{q}$ to coefficients of $\tilde{c}$ such that each coefficient is divisible by $2^{e'}$. Denote this modified ciphertext as $\tilde{c}' = \big(\tilde{c_0}', \tilde{c_1}'\big)$. It is easy to see that $\tilde{c}'$ still encrypts the same message relative to $\tilde{q}$. Since $\tilde{c}'$ divides $2^{e'}$, the inner product $\tilde{u}' = \tilde{c_0}' + \tilde{c_1}'\mathfrak{s}$ also divides $2^{e'}$ and $\tilde{u}'\langle 0\rangle = 0$. In fact, $\tilde{u}'\langle e', \cdots, 0\rangle = 0$. This means that dividing $\tilde{c}'$ by $2^{e'}$ would yield a smaller result with fewer bits to extract

$$[[\tilde{u}'/2^{e'}]_{\tilde{q}/2^{e'}}]_2 = \tilde{u}'\langle e - e'\rangle \oplus \tilde{u}'\langle e - e' - 1\rangle.$$

The generalization for p^r was proposed in [19].

3.2 Homomorphic Digit Extraction

Digit extraction is the most expensive step in the bootstrapping procedure. Given an integer $z \in \mathbb{Z}_{p^e}$, the aim of digit extraction is to homomorphically remove some lower p-ary digits. For our setting, we are interested in removing the $e-1$ lowest digits and focus on the digit extraction procedure by Halevi and Shoup [19], which we give in Algorithm 1.

Halevi-Shoup digit extraction requires a polynomial $F_e(X) \in \mathbb{Z}[X]$ called the *lifting polynomial* introduced in [19, Corollary 5.5], such that for $1 \leq e' \leq e$, if $[z]_{p^{e'}} = z\langle 0 \rangle$ then $[F_e(z)]_{p^{e'+1}} = z\langle 0 \rangle$. The lifting polynomial has degree p and is of the form $X^p + pG(X)$ for some polynomial $G(X)$.

Algorithm 1: Halevi-Shoup Digit Extraction

Input: $z \in \mathbb{Z}_{p^e}$

$u_{0,0} \longleftarrow z$;
for $j = 0, \ldots, e-2$ **do**
  $y \leftarrow z$;
  for $i = 0, \ldots, j$ **do**
    $u_{i,j+1} \leftarrow F_e(u_{i,j})$;
    $y \leftarrow (y - u_{i,j+1})/p$;
  end
  $u_{j+1,j+1} \leftarrow y$;
end
return $u_{e-1,e-1}$;

Figure 1 illustrates how the digit extraction algorithm works for a toy example, for $e = 3$. For ease of exposition, denote the digits of z as $[z_2 z_1 z_0]$. $*$ is used as a placeholder to represent any digit. Each horizontal arrow represents each time the lifting polynomial is evaluated. The input starts with the top left corner $z = u_{0,0}$. Finally, the least significant digit of $u_{i,0}$ is the i-th bits of z.

$$u_{0,0} = z = [z_2 z_1 z_0] \longrightarrow u_{0,1} = F_e(u_{0,0}) = [*0 z_0] \longrightarrow \quad u_{0,2} = F_e(u_{0,1}) = [00 z_0]$$

$$u_{1,1} = \tfrac{u_{0,0} - u_{0,1}}{p} = [*z_1] \longrightarrow \quad u_{1,2} = F_e(u_{1,1}) = [0 z_1]$$

$$u_{2,2} = \frac{\frac{u_{0,0} - u_{0,2}}{p} - u_{1,2}}{p} = [z_2]$$

Fig. 1. Toy Example for Homomorphic Digit Extraction with $z = [z_2 z_1 z_0]$.

As the digit extraction procedure is repeated for each polynomial coefficient, [18] proposed to pack them into ciphertexts and extract the digits in batches

to improve efficiency. Two linear maps, `CoeffToSlots` and `SlotsToCoeff`, were proposed to transform the polynomial coefficients into slots of a ciphertext and back. We refer readers to [19] for more details on the linear maps.

3.3 Bootstrapping Procedure

General (thick) bootstrapping. The complete bootstrapping algorithm is presented as follows:

1. `ModSwitch`. The modulus of the ciphertext is raised to p^e.
2. `InnerProduct`. The ciphertext is homomorphically multiplied with the bootstrapping key.
3. `CoeffToSlots`. The coefficients of $c_0+c_1\mathfrak{s}$ are mapped into slots of d-many ciphertexts.
4. `DigitExtract`. Run digit extraction on all d packed ciphertexts.
5. `SlotsToCoeff`. Transform the polynomial coefficients within slots back into polynomial coefficients.

Thin bootstrapping. An optimization by [8] reduces the number of digit extractions done under certain conditions. Recall that the slots of the ciphertext are elements of a finite extension field E. [8] observed that when the message is packed into the constant term of each slot, applying `SlotsToCoeff` transforms the plaintext polynomial with $\phi(m)$ many coefficients into a sparse polynomial of ℓ coefficients. More precisely,

$$[m_1, \cdots, m_\ell] \xrightarrow{\texttt{SlotsToCoeff}} m_1 + m_2 x^d + \cdots + m_\ell x^{d\ell}.$$

With such a transformation, one ciphertext is sufficient to pack all coefficients of the resultant polynomial. Consequentially, one digit extraction needs to be performed. The complete thin bootstrapping algorithm is presented as follows:

1. `ModSwitch`. The modulus of the ciphertext is raised to p^e.
2. `SlotsToCoeff`. Transform the constant terms of the ciphertext slots into an equivalent polynomial whose coefficients are the slots elements.
3. `InnerProduct`. The ciphertext is homomorphically multiplied with the bootstrapping key.
4. `CoeffToSlots`. Send the coefficients of $c_0 + c_1\mathfrak{s}$ into slots of a ciphertext.
5. `DigitExtract`. Run digit extraction on the packed ciphertext.

4 Practical RMFEs for bootstrapping

In this section, we review the theory of reverse multiplication-friendly embeddings (RMFEs), and then detail our approach for constructing RMFEs directly over Galois rings, which generalises the construction in [7, Lemma 4].

4.1 Background and terminology

In this subsection, we give a technical overview of RMFEs and present results relevant to our work. We begin with the following general definition from the recent study on RMFEs by Escudero et al. [14]. For lighter notation, we simply write a Galois ring $GR(p^r, s)$ as $\mathcal{G}$ and its degree-w extension $GR(p^r, sw)$ as $\mathcal{G}'$.

Definition 1. *[14, Definition 3] A pair (ϕ, ψ) of additive maps, with $\phi : \mathcal{G}^k \to \mathcal{G}'$ and $\psi : \mathcal{G}' \to \mathcal{G}^k$, is called a $(k, w; \eta)$ RMFE over $\mathcal{G}$ if for any $\mathbf{v}_1, \ldots, \mathbf{v}_\eta \in \mathcal{G}^k$ and denoting component-wise multiplication with $*$, we have*

$$\psi\left(\phi(\mathbf{v}_1) \cdots \phi(\mathbf{v}_\eta)\right) = \mathbf{v}_1 * \cdots * \mathbf{v}_\eta.$$

We refer to the parameter w as the *dimension* of the RMFE. Following the convention of [14], the parameter η is called the *degree* of an RMFE. In this work, the maps ϕ and ψ are referred to as the encode and decode map respectively. Another relevant map is the recode map $\pi = \phi \circ \psi$, which decodes an RMFE-packed element and re-encodes its data into an element that can be used for subsequent multiplications.

Proposition 5. *[14, Proposition 2 and Lemma 1] Let (ϕ, ψ) be an RMFE. Then ϕ and ψ are $\mathbb{Z}_{p^r}$-linear maps. Additionally, ϕ is injective and ψ is surjective.*

For our work, we are primarily concerned with RMFEs over $\mathcal{G}$ where ϕ and ψ are $\mathcal{G}$-linear, as opposed to just being $\mathbb{Z}_{p^r}$-linear. $\mathcal{G}$-linearity is not implied by Definition 1, thus we use the term $\mathcal{G}$-RMFE to distinguish RMFEs over $\mathcal{G}$ with the added property of being $\mathcal{G}$-linear. Naturally, any $\mathcal{G}$-RMFE is also a $\mathbb{Z}_{p^r}$-RMFE. We highlight the following useful technique from [14] that introduces a slight modification to an RMFE.

Proposition 6. *[14, Lemma 2] Let (ϕ, ψ) be a $(k, w; \eta)$-RMFE over a Galois ring $\mathcal{G}$, and let $\mathbf{1}$ denote the all-ones vector in $\mathcal{G}^k$. There exists a (k, w, η) RMFE (ϕ', ψ') such that $\phi'(\mathbf{1}) = 1$.*

Proof. We claim that $\phi(\mathbf{1})$ is a unit in the extension ring $\mathcal{G}'$ of $\mathcal{G}$. Suppose not. Then $\phi(\mathbf{1})$ is a multiple of p, and so $\phi(p^{r-1}\mathbf{1}) = p^{r-1}\phi(\mathbf{1}) = 0$. This implies $\ker \phi$ is nontrivial, contradicting Proposition 5. Thus the claim is true. We now proceed with the construction of (ϕ', ψ').

Define $\phi' : \mathcal{G}^k \to \mathcal{G}'$ and $\psi' : \mathcal{G}' \to \mathcal{G}^k$ such that

$$\phi' : v \mapsto \phi(\mathbf{v}) \cdot \phi(\mathbf{1})^{-1},$$
$$\psi' : \alpha \mapsto \psi(\alpha \cdot \phi(\mathbf{1})^\eta).$$

It is easy to see that $\phi'(\mathbf{1}) = \phi(\mathbf{1}) \cdot \phi(\mathbf{1})^{-1} = 1$. We show now that (ϕ', ψ') is a valid degree-η RMFE. It is easily checked that ϕ' and ψ' are linear over $\mathbb{Z}_{p^r}$. Finally, for any $\mathbf{v}_1, \ldots, \mathbf{v}_\eta \in \mathcal{G}^k$ we have

$$\psi'(\phi'(\mathbf{v}_1) \cdots \phi'(\mathbf{v}_\eta)) = \psi(\phi(\mathbf{v}_1) \cdots \phi(\mathbf{v}_\eta) \cdot \phi(\mathbf{1})^{-\eta} \cdot \phi(\mathbf{1})^\eta) = \mathbf{v}_1 * \cdots * \mathbf{v}_\eta. \quad \square$$

Escudero et al. [14] note that a degree-η RMFE is also a degree-η' RMFE where $\eta' \leq \eta$ if $\phi(\mathbf{1}) = 1$ (the converse is not necessarily true). We further observe that such RMFEs allow us to carry out polynomial evaluations. More specifically, for any $\mathbf{v} = (x_1, \ldots, x_k) \in \mathcal{G}^k$ and polynomial $f \in \mathbb{Z}_{p^r}[X]$ of degree at most η, the relation $\psi(f(\phi(\mathbf{v}))) = (f(x_1), \ldots, f(x_k))$ holds. We call this property *evaluation-friendliness*. For completeness, we record this observation in the following proposition with proof.

Proposition 7. *Let (ϕ, ψ) be a degree-η RMFE over $\mathcal{G}$ and let $\mathbf{1}$ denote the all-ones vector in $\mathcal{G}^k$. If $\phi(\mathbf{1}) = 1$, then (ϕ, ψ) is evaluation-friendly.*

Proof. First observe that for any integer $1 \leq \eta' \leq \eta$ and any η' elements $\mathbf{v}_1, \ldots, \mathbf{v}_{\eta'} \in \mathcal{G}^k$, we have that

$$\psi(\phi(\mathbf{v}_1) \cdots \phi(\mathbf{v}_{\eta'})) = \psi(\underbrace{\phi(\mathbf{v}_1) \cdots \phi(\mathbf{v}_{\eta'}) \cdot \phi(\mathbf{1})^{\eta - \eta'}}_{\eta\text{-many encodings}}) = \mathbf{v}_1 * \cdots * \mathbf{v}_{\eta'}.$$

Let $f = \sum_{i=0}^{\eta} f_i X^i$ be a polynomial over $\mathbb{Z}_{p^r}$. Then by the $\mathbb{Z}_{p^r}$-linearity of ϕ and ψ, we have that

$$\psi(f(\phi(\mathbf{v}))) = \sum_{i=0}^{\eta} f_i \psi(\phi(\mathbf{v})^i) = \sum_{i=0}^{\eta} f_i \mathbf{v}^i = f(\mathbf{v}),$$

where $\mathbf{v}^i$ denotes the vector obtained by raising the components of $\mathbf{v}$ to the i-th power, and $f(\mathbf{v})$ denotes the evaluation of f component-wise on $\mathbf{v}$. □

Since the technique in Proposition 6 works on arbitrary RMFEs, we can assume that from now on that $\phi(\mathbf{1}) = 1$ without loss of generality and that any RMFE we construct is evaluation-friendly.

We conclude this subsection by proving that an RMFE can be reduced mod $p^{r'}$, where $r' \leq r$. This result is relevant to our work later and a proof was not found in the literature, hence we include it here for completeness.

Proposition 8. *Let $\mathcal{G} = GR(p^r, s)$ and $\mathcal{G}' = GR(p^r, sw)$, and let (ϕ, ψ) be a $(k, w; \eta)$ $GR(p^r, s)$-RMFE. For r' a positive integer such that $r' \leq r$, let the bar-line $\overline{\ }$ denote reduction by modulo $p^{r'}$. Consider the Galois rings $\overline{\mathcal{G}} = \mathcal{G}/(p^{r'}) = GR(p^{r'}, s)$ and $\overline{\mathcal{G}'} = \mathcal{G}'/(p^{r'}) = GR(p^{r'}, sw)$.*

We define the mod $p^{r'}$-reduction $(\overline{\phi}, \overline{\psi})$ of (ϕ, ψ) as follows. For each $\mathbf{v} \in \overline{\mathcal{G}}^k$ there is some $\mathbf{w} \in \mathcal{G}^k$ such that $\mathbf{v} = \overline{\mathbf{w}}$, and for each $b \in \overline{\mathcal{G}'}$ there is some $\beta \in \mathcal{G}'$ such that $b = \overline{\beta}$. Thus define

$$\overline{\phi} : \overline{\mathcal{G}}^k \to \overline{\mathcal{G}'} : \mathbf{v} \mapsto \overline{\phi(\mathbf{w})},$$
$$\overline{\psi} : \overline{\mathcal{G}'} \to \overline{\mathcal{G}}^k : b \mapsto \overline{\psi(\beta)}.$$

The following is true:

1. *$\overline{\phi}$ and $\overline{\psi}$ are well-defined.*

2. $(\overline{\phi}, \overline{\psi})$ *is a* $(k, w; \eta)$ $\overline{\mathcal{G}}$*-RMFE.*
3. *If* (ϕ, ψ) *is evaluation-friendly, then* $(\overline{\phi}, \overline{\psi})$ *is also evaluation-friendly.*

Proof. For (i), we first show that $\overline{\phi}$ is well-defined. Let $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{G}^k$, and denote $\mathbf{w}_i \bmod p^{r'}$ as $\overline{\mathbf{w}_i}$ for $i = 1, 2$. Suppose now that $\overline{\mathbf{w}_1} = \overline{\mathbf{w}_2}$, which implies $\mathbf{w}_1 - \mathbf{w}_2 = p^{r'}\mathbf{v}$ for some $\mathbf{v} \in \mathcal{G}^k$. Then

$$\overline{\phi}(\overline{\mathbf{w}_1}) - \overline{\phi}(\overline{\mathbf{w}_2}) = \overline{\phi(\mathbf{w}_1 - \mathbf{w}_2)} = \overline{p^{r'}\phi(\mathbf{v})} = 0,$$

implying $\overline{\phi}(\overline{\mathbf{w}_1}) = \overline{\phi}(\overline{\mathbf{w}_2})$ as required. The proof that $\overline{\psi}$ is well-defined follows a similar argument.

For (ii), let $\mathbf{w}_i \in \mathcal{G}^k$ such that $\overline{\mathbf{w}_i} = \mathbf{v}_i$ for $1 \leq i \leq \eta$. We have that

$$\overline{\psi}\left(\prod_{i=1}^{\eta} \overline{\phi}(\mathbf{v}_i)\right) = \overline{\psi\left(\prod_{i=1}^{\eta} \phi(\mathbf{w}_i)\right)} = \overline{\mathbf{w}_1 * \cdots * \mathbf{w}_k} = \mathbf{v}_1 * \cdots * \mathbf{v}_k.$$

It is easily checked that ϕ and ψ are $\overline{\mathcal{G}}$-linear. Hence $(\overline{\phi}, \overline{\psi})$ is a $\overline{\mathcal{G}}$-RMFE.

For (iii), evaluation-friendliness is easy to see by substituting η in the above equation with $\eta' \leq \eta$. □

For lighter notation, we will use the same notation ϕ and ψ to also denote their reductions when the modulus is clear from the context.

4.2 Construction

In this subsection, we treat the construction of degree-η RMFEs over Galois rings. Previously, Cramer et al. [12] gave an explicit method to lift degree-2 RMFEs from finite fields to Galois rings. Escudero et al. then explain in [14, Lemma 6] that lifting degree-η RMFEs from finite fields to Galois rings is also possible, though no concrete method is given.

Our approach is to take the "interpolation" construction by Cascudo et al. in [7, Lemma 4] for degree-2 RMFEs over finite fields and explicitly generalise it to construct degree-η $\mathcal{G}$-RMFEs. Thus we obtain RMFEs directly over Galois rings without lifting from the finite field, in contrast to the previous methods in the literature. Constructing RMFEs in this manner is much more practical for our parameters which are often large, and we find this approach is ultimately sufficient for our purposes.

Theorem 1. *Let* $\mathcal{G} = GR(p^r, s)$ *be a Galois ring and* k*,* w*,* η *be positive integers. If* $1 \leq k \leq p^s + 1$ *and* $w \geq \eta(k-1) + 1$*, there exists a* $(k, w; \eta)$ $\mathcal{G}$*-RMFE.*

Proof. For any integer u, let $\mathcal{G}[X]_{\leq u}$ denote the set of polynomials over $\mathcal{G}$ with degree at most u and let ∞_{u+1} be a formal symbol such that $f(\infty_{u+1})$ is the coefficient of X^u in $f \in \mathcal{G}[X]_{\leq u}$. Let S denote a set of representatives of all cosets of the ideal $(p) \subset \mathcal{G}$, and pick k pairwise distinct elements $x_1, \ldots, x_k$ from $S \cup \{\infty_k\}$. We have that $|S| = p^s$, so one can pick at most $p^s + 1$ elements.

Now let $\mathcal{G}'$ denote the degree-w extension of $\mathcal{G}$, with generator ξ. Consider the $\mathcal{G}$-linear maps

$$\mathcal{E}_1 : \mathcal{G}[X]_{\leq k-1} \to \mathcal{G}^k : f \mapsto (f(x_1), \ldots, f(x_k))$$
$$\mathcal{E}_2 : \mathcal{G}[X]_{\leq w-1} \to \mathcal{G}' : f \mapsto f(\xi).$$

Before constructing the RMFE, we want to show that $\mathcal{E}_1$ and $\mathcal{E}_2$ are $\mathcal{G}$-module isomorphisms. It is clear $\mathcal{E}_2$ is an isomorphism. To show $\mathcal{E}_1$ is an isomorphism too, we show that $\mathcal{E}_1$ is injective since $|\mathcal{G}[X]_{\leq k-1}| = |\mathcal{G}^k|$. There are two cases.

In the first case, assume $\infty_k \notin \{x_1, \ldots, x_k\}$. Then $\{x_1, \ldots, x_k\}$ forms a commutative subtractive subset of $\mathcal{G}$. By [27, Corollary 9], the zero element $\mathbf{0} \in \mathcal{G}^k$ corresponds uniquely to the zero polynomial $0 \in \mathcal{G}[X]_{\leq k-1}$. Hence $\ker \mathcal{E}_1$ is trivial and so $\mathcal{E}_1$ is injective.

In the second case, assume $x_k = \infty_k$ without loss of generality. Let $f \in \mathcal{G}[X]_{\leq k-1}$ such that $f(x_i) = 0$ for all $1 \leq i \leq k$. In particular, $f(\infty_k) = 0$ and thus f has degree at most $k-2$. By [27, Corollary 9], the remaining $k-1$ points force f to be the zero function. Hence, $\ker \mathcal{E}_1$ is trivial and so $\mathcal{E}_1$ is injective.

Now let $x'_1, \ldots, x'_k$ form a new set of points, where $x'_i = x_i$ if $x_i \in S$ and $x'_i = \infty_{\eta(k-1)+1}$ otherwise. Define the map

$$\mathcal{E}'_1 : \mathcal{G}[X]_{\leq w-1} \to \mathcal{G}^k : f \mapsto (f(x'_1), \ldots, f(x'_k)).$$

Now we construct $\phi = \mathcal{E}_2 \circ \mathcal{E}_1^{-1}$ and $\psi = \mathcal{E}'_1 \circ \mathcal{E}_2^{-1}$. For any $\mathbf{v}_1, \ldots, \mathbf{v}_\eta \in \mathcal{G}^k$, let f_i denote $\mathcal{E}_1^{-1}(\mathbf{v}_i)$. Since $w \geq \eta(k-1)+1$ by assumption, $\prod_{i=1}^{\eta} f_i(a)$ is mapped by $\mathcal{E}_2^{-1}$ to the polynomial of degree $\eta(k-1)$ given by $\prod_{i=1}^{\eta} f_i$. Hence,

$$\psi\left(\prod_{i=1}^{\eta} \phi(\mathbf{v}_i)\right) = \psi\left(\prod_{i=1}^{\eta} f_i(\xi)\right) = \left(\left(\prod_{i=1}^{\eta} f_i(x_1)\right), \ldots, \left(\prod_{i=1}^{\eta} f_i(x_k)\right)\right)$$
$$= \mathbf{v}_1 * \cdots * \mathbf{v}_\eta.$$

Thus (ϕ, ψ) is a $(k, w; \eta)$ $\mathcal{G}$-RMFE. □

Remark 1. A simplified version of the construction above, limited to $\mathcal{G} = \mathbb{Z}_{p^r}$ and leaving out ∞_k, is given in Section 1.1 of [14], giving $k \leq p$ in that case. On the other hand, when the characteristic of $\mathcal{G}$ is p (i.e. when $\mathcal{G}$ is a finite field), $\eta = 2$, and $w = \eta(k-1)+1 = 2k-1$, we recover the construction of interpolation RMFEs by Cascudo et al. in [7, Lemma 4].

Corollary 1. *Let $\mathcal{G} = GR(p^r, s)$. For any $(k, w; \eta)$ $\mathcal{G}$-RMFE constructed using Theorem 1, k is bounded above as follows:*

$$k \leq \min\left\{p^s + 1, \left\lfloor \frac{w-1}{\eta} \right\rfloor + 1\right\}.$$

Furthermore, there exists a $\mathcal{G}$-RMFE meeting this bound.

Proof. Recall the sufficient condition $k \leq p^s + 1$ and $w \geq \eta(k-1)+1$ from Theorem 1. Rearranging the second inequality, we get $k+1 \leq (w-1)/\eta$. Since k is an integer, $k \leq \lfloor (w-1)/\eta \rfloor + 1$ and the bound in Corollary 1 follows. Given $\mathcal{G}$, w, and η, setting the value of k to $k = \min\{p^s + 1, \lfloor (w-1)/\eta \rfloor + 1\}$ suffices to construct a $(k, w; \eta)$ $\mathcal{G}$-RMFE by Theorem 1, completing the proof. □

4.3 Evaluating the recode map

The recode map π is crucial to ensuring an RMFE-packed elements can undergo further multiplications. However, the map is costly to evaluate in the BGV scheme. In this section we review two methods to evaluate the recode map more efficiently, and implement both in our experiments later on.

Suppose we have a $(k, w; \eta)$ $\mathcal{G}$-RMFE (ϕ, ψ). By Proposition 3, π is of the form $\pi = \sum_{i=0}^{w-1} \theta_i \sigma^{si}$, where σ^s fixes $\mathcal{G}$ precisely.

- Method 1. Define projection maps $\tau_1, \ldots, \tau_k$ where τ_i decodes an RMFE-packed element α and extracts the ith component of the decoded RMFE vector. More precisely,

$$\tau_i : \mathcal{G}' \to \mathcal{G} : \alpha \mapsto \psi(\alpha) = (a_1, \ldots, a_k) \mapsto a_i.$$

  We can then evaluate π by computing

$$\pi(x) = \sum_{i=1}^{k} \tau_i(x)\phi(\mathbf{e}_i),$$

  where $\mathbf{e}_1, \ldots, \mathbf{e}_k$ are the canonical basis elements of $(\mathcal{G})^k$. By Proposition 4, $\tau_i(x) = \sum_{j=0}^{w-1} \sigma^{sj}(\theta_i x)$. It is a standard optimisation technique to compute this sum in only $\log w$ automorphisms by

$$\tau_i(x) = (id + \sigma^s) \circ (id + \sigma^{2s}) \circ (id + \sigma^{4s}) \circ \cdots \circ (id + \sigma^{(w/2)s})(\theta_i x).$$

  In total, this method computes $\pi(x)$ with $(k \log w)$-many automorphisms, roughly $(k + \log w)$-many additions, and $2k$ constant-ciphertext multiplications.
- Method 2. The standard Baby Step-Giant Step technique can be used to factor the evaluation of $\pi(x)$ as

$$\pi(x) = \sum_{i=0}^{\lceil w/\sqrt{w} \rceil} \sigma^{si} \left(\sum_{j=0}^{\lceil \sqrt{w} \rceil} \sigma^{-si}(\theta_{i,j}) \sigma^{sj}(x) \right),$$

  where some of the $\theta_{i,j}$'s might be 0. This requires roughly $2\lceil \sqrt{w} \rceil$-many automorphisms, and roughly w-many additions and constant-ciphertext multiplications in total.

5 Adapting Bootstrapping for RMFE

In this section, we describe how we realise bootstrapping for RMFE-packed ciphertexts. The main drawback of general bootstrapping is the need to perform d-many digit extractions. On the other hand, thin bootstrapping requires one digit extraction at the cost of packing fewer data within a single ciphertext. By leveraging on RMFEs, we can bootstrap integer data RMFE-packed in a thickly-packed ciphertext, using only a single digit extraction. We present our adaptation of the thin bootstrapping algorithm for RMFE-packed ciphertext for RMFE-packed ciphertexts in Algorithm 2.

Algorithm 2: RMFE Bootstrap

Input: Noisy RMFE-packed ciphertext c

$c_1, \cdots, c_k \longleftarrow$ `SlotsToCtxts`(c);
foreach c_i **do**
  `ModSwitch`: Raise the modulus of c_i to p^e.
  `SlotsToCoeff`: Send the ciphertext slots into polynomial coefficients.
  `InnerProduct`: Homomorphically multiply with bootstrapping key.
  `CoeffToSlots`: Send the polynomial coefficients into ciphertext slots.
end
$c' \longleftarrow$ `CtxtsToSlots`$(c_1, \cdots, c_k)$;
return `DigitExtract`(c');

5.1 Splitting the Ciphertext

We introduce two new auxiliary functions `SlotsToCtxts` and `CtxtsToSlots`. Let $\{x_{1,0}, \ldots, x_{k,0}, \ldots, x_{1,\ell}, \ldots, x_{k,\ell}\}$ be the message encrypted within a RMFE-packed ciphertext. We assume that all $x_{i,j}$ are integers lying in $\mathbb{Z}_p$.

The function `SlotsToCtxts` splits a single RMFE-packed ciphertext into k-many thinly-packed ciphertexts. More precisely, for $1 \leq i \leq k$, the i-th thin ciphertext encrypts $\{x_{i,0}, \ldots, x_{i,\ell}\}$. This can be achieved using either of the following two methods, and we provide performance results of both methods in our experiements later on.

- Method 1. Apply the projection maps $\tau_1, \ldots, \tau_k$ from Subsection 4.3 on the input. This method requires $k \log(w)$-many automorphisms in total.
- Method 2. Apply the recode map, so that for $1 \leq j \leq \ell$, the jth SIMD slot holds an RMFE-packed element of the form $\sum_{i=1}^{k} x_{i,j}\phi(\mathbf{e}_i) \in \operatorname{im}\phi$. Then we use maps $\tau'_1, \ldots, \tau'_k$ to extract the integer $x_{i,j}$, i.e.

$$\tau'_i : \operatorname{im}\phi \to \mathcal{G} : \sum_{i=1}^{k} x_{i,j}\phi(\mathbf{e}_i) \mapsto x_{i,j}.$$

  By Proposition 3, evaluating all projection maps $\tau'_1, \ldots, \tau'_k$ only requires the same k-many automorphisms on the input. This method requires roughly $(2\sqrt{w} + k)$-many automorphisms in total.

The function `CtxtsToSlots` combines k-many thinly-packed ciphertexts into one RMFE-packed ciphertext. Instead of linear maps, the simpler (and cheaper) way to combine the various ciphertexts would be to first constant-multiply each ciphertext with the appropriate basis element of $\operatorname{im}\phi$, and then summing them together to reconstruct an RMFE-packed ciphertext. We remark that Method 1 in Subsection 4.3 is basically calling `SlotsToCtxts` and then `CtxtsToSlots`.

As an additional functionality, it is also possible to use `CtxtsToSlots` to compute other linear maps $L : \operatorname{im}\phi \to \mathcal{G}'$ by similarly multiplying the split ciphertexts with the appropriate precomputed constants from the image of the basis of $\operatorname{im}\phi$ under L, and then summing them together.

5.2 Digit extraction with RMFE

Let (ϕ, ψ) be a $\mathcal{G}$-RMFE where $\mathcal{G} = GR(p^e, s)$, where $e \geq 2$ and s a proper divisor of d. From now on, we assume that we only encode vectors $(a_1, \ldots, a_k) \in (\mathbb{Z}_{p^e})^k \subseteq \mathcal{G}^k$. In normal bootstrapping, the digit extraction algorithm acts on elements in the subring $\mathbb{Z}_{p^e}$ in each SIMD slot. In our case, we have RMFE-encoded elements in each SIMD slot, and we wish to apply the digit extraction algorithm on these RMFE-encoded elements while carrying over the operations of digit extraction to the RMFE slots containing integers. However, there is a technical challenge which we illustrate as follows.

The problem. Let F_e denote the Halevi-Shoup [19] lifting polynomial. A crucial computation in digit extraction is given by the formula $(x - F_e(x))/p$. However, we wish to do this same computation on RMFE-encoded elements. That is, for $\beta = \phi((a_1, \ldots, a_k))$ we want to perform the following (unworkable) computation:

$$\begin{aligned} \gamma' &\leftarrow \beta - F_e(\beta), \\ \gamma &\leftarrow \gamma'/p, \end{aligned}$$

with the effect that

$$\psi(\gamma) = \left(\frac{a_1 - F_e(a_1)}{p}, \ldots, \frac{a_k - F_e(a_k)}{p} \right) \bmod p^{e-1}.$$

This computation is likely to fail. While it is true that γ' encodes multiples of p since $\psi(\gamma') = (a_1 - F_e(a_1), \ldots, a_k - F_e(a_k))$ by Proposition 7, γ' itself is usually not a multiple of p since β is allowed to be a ring extension element.

More generally, several steps in the digit extraction algorithm must yield a multiple of p for the algorithm to continue, but they only do so if the input were an integer. We first present a naive workaround, followed by our new technique.

Naive solution. For this solution we require a $(k, w; \eta)$ RMFE (ϕ, ψ) where $\eta \geq p$. The idea is to simply apply the recode map π after every evaluation of the lifting polynomial F_e, as shown in Algorithm 3. Let $\{\mathbf{e}_1, \ldots, \mathbf{e}_k\}$ be the canonical basis of $(\mathcal{G})^k$. We may assume that the element y in Algorithm 3 begins with the form

$$y = \phi((a_1, \ldots, a_k)) = \sum_{t=1}^{k} a_t \phi(\mathbf{e}_t)$$

since the input to digit extraction is a fresh RMFE encoding. Recoding the term $u_{i,j+1}$ resets it to be of the same form as y, with the RMFE data being the coefficients. Thus we have that

$$y - \pi(F_e(u_{i,j+1})) = \sum_{t=1}^{k} a_t \phi(\mathbf{e}_t) - \sum_{t=1}^{k} F_e(b_t)\phi(\mathbf{e}_t) = \sum_{t=1}^{k} (a_t - F_e(b_t))\phi(\mathbf{e}_t).$$

where $b_1, \ldots, b_k$ are the integers RMFE-packed into $u_{i,j+1}$. Each $(a_t - F_e(b_t))$ is divisible by p following the normal digit extraction algorithm, hence $y -$

$\pi(F_e(u_{i,j+1}))$ as a whole is divisible by p. Furthermore,

$$\psi(y/p) = \left(\frac{a_1 - F_e(b_1)}{p}, \ldots, \frac{a_k - F_e(b_k)}{p}\right) \bmod p^{e-1}.$$

as desired, carrying over the action of digit extraction to the RMFE-packed data.

Algorithm 3: Naive Digit Extraction with RMFE

Input: $z \in \operatorname{im}\phi, e \in \mathbb{Z}$

$u_{0,0} \longleftarrow z$;
for $j = 0, \ldots, e-2$ **do**
  $y \leftarrow z$;
  for $i = 0, \ldots, j$ **do**
    $u_{i,j+1} \leftarrow F_e(u_{i,j})$;
    $y \leftarrow (y - \texttt{Recode}(u_{i,j+1}))/p$; `// recode after evaluating` F_e
  end
  $u_{j+1,j+1} \leftarrow y$;
end
return $u_{e-1,e-1}$;

Our solution. The naive solution requires $(\sum_{n=1}^{e-1} n)$-many evaluations of the recode map, which can be costly in terms of latency and noise. We present our new solution using novel correction maps that require only one evaluation of the recode map at the end, no matter the value of e. This technique requires a $(k, w; \eta)$ RMFE (ϕ, ψ) where $\eta \geq p^{e-1}$. For $0 \leq i \leq e-1$, define correction maps $L_i : \operatorname{im}\phi \to E$ such that for all $x = \phi(x_1, \ldots, x_k)$,

$$L_i(x) = \sum_{i=1}^{k} x_i \phi(\mathbf{e}_i)^{p^i}.$$

Observe that L_0 is simply the identity map on $\operatorname{im}\phi$.

Algorithm 4 shows how these maps are introduced into digit extraction. Each $L_i(x)$ can be computed cheaply before digit extraction using just additions and constant multiplications in the `CtxtsToSlots` step. The evaluation of the recode map at the end does not require any ciphertext-ciphertext multiplications. Hence, the multiplicative degree of Algorithm 4 is identical to that of conventional digit extraction given in Algorithm 1.

Before showing that our technique overcomes the divisibility by p problem, we require the following helpful lemmas.

Lemma 1. *Let $f(X) \in \mathbb{Z}_{p^e}[X]$ be a monic polynomial of the form $f(X) = X^p + pg(X)$ for any $g(X) \in \mathbb{Z}_{p^e}[X]$. Denote by f^n the n-fold composition of f. For any $\alpha, \beta \in GR(p^s, e)$ and $1 \leq n \leq e-1$, if $\alpha \equiv \beta \mod p$ then $f^n(\alpha) \equiv f^n(\beta) \mod p^{n+1}$.*

Proof. Write β as $\alpha + p\varepsilon$ for some $\varepsilon \in GR(p^s, e)$. We proceed by induction on n. Consider the base case $n = 1$. We first have $f(\beta) = (\alpha+p\varepsilon)^p + pg(\alpha+p\varepsilon)$. By the Binomial Theorem, $(\alpha+p\varepsilon)^p \equiv \alpha^p \mod p^2$. Next, observe that $g(\alpha+p\varepsilon) \equiv g(\alpha) \mod p$, meaning $g(\alpha+p\varepsilon) = g(\alpha)+px$ for some $x \in GR(p^s, e)$. Multiplying both sides by p and taking modulo p^2, we get $pg(\alpha + p\varepsilon) \equiv pg(\alpha) \mod p^2$. Hence, $f(\beta) \equiv \alpha^p + pg(\alpha) \equiv f(\alpha) \mod p^2$ as required.

Assuming $f^n(\alpha) \equiv f^n(\beta) \mod p^{n+1}$ for some $1 \leq n \leq e-2$, we want to show that $f^{(n+1)}(\alpha) \equiv f^{(n+1)}(\beta) \mod p^{n+2}$. By the inductive hypothesis,

$$f^{(n+1)}(\beta) = f\left(f^n(\alpha) + p^{n+1}y\right) = \left(f^n(\alpha) + p^{n+1}y\right)^p + pg\left(f^n(\alpha) + p^{n+1}y\right)$$

for some $y \in GR(p^s, e)$. The mod-p^{n+2} binomial expansion of $(f^n(\alpha) + p^{n+1}y)^p$ yields $(f^n(\alpha))^p$. We have that $g\left(f^n(\alpha) + p^{n+1}y\right) \equiv g(f^n(\alpha)) \mod p^{n+1}$, implying $g(f^n(\alpha) + p^{n+1}y) = g(f^n(\alpha)) + p^{n+1}z$ for some z. Multiplying both sides by p and taking modulo p^{n+2}, we get $pg(f^n(\alpha) + p^{n+1}y) \equiv pg(f^n(\alpha)) \mod p^{n+2}$. It follows that $f^{(n+1)}(\beta) \equiv (f^n(\alpha))^p + pg(f^n(\alpha)) \equiv f^{(n+1)}(\alpha) \mod p^{n+2}$. □

Lemma 2. *For any $\alpha \in GR(p^s, e)$ and $1 \leq n \leq e$, we have $\alpha^{p^n} \equiv \sigma(\alpha^{p^{n-1}}) \mod p^n$.*

Proof. The proof is trivial when $n = 1$. For $n \geq 2$, let $f(X) = X^p \in \mathbb{Z}_{p^e}[X]$. Write α^{p^n} as $f^{n-1}(\alpha^p)$ and $\sigma(\alpha^{p^{n-1}})$ as $f^{n-1}(\sigma(\alpha))$. Noting that $\alpha^p \equiv \sigma(\alpha) \mod p$, Lemma 2 then follows by application of Lemma 1. □

To show that Algorithm 4 resolves the problem of divisibility by p, we proceed with an induction-style argument. We want to show that if division by p is possible in the j-th iteration of the outer loop, then it is likewise possible in the $(j+1)$-th iteration.

Assume that the input is $z = \sum_{t=1}^{k} a_t\phi(\mathbf{e}_t)$ where $a_1, \ldots, a_k \in \mathbb{Z}_{p^e}$, and write $F_e(X)$ as $X^p + pG(X)$. We begin with the base case where $j = 0$, $y \leftarrow L_1(z)$, and $u_{0,0} \leftarrow z$. Then $u_{0,1} \leftarrow F_e(z)$. We thus have that

$$\begin{aligned} y - u_{0,1} &= L_1(z) - z^p - pG(z) \\ &= \left(\sum_{t=1}^{k} a_t\phi(\mathbf{e}_t)^p\right) - \left(\sum_{t=1}^{k} a_t^p\phi(\mathbf{e}_t)^p\right) - \epsilon - pG(z) \\ &= \left(\sum_{t=1}^{k} (a_t - a_t^p)\phi(\mathbf{e}_t)^p\right) - \epsilon - pG(z), \end{aligned}$$

where ϵ is the component that is a multiple of p arising from the binomial expansion of z^p. Since $a_t \in \mathbb{Z}_{p^e}$, we know that $a_t - a_t^p$ is divisible by p. Thus the entire expression is also divisible by p, as required.

Assume that division by p is possible for some j-th iteration of the outer loop, where $j \geq 0$. This means that the following expression holds:

$$\frac{\frac{\frac{L_{j+1}(z) - u_{0,j+1}}{p} - u_{1,j+1}}{p} - \cdots}{p} - u_{j,j+1} \over p} = u_{j+1,j+1} \mod p^{e-j-1}, \tag{1}$$

Observe that Equation 1 can be rewritten as

$$L_{j+1}(z) - u_{0,j+1} - pu_{1,j+1} - \cdots - p^{j+1}u_{j+1,j+1} = 0 \mod p^e. \tag{2}$$

Note that $u_{i,j} = F_e^{(j-i)}(u_{i,i})$. Substituting, we get

$$L_{j+1}(z) - F_e^{(j+1)}(u_{0,0}) - pF_e^{(j)}(u_{1,1}) - \cdots - p^{j+1}u_{j+1,j+1} = 0 \mod p^e.$$

We want to show that assuming Equation 2 holds, then the $(j+1)$-th iteration can be computed. More precisely, it suffices to show that the expression

$$L_{j+2}(z) - F_e^{(j+2)}(u_{0,0}) - pF_e^{(j+1)}(u_{1,1}) - \cdots - p^{j+1}F_e(u_{j+1,j+1}) \tag{3}$$

is divisible by p^{j+2}. Indeed, by Lemma 1 we have that

$$F_e^{(j+2-i)}(u_{i,i}) \equiv F_e^{(j+1-i)}(\sigma(u_{i,i})) \mod p^{j+2-i},$$

which implies

$$p^i F_e^{(j+2-i)}(u_{i,i}) \equiv p^i F_e^{(j+1-i)}(\sigma(u_{i,i})) \mod p^{j+2}.$$

Substituting this into Expression 3 and taking modulo p^{j+2}, we get

$$L_{j+2}(z) - F_e^{(j+1)}(\sigma(u_{0,0})) - pF_e^{(j)}(\sigma(u_{1,1})) - \cdots - p^{j+1}\sigma(u_{j+1,j+1}) \tag{4}$$

$$\equiv L_{j+2}(z) - \sigma\left(F_e^{(j+1)}(u_{0,0}) + pF_e^{(j)}(u_{1,1}) + \cdots + p^{j+1}u_{j+1,j+1}\right) \tag{5}$$

$$\equiv L_{j+2}(z) - \sigma\left(L_{j+1}(z)\right) \mod p^{j+2}, \tag{6}$$

where the step from Equation 5 to 6 is implied by Equation 2. To complete the proof, we are left to show that $L_{j+2}(z) - \sigma\left(L_{j+1}(z)\right) \equiv 0 \mod p^{j+2}$. Indeed, by Lemma 2 we have that

$$L_{j+2}(z) - \sigma\left(L_{j+1}(z)\right) \equiv \sum_{i=0}^{k} a_i \left(\phi(\mathbf{e}_i)^{p^{j+2}} - \sigma\left(\phi(\mathbf{e}_i)^{p^{j+1}}\right)\right) \equiv 0 \mod p^{j+2}.$$

By enabling division by p, Algorithm 4 runs without failure. Since η is assumed to be at least p^{e-1}, the action of digit extraction is carried over correctly to the integers RMFE-packed into the input z.

5.3 RMFE parameter selection

In the context of BGV, we fix the prime p and a cyclotomic order m, which in turn fixes the SIMD slot degree d. The SIMD slot algebra just before digit extraction is of the form $E = GR(p^e, d)$. We set $\eta = p$ for Algorithm 3, and set $\eta = p^{e-1}$ for Algorithm 4. By Proposition 1, we have a tower of Galois rings

$$GR(p^e, s) \subseteq GR(p^e, sw) \subseteq E$$

Algorithm 4: Digit Extraction with RMFE and correction mapping

Input: $z \in \operatorname{im}\phi, e \in \mathbb{Z}$
$u_{0,0} \longleftarrow z$;
for $j = 0, \ldots, e-2$ **do**
  $y \leftarrow L_{j+1}(z)$;
  for $i = 0, \ldots, j$ **do**
    $u_{i,j+1} \leftarrow F_e(u_{i,j})$;
    $y \leftarrow (y - u_{i,j+1})/p$;
  end
  $u_{j+1,j+1} \leftarrow y$;
end
$u_{e-1,e-1} \leftarrow \texttt{Recode}(u_{e-1,e-1})$;
return $u_{e-1,e-1}$;

where $s, w \geq 1$ and sw divides d. Let $\mathcal{G} = GR(p^e, s)$ and $\mathcal{G}' = GR(p^e, sw)$. Theorem 1 allows us to build a $\mathcal{G}$-RMFE given by $\phi : \mathcal{G}^k \to \mathcal{G}' \subseteq E$ and $\psi : \mathcal{G}' \to \mathcal{G}^k$. In practice, we restrict the input space of ϕ to $(\mathbb{Z}_{p^e})^k \subseteq \mathcal{G}^k$, though the map itself is actually $\mathcal{G}$-linear by construction. The question then is which parameter should be the independent and dependent variables for conducting experiments. In our work, we chose w as the independent variable.

Motivation. Assuming encoding and decoding can be done in the clear, we need only concern ourselves with the encrypted evaluation of the recode map $\pi = \phi \circ \psi : \mathcal{G}' \to \mathcal{G}'$. By Proposition 3, so long as sw divides d, π is always represented as an E-linear combination of w-many automorphisms of E no matter the value of s. Hence the cost of recoding depends only on w.

It is observed experimentally that recoding is a major bottleneck in RMFE-bootstrapping, hence it makes sense to take w as an independent variable, and find the best RMFEs of dimension w that can encode into E. The following example illustrates how RMFEs are chosen based on E and w.

Example 1. Let $E = GR(11^3, 8192)$. Given $w = 1024$, we construct two RMFEs of dimension w by Theorem 1. First, let $s = 1$. We construct an RMFE (ϕ_1, ψ_1)

$$\phi_1 : (\mathbb{Z}_{11^3})^{k_1} \to GR(11^3, 1024) \subseteq E,$$
$$\psi_1 : GR(11^3, 1024) \to (\mathbb{Z}_{11^3})^{k_1}.$$

By Corollary 1, maximising k_1 yields $k_1 = \min\{11 + 1, \lfloor 1023/11 \rfloor + 1\} = 12$.

Now, let $s = 2$. Then the we have an RMFE (ϕ_2, ψ_2)

$$\phi_2 : (\mathbb{Z}_{11^3})^{k_2} \subseteq GR(11^3, 2)^{k_2} \to GR(11^3, 2048) \subseteq E,$$
$$\psi_2 : GR(11^3, 2048) \to GR(11^3, 2)^{k_2}.$$

By Corollary 1 again, maximising k_2 yields $k_2 = \min\{121 + 1, \lfloor 1023/11 \rfloor + 1\} = 94$. By Proposition 3, representing π_2 takes the same number of E-elements as π_1. Notice that using higher values of $s > 2$ will still yield $k_s = 94$. Thus we pick (ϕ_2, ψ_2) as our candidate RMFE associated to $w = 1024$ and E.

6 Implemenation and Performance

6.1 Implementation Details

The experiments were performed on an Intel® Xeon® Platinum 8368Q CPU with a maximum turbo frequency of 3.7 GHz and 571 GB RAM. Multi-threading was not used for the experiments.

Initialisation. The cyclotomic polynomial $\Phi_{32768}(X) = X^{16384} + 1$ was used with varying primes p. The encryption scheme was initialized using HElib [22], with $\mathsf{bits} = 400$, resulting in the maximum ciphertext bit-width ranging from 538 to 541. Our implementation was estimated to have achieved at least 100-bit security, using the lattice estimator by Albrecht et al. [1]. Magma [4] was used to generate the RMFEs. The choice of primes and RMFE dimensions selected for the experiments are given in Table 1.

Prime, p	$3, 5, 7, 11, 13, 53, 59$
RMFE Dimension, w	$128, 256, 512, 1024$

Table 1. Parameter choices.

Workflows tested. We implement two RMFE bootstrapping styles. The first *correction* style uses our novel correction map technique in digit extraction. Furthermore, `CtxtsToSlots` is used to compute the ciphertext to be passed into digit extraction along with the images of this ciphertext under the correction maps using constant multiplications and ciphertext additions. For this style, we use RMFEs with $\eta = p^{e-1}$. The second *naive* style uses the naive RMFE digit extraction procedure of recoding after every evaluation of the lifting polynomial. This style uses RMFEs with $\eta = p$.

Each bootstrapping style is further divided into two variants. The first variant, referred to as *trace-and-merge*, uses Method 1 for `SlotsToCtxts` and Method 1 for evaluating the recode map. The second variant, *BSGS*, uses Method 2 for `SlotsToCtxts` and Method 2 for evaluating the recode map.

We thus have four bootstrapping workflows in total, which are benchmarked against conventional thin bootstrapping. For bootstrapping parameters where $e = 2$, we only test the performance of the naive style, since the correction style does not reduce the number of recodes further in that case.

6.2 Measuring and Recording Performance

The complete experimental results are provided in Tables 4 and 5 in Appendix A. All timings are averaged over ten runs. For each bootstrapping workflow we measure the number of rounds of squaring performed on the bootstrapped ciphertext,

recoding where necessary, before decryption fails. The minimum among the ten runs is reported as the number of remaining multiplications for that workflow.

To measure holistic performance, we compute the *throughput* metric given by the following formula:

$$\text{throughput} := \frac{\text{post-bootstrap capacity} \times \#\text{slots}}{\text{latency}},$$

where $\#\text{slots} := k \cdot \ell$, i.e. the total number of slots of an RMFE-packed ciphertext. Latency is the average time taken (in seconds) for the bootstrapping workflow to complete. For comparison, we compute and report the ratio of the throughput of our workflows to that of vanilla thin bootstrapping.

For $e \geq 3$, the performance of the trace-and-merge variant of the naive style is too poor to consider, often producing ciphertexts that either cannot undergo further multiplications or are no longer decryptable. This is likely due to the fact that trace-and-merge recoding consumes more capacity than BSGS recoding, and it is applied too many times when in the naive style. Hence, their results are omitted from Table 5.

6.3 Experimental Results

Process	Corr. (TnM), $k = 5$, #rec = 1		Corr. (BSGS), $k = 5$, #rec = 2		Naive (TnM) $k = 103$, #rec = 6		Naive (BSGS), $k = 103$, #rec = 7	
	Capacity	Latency	Capacity	Latency	Capacity	Latency	Capacity	Latency
Initial	391.1	–	391.1	–	391.1	–	391.1	–
`SlotsToCtxts`	369.2	2.284	340.4	4.142	366.6	58.47	332.7	46.27
`MultKey`	346.5	4.732	346.6	4.753	346.5	102.3	346.7	99.26
`CtxtsToSlots`	333.8	0.078	333.7	0.075	325.3	0.418	325.3	0.410
`DigitExtract`	101.0	4.240	115.2	6.07	−6.61	377.7	57.63	32.61
Final	101.0	11.33	115.2	15.04	−6.61	538.9	57.63	178.5

Table 2. Breakdown of timings (in seconds) and changes in ciphertext capacity for bootstrapping workflows for $m = 32768$, $p = 5$, $w = 512$. The number of recodes called for each workflow is given by #rec.

Correction versus naive. Table 2 provides a detailed breakdown of timings and changes in ciphertext capacity for the various RMFE bootstrapping workflows. The process `MultKey` refers to the execution of lines 3 to 6 in Algorithm 2, that is, `ModSwitch`, `InnerProduct`, `SlotsToCoeff`, and `CoeffToSlots`.

From Table 2, the use of correction maps resulted in lower capacity loss for RMFE digit extraction. In the trace-and-merge variant where naive RMFE digit extraction produced an undecryptable ciphertext with -6 levels of capacity, using correction maps yielded 101 levels. In the BSGS variant, using correction maps raised the remaining ciphertext capacity from 57 to 115 levels.

Additionally, we observe improvements in the latency of RMFE digit extraction. Using correction maps instead of naively recoding multiple times reduced the execution time of RMFE digit extraction from 377.7 to 4.240 seconds for the trace-and-merge variant. For the BSGS variant, we observe a reduction in latency from 32.61 to 6.070 seconds. These improvements can be attributed to the fact that the use of correction maps reduces the number of recodes required to just one recode at the end.

Table 3 illustrates the practical impact of the RMFE bootstrapping workflows compared to conventional thin bootstrapping by comparing the number of multiplications remaining, the throughput, and a total latency for various p.

The number of recodes needed for the naive style is dependent on the value of e. Hence we expect correction style bootstrapping to yield stronger savings in total latency and capacity consumption over the naive style for larger e (and consequently smaller p). This trend can be observed in Table 3, where correction maps produced an additional 2, 4, and 6 multiplicative depth for $e = 3, 4, 5$ respectively. This is a signifcant increase from the 2-3 multiplications allowed by the naive BSGS workflow shown in the same table.

Prime, p (e)	Workflow Style/Variant	RMFE Count, k	Capacity Before/After	Remaining Mults	Latency (sec)	Throughput (capacity×slots/sec)	Ratio
3 (5)	-	1*	392/179	10	2.14	167.19	1.000
	Corr./TnM	7	392/136	7	14.9	127.70	0.764
	Corr./BSGS	7	392/150	8	16.7	126.00	0.754
	Naive/BSGS	171	391/67	2	342	67.00	0.401
5 (4)	-	1*	391/151	8	2.82	107.06	1.000
	Corr./TnM	5	391/101	5	11.3	89.08	0.832
	Corr./BSGS	5	391/115	6	15.0	76.48	0.714
	Naive/BSGS	103	391/58	2	179	66.93	0.625
11 (3)	-	1*	390/144	7	2.62	109.87	1.000
	Corr./TnM	5	390/94	4	11.1	84.98	0.773
	Corr./BSGS	5	390/108	5	14.7	73.22	0.666
	Naive/BSGS	47	390/77	3	73.3	98.71	0.898

*HElib thin bootstrapping

Table 3. Comparison of bootstrapping timings, remaining multiplications, and throughput for each workflow against HElib thin bootstrapping for $m = 32768$, $w = 512$, and various p.

Trace-and-merge versus BSGS. Evaluating the recode map takes $(k \log w)$-many automorphisms in the trace-and-merge variant, and roughly $(2\lceil\sqrt{w}\rceil)$ in the BSGS variant. Thus we expect the performance of trace-and-merge variants to benefit from small values of k, while the BSGS variants perform best when k is large. Due to the recursive nature of the trace evaluation, trace-and-merge usually consumes more capacity than the BSGS variant for the same style.

This can be seen in detail in Table 2. For the correction style where $k = 5$, RMFE digit extraction in the trace-and-merge variant takes almost 2 seconds less at the cost of about 14 more levels of capacity compared to the BSGS variant despite using the same number of recode calls. On the other hand for the naive style where $k = 103$, the latency of RMFE digit extraction is more than $10\times$ longer in the trace-and-merge variant compared to the BSGS variant. Furthermore the ciphertext produced by the naive trace-and-merge workflow is undecryptable, while the BSGS variant still has 57.63 levels.

Generally the trace-and-merge technique improves the performance of the correction style RMFE boostrapping, as illustrated in Table 3.

Holistic performance. Generally, using RMFE bootstrapping results in lower throughput compared to conventional thin bootstrapping. This can be attributed to increases in latency and capacity consumption in RMFE bootstrapping.

For the smallest primes $p = 3, 5$, we observe that using our correction map technique in conjunction with the trace-and-merge recoding reduces the degradation in throughput compared to just using the naive BSGS workflow. This is illustrated in Table 3, where the naive BSGS workflows for $p = 3, 5$ have the worst throughput ratios, and the correction trace-and-merge workflows have the best. This is because the correction map technique significantly reduces capacity loss and latency, and the trace-and-merge recoding further improves latency.

For $p \geq 7$, naive style bootstrapping yields the best throughputs out of the RMFE bootstrapping workflows. This is likely because naive RMFE digit extraction uses less recodes for the larger values of p compared to the smaller values. Hence, savings in capacity and latency by the correction map and trace-and-merge techniques are not as pronounced, and the larger value of k admitted by RMFEs for the naive style begin to contribute more significantly to the throughput. However, we stress that even in this case, the number of multiplications remaining after naive BSGS workflow is often very low, and our correction map technique is crucial in practice to raise the number multiplications and bring the latency down to a more reasonable timing as illustrated in Table 3.

7 Conclusion and Future Work

In this work we developed bootstrapping for RMFE-packed ciphertexts, analogous to bootstrapping for thinly-packed ciphertexts for the case $r = 1$. Our main contribution is the new technique of using novel correction maps to enable Halevi-Shoup digit extraction on an RMFE-packed Galois ring element without changing the multiplicative degree and requiring only a single recode at the end. Our method distinguishes itself from conventional digit extraction that requires the input to be an integer, and further distinguishes itself from the naive workaround of recoding after every evaluation of the lifting polynomial which incurs a large cost in latency and capacity for larger e.

We extend the thin bootstrapping procedure to take an RMFE-packed ciphertext as input, while still performing a single digit extraction. We proposed two RMFE bootstrapping styles; the *correction* style which uses our new digit

extraction technique, and the *naive style* which performs RMFE digit extraction by recoding after every evaluation of the lifting polynomial. Each style is further subdivided into two variants; the trace-and-merge variant which leverages the recursive evaluation of trace maps, and the BSGS variant which does not.

All four RMFE bootstrapping workflows were implemented in HElib and benchmarked against conventional thin bootstrapping, comparing latency, multiplications remaining, and throughput.

Our experimental results show that our correction map technique is effective at reducing capacity loss, providing up to 6 additional multiplicative depth compared to the naive style of RMFE bootstrapping, which usually yields an impractically low multiplicative depth of 2 to 3. Improvements in latency were also observed. Trace-and-merge recoding was shown to further improve latency, at the cost of some capacity. Using these two techniques together increases remaining multiplicative depth post-bootstrapping multiplications to a practically useful number while bringing down the latency to a reasonable timing, at the cost of lower RMFE packing capacity.

As future work, extending the use of correction maps to Chen-Han digit extraction [8] can be investigated. The recode map continues to be a significant bottleneck, making it a good target for further optimisation. It also remains to be seen whether the `MultKey` portion can be done directly on RMFE-packed ciphertext to avoid executing `MultKey` multiple times on the split ciphertexts.

This work represents the first non-trivial effort to unlock bootstrapping for the FIMD technique [2], and gives an early systematic attempt at extending the input space of digit extraction beyond the integers. It is hoped that the techniques proposed in this work will inspire further research in these two directions.

References

1. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Cryptology ePrint Archive, Paper 2015/046 (2015)
2. Aung, K.M.M., Lim, E., Sim, J.J., Tan, B.H.M., Wang, H., Yeo, S.L.: Field instruction multiple data. In: EUROCRYPT 2022, Part I. LNCS, vol. 13275, pp. 611–641. Springer, Cham (2022)
3. Block, A.R., Maji, H.K., Nguyen, H.H.: Secure computation with constant communication overhead using multiplication embeddings. In: INDOCRYPT 2018. LNCS, vol. 11356, pp. 375–398 (2018)
4. Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system. I. The user language. J. Symbolic Comput. **24**(3-4), 235–265 (1997), computational algebra and number theory (London, 1993)
5. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: CRYPTO 2012. LNCS, vol. 7417, pp. 868–886 (2012)
6. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: ITCS 2012. pp. 309–325. ACM (2012)
7. Cascudo, I., Cramer, R., Xing, C., Yuan, C.: Amortized complexity of information-theoretically secure MPC revisited. In: CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 395–426. Springer, Cham (2018)

8. Chen, H., Han, K.: Homomorphic lower digits removal and improved FHE bootstrapping. In: EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 315–337. Springer, Cham (2018)
9. Cheon, J.H., Kim, A., Kim, M., Song, Y.S.: Homomorphic encryption for arithmetic of approximate numbers. In: ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 409–437 (2017)
10. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In: ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 3–33 (2016)
11. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In: ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 377–408 (2017)
12. Cramer, R., Rambaud, M., Xing, C.: Asymptotically-good arithmetic secret sharing over $\mathbb{Z}/p^{\ell}\mathbb{Z}$ with strong multiplication and its applications to efficient MPC. In: CRYPTO 2021, Part III. LNCS, vol. 12827, pp. 656–686. Springer, Cham (2021)
13. Ducas, L., Micciancio, D.: FHEW: Bootstrapping homomorphic encryption in less than a second. In: EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 617–640 (2015)
14. Escudero, D., Liu, H., Xing, C., Yuan, C.: Degree-D reverse multiplication-friendly embeddings: Constructions and applications. Cryptology ePrint Archive, Report 2023/173 (2023)
15. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Paper 2012/144 (2012)
16. Geelen, R., Vercauteren, F.: Bootstrapping for BGV and BFV revisited. Journal of Cryptology **36**(12), 1432–1378 (Mar 2023). https://doi.org/10.1007/s00145-023-09454-6
17. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: 41st ACM STOC. pp. 169–178. ACM Press (2009)
18. Gentry, C., Halevi, S., Smart, N.P.: Better bootstrapping in fully homomorphic encryption. In: PKC 2012. LNCS, vol. 7293, pp. 1–16. Springer, Berlin, Heidelberg (2012)
19. Halevi, S., Shoup, V.: Bootstrapping for HElib. In: EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 641–670. Springer, Berlin, Heidelberg (2015), Updated in 2020 at https://eprint.iacr.org/2014/873
20. Halevi, S., Shoup, V.: Faster homomorphic linear transformations in HElib. In: CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 93–120. Springer, Cham (2018)
21. Halevi, S., Shoup, V.: Design and implementation of HElib: a homomorphic encryption library. Cryptology ePrint Archive, Report 2020/1481 (2020)
22. Helib. https://github.com/homenc/HElib (Sep 2022), commit acdb264
23. Iliashenko, I., Zucca, V.: Faster homomorphic comparison operations for BGV and BFV. Proceedings on Privacy Enhancing Technologies **2021**(3), 246–264 (2021). https://doi.org/10.2478/popets-2021-0046
24. Lattigo v2.2.0. Online: http://github.com/ldsec/lattigo (Jul 2021), ePFL-LDS
25. Meftah, S., Tan, B.H.M., Mun, C.F., Aung, K.M.M., Veeravalli, B., Chandrasekhar, V.: Doren: Toward efficient deep convolutional neural networks with fully homomorphic encryption. IEEE Transactions on Information Forensics and Security **16**, 3740–3752 (2021). https://doi.org/10.1109/TIFS.2021.3090959
26. PALISADE lattice cryptography library (release 1.11.5). https://gitlab.com/palisade/palisade-release (Sep 2021), PALISADE Project
27. Quintin, G., Barbier, M., Chabot, C.: On generalized reed–solomon codes over commutative and noncommutative rings. IEEE transactions on information theory **59**(9), 5882–5897 (2013)

28. Raghavendran, R.: Finite associative rings. Compositio Mathematica **21**(2), 195–229 (1969)
29. Roman, S.: Field Theory. Graduate Texts in Mathematics, Springer New York (2013)
30. Microsoft SEAL (release 3.6). `https://github.com/Microsoft/SEAL` (Nov 2020), microsoft Research, Redmond, WA.
31. Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. DCC **71**(1), 57–81 (2014)
32. Wan, Z.X.: Lectures on finite fields and Galois rings. World Scientific Publishing Company (2003)

A Complete Experiment Results

Prime, p $(e, \log_2 p)$	Workflow	RMFE Dimension, w	RMFE Count, k	Capacity Before/After	Remaining Mults	Latency (sec)	Throughput (capacity×slots/sec)	Ratio
53 $(2, 5.728)$	Thin Bootstrap	–	1	389/184	9	2.01	182.90	1.000
	Naive (trace-and-merge)	128	3	389/137	5	5.65	145.63	0.796
		256	5	389/129	5	9.47	136.22	0.745
		512	10	389/123	5	19.5	125.95	0.689
		1024	20	389/120	5	41.4	115.80	0.633
	Naive (BSGS)	128	3	389/150	6	6.59	136.50	0.746
		256	5	389/144	6	10.6	136.22	0.745
		512	10	389/141	5	19.0	148.05	0.809
		1024	20	389/140	5	35.6	157.22	0.860
59 $(2, 5.883)$	Thin Bootstrap	–	1	388/182	8	2.12	171.99	1.000
	Naive (trace-and-merge)	128	3	389/131	5	5.80	135.45	0.788
		256	5	389/128	5	9.61	133.25	0.775
		512	9	389/123	5	17.7	124.72	0.682
		1024	18	389/117	5	37.8	111.38	0.609
	Naive (BSGS)	128	3	389/146	6	6.74	129.94	0.756
		256	5	389/143	6	10.7	133.42	0.776
		512	9	389/140	5	18.1	139.3	0.810
		1024	18	389/136	5	33.8	144.84	0.842

Table 4. Experimental results for $m = 32768$, $e = 2$, and varying p, w.

Prime, p $(e, \log_2 p)$	Workflow	RMFE Dimension, w	RMFE Count, k	Capacity Before/After	Remaining Mults	Latency (sec)	Throughput (capacity×slots/sec)	Ratio
3 (5, 1.585)	Thin Bootstrap	–	1	392/179	10	2.14	167.19	1.000
	Correction (trace-and-merge)	128	2	392/146	7	4.36	134.03	0.802
		256	4	392/140	7	8.38	133.70	0.800
		512	7	392/136	7	14.9	127.70	0.764
		1024	13	391/129	6	30.5	109.91	0.657
	Correction (BSGS)	128	2	392/155	8	5.67	109.28	0.654
		256	4	392/152	8	10.5	115.52	0.691
		512	7	392/150	8	16.7	126.00	0.754
		1024	13	391/145	7	31.2	120.93	0.723
	Naive (BSGS)	128	43	392/85	2	62.1	117.64	0.704
		256	86	392/76	2	142	92.19	0.551
		512	171	392/67	2	342	67.00	0.401
5 (4, 2.322)	Thin Bootstrap	–	1	391/151	8	2.82	107.06	1.000
	Correction (trace-and-merge)	128	2	392/110	5	4.93	89.32	0.834
		256	3	392/108	5	7.05	91.91	0.858
		512	5	392/101	5	11.3	89.08	0.832
		1024	9	391/98	4	21.2	83.30	0.778
	Correction (BSGS)	128	2	392/120	6	6.40	75.00	0.701
		256	3	392/120	6	9.53	75.60	0.706
		512	5	392/115	6	15.0	76.48	0.714
		1024	9	391/113	5	25.5	79.67	0.744
	Naive (BSGS)	128	26	392/68	2	41.7	84.80	0.792
		256	52	392/65	2	83.0	81.38	0.760
		512	103	392/58	2	179	66.93	0.625
		1024	205	391/52	2	453	47.06	0.440
7 (4, 2.807)	Thin Bootstrap	–	1	391/209	11	1.87	447.89	1.000
	Correction (trace-and-merge)	128	3	391/169	8	5.36	378.56	0.845
		256	6	391/163	7	10.0	389.73	0.870
		512	11	391/158	7	18.4	378.41	0.844
		1024	21	391/153	6	36.2	355.42	0.794
	Correction (BSGS)	128	3	391/181	8	6.47	335.76	0.750
		256	6	391/176	8	10.9	387.02	0.864
		512	11	391/173	8	19.5	390.12	0.871
		1024	21	391/171	7	33.5	428.53	0.957
	Naive (BSGS)	128	19	391/156	5	24.4	486.56	1.086
		256	37	392/152	5	46.0	488.22	1.090
		512	74	392/144	5	107	397.73	0.888
		1024	147	391/139	5	282	289.54	0.646
11 (3, 3.459)	Thin Bootstrap	–	1	390/144	7	2.62	109.87	1.000
	Correction (trace-and-merge)	128	2	390/100	5	4.76	84.00	0.765
		256	3	390/96	4	6.86	83.90	0.764
		512	5	390/94	4	11.1	84.98	0.773
		1024	9	390/89	4	20.3	78.94	0.718
	Correction (BSGS)	128	2	390/113	5	6.22	72.66	0.661
		256	3	390/109	5	9.28	70.41	0.641
		512	5	390/108	5	14.7	73.22	0.666
		1024	9	390/105	5	25.1	75.39	0.686
	Naive (BSGS)	128	12	390/86	3	19.2	107.76	0.981
		256	24	390/81	3	37.2	104.41	0.950
		512	47	390/77	3	73.3	98.71	0.898
		1024	94	390/71	3	156	85.34	0.777
13 (3, 3.700)	Thin Bootstrap	–	1	390/150	7	2.60	115.35	1.000
	Correction (trace-and-merge)	128	DNE	–	–	–	–	–
		256	2	390/98	4	5.16	76.05	0.659
		512	4	390/95	4	9.29	81.89	0.710
		1024	7	390/91	4	16.0	79.72	0.691
	Correction (BSGS)	128	DNE	–	–	–	–	–
		256	2	390/110	5	8.34	52.80	0.458
		512	4	390/110	5	13.7	64.13	0.556
		1024	7	390/107	5	22.3	67.30	0.583
	Naive (BSGS)	128	10	390/89	3	17.1	104.31	0.904
		256	20	390/82	3	32.6	100.61	0.872
		512	40	390/78	3	63.7	97.97	0.849
		1024	79	390/74	3	132	88.73	0.769

Table 5. Experimental results for $m = 32768$, $e \geq 3$, and varying p, w. RMFEs that are impossible to construct are denoted DNE (does not exist). There are no results for the naive style for $p = 3, w = 1024$ as the system ran out of memory during computation.