Field Instruction Multiple Data

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

[1] Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore
{mi_mi_aung, lim_enhui, benjamin_tan}@i2r.a-star.edu.sg
[2] School of Physical & Mathematical Sciences, Nanyang Technological University, Singapore
{junjie005@e.ntu.edu.sg,hxwang@ntu.edu.sg}

Abstract. Fully homomorphic encryption (FHE) has flourished since it was first constructed by Gentry (STOC 2009). Single instruction multiple data (SIMD) gave rise to efficient homomorphic operations on vectors in $(\mathbb{F}_{t^d})^\ell$, for prime t. RLWE instantiated with cyclotomic polynomials of the form $X^{2^N}+1$ dominate implementations of FHE due to highly efficient fast Fourier transformations. However, this choice yields very short SIMD plaintext vectors and high degree extension fields, e.g. $\ell < 100, d > 100$ for small primes ($t = 3, 5, \ldots$).

In this work, we describe a method to encode more data on top of SIMD, *Field Instruction Multiple Data*, applying reverse multiplication friendly embedding (RMFE) to FHE. With RMFE, length-k $\mathbb{F}_t$ vectors can be encoded into $\mathbb{F}_{t^d}$ and multiplied once. The results have to be recoded (decoded and then re-encoded) before further multiplications can be done. We introduce an FHE-specific technique to additionally evaluate arbitrary linear transformations on encoded vectors for free during the FHE recode operation. On top of that, we present two optimizations to unlock high degree extension fields with small t for homomorphic computation: r-fold RMFE, which allows products of up to 2^r encoded vectors before recoding, and a three-stage recode process for RMFEs obtained by composing two smaller RMFEs. Experiments were performed to evaluate the effectiveness of FIMD from various RMFEs compared to standard SIMD operations. Overall, we found that FIMD generally had $> 2\times$ better (amortized) multiplication times compared to FHE for the same amount of data, while using almost $k/2\times$ fewer ciphertexts required.

Keywords: Homomorphic Encryption · Finite Extension Fields · Reverse Multiplication Friendly Embeddings · Single Instruction Multiple Data.

⋆ Work was done while Sze Ling Yeo was at the Institute for Infocomm Research.

1 Introduction

Fully homomorphic encryption (FHE) has seen a lot of improvements since it was first realized by Gentry [24]. Currently, there are four main schemes in wide use, Brakerski-Gentry-Vaikunathan (BGV) [5], Brakerski-Fan-Vercauteren (BFV) [4,22], Cheon-Kim-Kim-Song (CKKS) [11] and FHEW/TFHE [14,15,21]. The first two schemes support finite field operations, the CKKS scheme supports approximate arithmetic of numbers and the FHEW/TFHE family operates over bits or low-precision numbers.

Most implementations of FHE, such as SEAL [36], PALISADE [35] and Lattigo [33] focus on the case where BGV and BFV are instantiated with power-of-two cyclotomic polynomial moduli. These parameters enjoy highly efficient arithmetic due to negacyclic fast fourier transforms (FFT). However, small primes such as $t = 3, 5, 7, 11$ are not useful in these cases due to the unfriendly decomposition of the plaintext space into very few slots with a slot algebra corresponding to finite fields of high extension degree. Therefore, use of BGV and BFV focused on word-sized homomorphic encryption which uses large primes of 32 or more bits, integer and fractional encodings proposed in [8, 16, 19, 20] and an alternate "polynomial modulus" for high-precision integer arithmetic in [3, 10].

Various papers have been published on the use of finite extension fields of low to medium extension degree (≤ 64) for homomorphic computation, which we elaborate on in a later section on related work. However, there remains a lack of techniques that unlock the use of high degree extension fields for homomorphic computation, which would lead to improvements for small-prime arithmetic circuits due to the faster arithmetic enabled by negacyclic FFT.

Our Contributions. In this work, we introduce field instruction multiple data (FIMD), a method to encode more data into FHE ciphertexts by leveraging the inherent *vector of extension fields* plaintext structure from SIMD. We add another level of packing to FHE, embedding a vector of base/intermediate field elements into each slot of a SIMD plaintext such that homomorphic operations can be performed on the encoded vectors. Field addition, multiplication and linearized polynomial evaluations correspond to component-wise addition, multiplication, and linear maps on the encrypted vectors.

To that end, we apply reverse multiplication friendly embedding (RMFE) defined by Cascudo et al. [7] to FHE. RMFE allows us to encode a length-k vector of small field elements $(\mathbb{F})^k$ into a single element of a larger extension field $\mathbb{E}/\mathbb{F}$. Products of these extension field elements then "correspond" to component-wise multiplication on the underlying vectors. However, this process is not a homomorphism and thus cannot support an arbitrary number of multiplications. To address this limitation in MPC, Cascudo et al. [7] defined a recoding protocol `ReEncode` which decodes and re-encodes field elements in one go after each MPC multiplication so that it can be used in subsequent multiplications.

The key to applying RMFEs to FHE is that the encode, decode, and recoding (i.e. decode then re-encode) operations are $\mathbb{F}$-linear maps between $(\mathbb{F})^k$ and $\mathbb{E}$. Such maps can be represented by linearized polynomials and therefore

evaluated in FHE using low-noise Frobenius automorphisms. Furthermore, we show that rotations, shifts and even arbitrary linear transformations M on the encrypted vector can be done for free by modifying the recode operation into a composition of decode, M, and encode. This gives FIMD more flexibility than SIMD in terms of the overhead of linear transformations over the plaintext space but FIMD requires the recode operation after multiplications. Crucially, we exploit the fact that the recode operation in FHE is non-interactive and does not require any pre-processing. Performing arbitrary linear transformations during the `ReEncode` protocol of Cascudo et al. [7] would be almost impossible because randomness specific to any desired linear transformation has to be prepared beforehand.

Besides that, we propose r-fold RMFE to amortize the overhead of the homomorphic RMFE recode operation over several multiplications, at the cost of lower packing efficiency. Instead of decoding after a single multiplication, r-fold RMFE allows up to encodings of 2^r vectors to be multiplied together before decoding. With an additional requirement that field elements encoding multiplications of fewer than 2^r vectors can be added together, r-fold RMFE can allow multivariate polynomials of degree up to 2^r to be evaluated before decoding. This generalization of RMFE could be of independent interest.

On top of that, we introduce a three-stage process for recoding operations for RMFEs composed of two component RMFEs. Exploiting the fact that such RMFEs is built on a tower of field extensions $\mathbb{F} \subset \mathbb{E}_1 \subset \mathbb{E}_2$, we apply three linear maps $\psi_{\mathsf{out}} : \mathbb{E}_2 \to (\mathbb{E}_1)^{k_{\mathsf{out}}}$, $\phi_{\mathsf{out}} : (\mathbb{E}_1)^{k_{\mathsf{out}}} \to \mathbb{E}_2$, and $\pi'_{\mathsf{in}} : (\mathbb{E}_1)^{k_{\mathsf{out}}} \to (\mathbb{E}_1)^{k_{\mathsf{out}}}$, each of which have lower degree than the recoding map $\pi : \mathbb{E}_2 \to \mathbb{E}_2$. All together, this approach reduces the number of Frobenius automorphisms needed compared to the standard recode process.

Finally, we perform several experiments to compare the efficiency of the various flavors of RMFE introduced in this work against each other and standard FHE multiplications. FIMD improves the performance of FHE for small plaintext moduli, not only achieving more than $2\times$ faster multiplications amortized but also using up to $k/2\times$ fewer ciphertexts in the whole process.

Related Work. Exploiting finite fields for homomorphic computation was first considered by Kim et al. [32]. They showed that equality of two encrypted integers could be efficiently computed using Fermat's Little Theorem. For more complex operations, Jäschke and Armknecht [31] explored using addition and multiplication in extension fields to compute integer addition but found them lacking. Leveraging the vector space nature of extension fields, Tan et al. [39] proposed the extract then compute method for comparison of encrypted integers. Illiashenko and Zucca [30] took advantage of the nature of comparison polynomials, reaching comparable efficiency to THFE-based methods for homomorphic comparisons.

Studies were also done for encoding integers and fixed-point numbers such that arithmetic was efficient. Dowlin et al. [19] considered decomposing integers and fractional numbers into base-2 representations and then encoding them as polynomials for fast arithmetic. Costache et al. [16] then showed that the two

methods of Dowlin et al. were isomorphic and derived lower bounds for the representation to support homomorphic computation. This was further extended by Castryck et al. [8] to a more flexible encoding based on Laurent polynomials and more fine-grained decomposition of the FHE plaintext space with composite plaintext modulus. In another direction, Chen et al. [10] proposed to replace the plaintext modulus t with $X - b$ for some base b. This yields the plaintext space $\mathbb{Z}/(b^n + 1)\mathbb{Z}$ which enables high-precision arithmetic. Bootland et al. [3] generalized this to support complex-valued data by considering polynomials of the form $X^m + b$.

Lastly, RMFE was studied to improve MPC over finite fields. Cascudo et al. [7] used it to improve the amortized communication complexity of MPC protocols at the expense of the size of the field; previous work had to reduce the adversary threshold instead. Concurrently, Block et al. [2] applied it to achieve more efficient batched multiplications in MPC over binary fields. Since then, RMFE has been generalized to support MPC over Galois rings by Cramer et al. [17] and alternatively into circuit-amortization friendly embeddings to evaluate more complex circuits in a single multiplication by Dalskov et al. [18]. For HE-based MPC over $\mathbb{Z}_{2^k}$, methods were devised for protocols Overdrive2k [34] and MHz2k [12] for packing $\mathbb{Z}_{2^k}$-messages into polynomials, supporting depth-1 homomorphic correspondence.

2 Preliminaries

2.1 Fully Homomorphic Encryption

A leveled fully homomorphic encryption (FHE) scheme is a homomorphic encryption scheme that supports evaluation of circuits of at most depth L, for some pre-defined non-negative integer L. Encryptions of messages m will be denoted with $\overline{m}$ to emphasize their underlying encrypted messages. We will use $\mathcal{P}$ to denote the space of possible messages for the FHE scheme.

- $(pk, evk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda, 1^L)$: Given security and level parameters λ and L as inputs, output a public key pk, secret key sk and evaluation key evk.
- $c = \overline{m} \leftarrow \mathsf{Enc}(pk, m)$: Given a public key pk and message $m \in \mathcal{P}$ as inputs, output a ciphertext $c = \overline{m}$ that encrypts m.
- $m' \leftarrow \mathsf{Dec}(sk, c)$: Given a secret key sk and ciphertext c as inputs, and outputs a message $m' \in \mathcal{P}$.
- $c' \leftarrow \mathsf{Eval}(evk, f, \overline{m_1}, ..., \overline{m_n})$: Given evaluation key evk, function $f : \mathcal{P}^n \to \mathcal{P}$ and encryption $\overline{m_1}, ..., \overline{m_n}$ of ciphertexts $m_1, ..., m_n \in \mathcal{P}$, output a ciphertext c' such that $\mathsf{Dec}(sk, c') = f(m_1, ..., m_n)$.

Usually, the Eval algorithm uses sub-routines of which the most common ones are homomorphic addition and multiplication.

- $c^+ \leftarrow \mathsf{EvalAdd}(evk, \overline{m_1}, \overline{m_2})$: Given an evaluation key evk and two ciphertexts $\overline{m_1}, \overline{m_2}$ as inputs, output a ciphertext $c^+ = \overline{m_1 + m_2}$, encrypting the sum of the encrypted input messages.

– $c^{\times} \leftarrow \mathsf{EvalMul}(evk, \overline{m_1}, \overline{m_2})$: Given an evaluation key evk and two ciphertexts $\overline{m_1}, \overline{m_2}$ as inputs, output a ciphertext $c^{\times} = \overline{m_1 \times m_2}$, encrypting the product of the encrypted input messages.

For all known (leveled) FHE schemes, which are based on (Ring) Learning with Error ((R)LWE) problems, ciphertexts are noisy encryptions of their underlying plaintext. This means that only a limited number of computations can be performed before the noise in the ciphertexts overwhelms the data and the result is unusable.

Single Instruction Multiple Data (SIMD). Let $R = \mathbb{Z}[X]/\Phi_m(X)$, where $\Phi_m(X)$ is the m-th cyclotomic polynomial and $R_q = R/qR$. Generation two FHE schemes such as BGV and BFV typically use plaintext spaces of the form $\mathcal{P} = R_t$ for some prime t. Smart and Vercauteren [37] noted that, if t and m are co-prime, $\Phi_m(X) \equiv \prod_{i=1}^{\ell} f_i(X) \bmod t$, with $\deg f_i(X) := d = \phi(m)/\ell$ for all $i \in \{1, \ldots, \ell\}$. They proposed single instruction multiple data (SIMD), simultaneously operating on vectors of messages, for FHE by exploiting the following ring isomorphisms

$$\mathcal{P} = R_t \cong \mathbb{Z}_t[X]/\langle f_1(X)\rangle \times \mathbb{Z}_t[X]/\langle f_2(X)\rangle \times \cdots \times \mathbb{Z}_t[X]/\langle f_\ell(X)\rangle \cong \prod_{i=1}^{\ell} \mathbb{F}_{t^d}.$$

These isomorphisms are a result of applying the Chinese Remainder Theorem on the polynomial ring $\mathbb{Z}[X]/\Phi_m(X)$ with the decomposition of $\Phi_m(X)$ into its irreducible factors modulo t. From this, vectors in $(\mathbb{F}_{t^d})^{\ell}$ can be encoded in a single ciphertext and enjoy homomorphic component-wise $\mathbb{F}_{t^d}$ addition and multiplications. Furthermore, the elements within the vectors can be moved around via ring automorphisms $\kappa : X \mapsto X^{\kappa}$, for $\kappa \in \mathbb{Z}_m^*$. In particular, for prime t, the automorphism $X \mapsto X^t$ corresponds to a component-wise Frobenius map on plaintext vectors. Halevi and Shoup [29] present a thorough introduction on the SIMD plaintext structure. Thus, with SIMD, we have a third sub-routine for intra-vector data manipulation through homomorphic automorphisms.

– $c^{\star} \leftarrow \mathsf{EvalAut}(evk, \kappa, \overline{m})$: Given an evaluation key evk, ciphertext $\overline{m}$ and automorphism $\kappa \in \mathbb{Z}_m^*$, output a ciphertext $c^{\star} = \overline{\kappa(m)}$.

This third sub-routine, more specifically the Frobenius automorphism, is key to the effective application of finite extension fields for homomorphic computation. In practice, multiple powers of the Frobenius automorphism, typically the set of $\{t^i\}_{i=1}^{d-1}$ will be needed and techniques have been developed to optimize the computational complexity and evaluation key sizes for evaluating more than one automorphism on a single ciphertext [28]. Besides that, through $\mathsf{EvalAut}$ and multiplicative masks, basic data movement of shifts and rotations on encoded vectors $\mathbf{x} = (x_1, ..., x_\ell) \in (\mathbb{F}_{t^d})^{\ell}$.

– $c' \leftarrow \mathsf{FHE.EvalShift}(evk, \rho, \overline{\mathbf{x}})$: Let $\rho < 0$ denote a left shift and $\rho \geq 0$ denote a right shift. Using $\mathsf{FHE.EvalAut}$ defined above, output the ciphertext

$$c' = \begin{cases} \overline{(x_{|\rho|}, ..., x_\ell, 0, ..., 0)}, & \text{if } \rho < 0; \\ \overline{(0, ..., 0, x_1, ..., x_{\ell-\rho})}, & \text{otherwise.} \end{cases}$$

– $c' \leftarrow \mathsf{FHE.EvalRot}(evk, \rho, \overline{\hat{\mathbf{x}}})$: Let $\rho < 0$ denote a left rotation and $\rho \geq 0$ denote a right rotation. Using $\mathsf{FHE.EvalAut}$ defined above, output the ciphertext

$$c' = \begin{cases} \overline{(x_{|\rho|}, ..., x_\ell, x_1, ..., x_{|\rho|-1})}, & \text{if } \rho < 0; \\ \overline{(x_{\ell-\rho+1}, ..., x_\ell, x_1, ..., x_{\ell-\rho})}, & \text{otherwise.} \end{cases}$$

2.2 Finite Extension Fields

Let q be a prime power. Extension fields $\mathbb{F}_{q^w}$ are $\mathbb{F}_q$-vector spaces of dimension w. A *q-linearized polynomial* $f(Y)$ is a polynomial of the form

$$f(Y) = f_0 + f_1 Y^{q^1} + f_2 Y^{q^2} + \cdots + f_{z-1} Y^{q^{z-1}} \in \mathbb{F}_q[Y],$$

where any non-zero coefficient of f is attached to a monomial Y^{q^a} for some positive integer a. In the following lemma, we review how $\mathbb{F}_q$-linear maps between subspaces of $\mathbb{F}_{q^w}$ can be expressed as q-linearized polynomials.

Lemma 1. *Let V and W be $\mathbb{F}_q$-linear subspaces of $\mathbb{F}_{q^w}$, and let $T : V \to W$ be an $\mathbb{F}_q$-linear map. Then, there exists a unique q-linearized polynomial $f_T(Y)$ with $\deg f_T \leq q^{\dim(V)}$, such that for any $\alpha \in V$, $f_T(\alpha) = T(\alpha)$.*

Proof. Let $\{\alpha_1, \alpha_2, \ldots, \alpha_k\}$ be a basis for V, and let A denote the Moore matrix given by

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_k \\ \alpha_1^q & \alpha_2^q & \cdots & \alpha_k^q \\ \vdots & \vdots & \cdots & \vdots \\ \alpha_1^{q^{k-1}} & \alpha_2^{q^{k-1}} & \cdots & \alpha_k^{q^{k-1}} \end{bmatrix}.$$

Evaluation of a q-linearized polynomial is clearly $\mathbb{F}_q$-linear, so for $f_T(\alpha) = T(\alpha)$ to hold for any α, $f_T(Y)$ must have coefficients $f_0, f_1, \ldots, f_{k-1}$ such that the following matrix equation holds:

$$\begin{bmatrix} f_0 & f_1 & \cdots & f_{k-1} \end{bmatrix} A = \begin{bmatrix} T(\alpha_1) & T(\alpha_2) & \cdots & T(\alpha_k) \end{bmatrix}.$$

Since $\alpha_1, \alpha_2, \ldots, \alpha_k$ are linearly independent, A has a nonzero determinant by [26, Lemma 1.3.3]. A is also a square matrix, so its inverse A^{-1} always exists. Thus $f_0, f_1, \cdots, f_{k-1}$ is found by computing $\begin{bmatrix} T(\alpha_1) & T(\alpha_2) & \cdots & T(\alpha_k) \end{bmatrix} A^{-1}$. From this computation it is also clear that $f_T(Y)$ is unique. The proof is complete.

For FHE schemes with plaintext space $\mathcal{P} \cong (\mathbb{F}_{t^d})^\ell$, homomorphic evaluation of component-wise t-linearized polynomials can be easily done using $\mathsf{EvalAut}$. Each monomial Y^{t^i} can be homomorphically computed with $\mathsf{EvalAut}(evk, t^i, \overline{Y})$, without multiplications and therefore almost no depth.

– $c' \leftarrow \mathsf{EvalLinearMap}(evk, f_T, \overline{m})$: For simplicity, we use $\overline{m_1} + \overline{m_2}$ to denote $\mathsf{EvalAdd}(evk, \overline{m_1}, \overline{m_2})$ for FHE ciphertexts $\overline{m_1}, \overline{m_2}$ and $a \cdot \overline{m}$ to mean the product of a plaintext a and ciphertext $\overline{m}$. Then, let $f_T(X) = \sum_{i=0}^{d-1} a_i X^{t^i}$ and output $c' := \sum_{i=0}^{d-1} a_i \cdot \mathsf{EvalAut}(evk, p^i, \overline{m}) = \overline{\sum_{i=0}^{d-1} a_i m^{t^i}}$.

2.3 Reverse Multiplication Friendly Embeddings

Introduced by Cascudo et al. [7] and concurrently studied by Block et al. [2], reverse multiplication friendly embeddings (RMFE) are methods of embedding $(\mathbb{F}_t)^k$ into $\mathbb{F}_{t^d}$ such that component-wise multiplication in the vector space (denoted with $*$) corresponds to multiplication over the field.

Definition 1. *Let q be a prime power and $\mathbb{F}_q$ denote the finite field of q elements. For integers $k, w \geq 1$, a $(k, w)_q$-RMFE is a pair of $\mathbb{F}_q$-linear maps, (ϕ, ψ), where $\phi : (\mathbb{F}_q)^k \to \mathbb{F}_{q^w}$ and $\psi : \mathbb{F}_{q^w} \to (\mathbb{F}_q)^k$ such that for all $\mathbf{x}, \mathbf{y} \in (\mathbb{F}_q)^k$,*

$$\mathbf{x} * \mathbf{y} = \psi(\phi(\mathbf{x}) \cdot \phi(\mathbf{y})).$$

There are two approaches to constructing RMFEs, polynomial interpolation, and algebraic function fields. From these, others can be obtained by composing these base constructions appropriately.

RMFEs from Polynomial Interpolation. The core idea is that a vector $\mathbf{x} = (x_1, ..., x_k) \in (\mathbb{F}_q)^k$ can be encoded as a polynomial $f \in \mathbb{F}_q[X]$ via interpolation, that is, we require that $f(P_i) = x_i$ for some fixed set of points $\{P_i \in \mathbb{F}_q\}$. Hence products of polynomials, when evaluated at the points $\{P_i\}$, yield component-wise products of the vectors corresponding to each polynomial. The following theorem constructs an RMFE that can then be constructed based on this principle. The caveat is the value of k is limited by the number of points available in $\mathbb{F}_q$.

Theorem 1 ([7, Lemma 4]). *For a base finite field $\mathbb{F}_q$ and $1 \leq k \leq q + 1$, there exists a $(k, 2k - 1)_q$-RMFE.*

Proof. Let $\mathbb{F}_q[X]_{\leq m}$ denote the set of polynomials in $\mathbb{F}_q[X]$ whose degree is at most m and define ∞_{m+1} as a formal symbol such that $f(\infty_{m+1})$ is the coefficient of X^m for $f \in \mathbb{F}_q[X]_{\leq m}$. Let $P_1, ..., P_k$ be pair-wise distinct elements in $\mathbb{F}_q \cup \{\infty_k\}$ and let α be a root of a monic irreducible polynomial $F(X)$ of degree $2k - 1$. Then $\mathbb{F}_{q^{2k-1}} \cong \mathbb{F}_q(\alpha) \cong \mathbb{F}_q[X]/(F(X))$.

Polynomial interpolation yields the following $\mathbb{F}_q$-vector space isomorphism between $\mathbb{F}_q[X]_{\leq k-1}$ and $(\mathbb{F}_q)^k$:

$$\mathcal{E}_1 : \mathbb{F}_q[X]_{\leq k-1} \to (\mathbb{F}_q)^k; \quad f \mapsto (f(P_1), ..., f(P_k)).$$

The evaluation embedding into $(\mathbb{F}_q)^k$ can be extended naturally to any set of polynomials of a limited degree. In particular, for polynomials in $\mathbb{F}_q[X]_{\leq 2k-2}$,

$$\mathcal{E}_1' : \mathbb{F}_q[X]_{\leq 2k-2} \to (\mathbb{F}_q)^k; \quad f \mapsto (f(P_1'), ..., f(P_k')),$$

where $P_i' := P_i$ if $P_i \in \mathbb{F}_q$ and $P_i' := \infty_{2k-1}$ if $P_i = \infty_k$. Finally, we use the following isomorphism to map polynomials to the extension field $\mathbb{F}_{q^{2k-1}}$,

$$\mathcal{E}_2 : \mathbb{F}_q[X]_{\leq 2k-2} \to \mathbb{F}_{q^{2k-1}}; \quad f = \sum_{i=0}^{2k-2} f_i X^i \mapsto f(\alpha) = \sum_{i=0}^{2k-2} f_i \alpha^i.$$

The $(k, 2k-1)_q$-RMFE is obtained by defining $\phi = \mathcal{E}_2 \circ \mathcal{E}_1^{-1}$, where $\mathcal{E}_2$ is restricted to the subset $\mathbb{F}_q[X]_{\leq k-1}$, and $\psi = \mathcal{E}_1' \circ \mathcal{E}_2^{-1}$. To see the correctness of the procedure, let $f_x = \mathcal{E}_1^{-1}(\mathbf{x}), f_y = \mathcal{E}_1^{-1}(\mathbf{y})$ be the polynomial encoding of the vectors $\mathbf{x}, \mathbf{y} \in (\mathbb{F}_q)^k$. $\mathcal{E}_2^{-1}(\phi(\mathbf{x}) \cdot \phi(\mathbf{y})) = f_{\mathbf{x}} f_{\mathbf{y}}$, since there is no overflow on the monomials X^i. Therefore,

$$\begin{aligned}\psi(\phi(\mathbf{x}) \cdot \phi(\mathbf{y})) &= \mathcal{E}_1'(f_{\mathbf{x}} f_{\mathbf{y}}) \\ &= (f_{\mathbf{x}} f_{\mathbf{y}}(P_1), ..., f_{\mathbf{x}} f_{\mathbf{y}}(P_k)) \\ &= (f_{\mathbf{x}}(P_1) f_{\mathbf{y}}(P_1), ..., f_{\mathbf{x}}(P_k) f_{\mathbf{y}}(P_k)) \\ &= \mathbf{x} * \mathbf{y}.\end{aligned}$$

Algebraic Function Fields. A function field $K/\mathbb{F}_q$ is an algebraic extension of the rational function field $\mathbb{F}_q(X)$, which contains all fractions of polynomials in $\mathbb{F}_q[X]$. Every function field K has an infinite set of "points" called *places*, denoted by P and has a degree $\deg P$. The number of places with a given degree is finite and in particular, places P of $\deg P = 1$ are called *rational*.

For functions $f \in K$ and a place P, either f has a pole at P (i.e. $(1/f)(P) = 0$), or f can be evaluated at P and $f(P)$ can be thought of as an element of $\mathbb{F}_{q^{\deg P}}$. The elements of the function field K always have the same number of zeroes and poles, up to multiplicity, called the order. For any two functions f, g that do not have poles at P,

1. $\lambda(f(P)) = (\lambda f)(P)$, for every $\lambda \in \mathbb{F}_q$;
2. $f(P) + g(P) = (f + g)(P)$ and
3. $f(P) \cdot g(P) = (f \cdot g)(P)$.

A *divisor* is a formal sum of places, $G = \sum c_P P$, with $c_P \in \mathbb{Z}$ and only finitely many $c_P \neq 0$. This set of places also is called the support of G and denoted with $supp(G)$. Just like places, a divisor G has a degree, $\deg G := \sum c_P \deg P \in \mathbb{Z}$. For any function $f \in K \backslash \{0\}$, there is a *principal divisor* associated to f, denoted with (f). Roughly speaking, this principal divisor has the form $(f) = \sum a_P P$, where $a_P = o$ if f has a zero of order o at P, $a_P = -o$ if f has a pole of order o at P and $a_P = 0$ if P is neither a zero or pole of f.

The Riemann-Roch space associated with a divisor, $G = \sum c_P P$, is denoted by $\mathcal{L}(G) = \{0\} \cup \{f \in K \backslash \{0\} \mid (f) + G = \sum a_P P \text{ and } a_P \geq 0, \forall P\}$. It is the set of all functions in K that have poles and zeroes at the set of places prescribed by G along with the zero function. To be more precise, every function $f \in \mathcal{L}(G)$ has a zero of order at least $|c_P|$ at the places P if $c_P \leq 0$ and can have a pole of order of at most c_P at the places P with $c_P \geq 0$. For any other place $Q \notin supp(G)$, $f(Q) \in \mathbb{F}_{q^{\deg Q}}$. This space is a vector space over $\mathbb{F}_q$ and its dimension $\ell(G)$ is not more than $\deg G + 1$ [6, Lemma 2.51].

Another important fact is that given $f, g \in \mathcal{L}(G)$, the product $f \cdot g$ resides in $\mathcal{L}(2G)$. For every function field K, there is a a non-negative integer associated with it called the genus, denoted with $g(K) := \max_G \deg G - \ell(G) + 1$ where G runs over all divisors of K.

From Polynomial Interpolation to Function Fields. To give more intuition for the more abstract RMFEs from algebraic function fields, we sketch how

RMFEs from the rational function field $\mathbb{F}_q(Y)$ parallel the RMFEs obtained from polynomial interpolation. The points $P_i \in \mathbb{F}_q$ of polynomial interpolation can be understood as univariate polynomials $(Y - P_i)$, which are *rational* places in $\mathbb{F}_q(Y)$, and ∞ roughly corresponding to the place $(1/Y)$. More general places in $\mathbb{F}_q(Y)$ include the ideals $Q = (f(Y))$, where f are irreducible polynomials. The degree of such a place Q is equal to the degree of the polynomial $f(Y)$.

Recall that in interpolation RMFEs, vectors were mapped to polynomials in $\mathbb{F}_q[X]_{\leq 2k-2}$. In rational function field RMFEs, this corresponds to mapping vectors to functions living in a particular subset of the Riemann-Roch space $\mathcal{L}(G)$ of some divisor G, such that G does not have the places $\{(Y - P_i)\}$ in its support. To embed functions from $\mathcal{L}(G)$ into $\mathbb{F}_{q^w}$, we "evaluate" them at a fixed place $R = (f(Y))$ whose degree is w. This evaluation corresponds to considering the residues of our functions, modulo $f(Y)$.

RMFEs from Algebraic Function Fields. Here, we state the properties of RMFEs that can be obtained from algebraic function fields. Proofs for the following theorems and corollaries can be found in [7].

Theorem 2 ([7, Lemma 6]). *Let $K/\mathbb{F}_q$ be an algebraic function field with genus g and k distinct rational places $P_1, ..., P_k$. Let G be a divisor of K such that $supp(G) \cap \{P_1, ..., P_k\} = \emptyset$ and $\ell(G) - \ell(G - \sum_{i=1}^{k} P_i) = k$. If there exists a place R with $w = \deg R > 2 \deg G$, then there exists a $(k, w)_q$-RMFE.*

In particular, the conditions of Theorem 2 are satisfied as long as there is a place of sufficiently high degree.

Corollary 1 ([7, Corollary 1]). *Let $K/\mathbb{F}_q$ be an algebraic function field of genus g and suppose that there are k distinct rational places $(P_1, ..., P_k)$ and a place of degree $w \geq 2k + 4g - 1$. Then, there exists a $(k, w)_q$-RMFE.*

Finally, we state how RMFEs can be composed to yield more RMFEs.

Theorem 3 ([7, Lemma 5]). *Suppose $(\phi_{\mathsf{in}}, \psi_{\mathsf{in}})$ is a $(k_{\mathsf{in}}, w_{\mathsf{in}})_q$-RMFE and $(\phi_{\mathsf{out}}, \psi_{\mathsf{out}})$ is a $(k_{\mathsf{out}}, w_{\mathsf{out}})_{q^{w_{\mathsf{in}}}}$-RMFE. (ϕ, ψ) is a $(k_{\mathsf{in}}k_{\mathsf{out}}, w_{\mathsf{in}}w_{\mathsf{out}})_q$-RMFE, where*

$$\phi : (\mathbb{F}_q)^{k_{\mathsf{in}}k_{\mathsf{out}}} \to \mathbb{F}_{q^{w_{\mathsf{in}}w_{\mathsf{out}}}}$$

$$\begin{pmatrix} x_1, & ..., & x_{k_{\mathsf{in}}}, \\ x_{k_{\mathsf{in}}+1}, & ..., & x_{2k_{\mathsf{in}}}, \\ \vdots & \ddots & \vdots \\ x_{k_{\mathsf{in}}(k_{\mathsf{out}}-1)+1}, & ..., & x_{k_{\mathsf{in}}k_{\mathsf{out}}} \end{pmatrix} \mapsto \phi_{\mathsf{out}} \begin{pmatrix} \phi_{\mathsf{in}}(x_1, ..., x_{k_{\mathsf{in}}}), \\ \phi_{\mathsf{in}}(x_{k_{\mathsf{in}}+1}, ..., x_{2k_{\mathsf{in}}}), \\ \cdots, \\ \phi_{\mathsf{in}}(x_{k_{\mathsf{in}}(k_{\mathsf{out}}-1)+1}, ..., x_{k_{\mathsf{in}}k_{\mathsf{out}}}) \end{pmatrix}$$

and

$$\psi : \mathbb{F}_{q^{w_{\mathsf{in}}w_{\mathsf{out}}}} \to (\mathbb{F}_q)^{k_{\mathsf{in}}k_{\mathsf{out}}}$$

$$\alpha \mapsto \psi_{\mathsf{out}}(\alpha) = (u_1, ..., u_{k_{\mathsf{out}}}) \in (\mathbb{F}_{q^{w_{\mathsf{in}}}})^{k_{\mathsf{out}}} \mapsto (\psi_{\mathsf{in}}(u_1), ..., \psi_{\mathsf{in}}(u_{k_{\mathsf{out}}})) \in (\mathbb{F}_q)^{k_{\mathsf{in}}k_{\mathsf{out}}}.$$

3 Field Instruction Multiple Data (FIMD)

In this section, we present how RMFE and SIMD can be combined to encode and work on more data in a single FHE ciphertext. Then, we describe an extension to RMFE that removes the need to recode after each multiplication, at the expense of more expensive recoding operations. Finally, we introduce some optimizations tailored to composite RMFEs.

3.1 RMFE with FHE

The BGV and BFV FHE schemes offer the plaintext space $\mathcal{P} \cong \prod_{i=1}^{\ell} \mathbb{F}_{t^d}$. Typical FHE-based secure computation systems only use base field operations and pack an $\mathbb{F}_t$ element in each slot. RMFE unlocks the full capacity of $\mathcal{P}$ by introducing a "new" dimension in $\mathcal{P}$ hidden within the $\mathbb{F}_{t^d}$ algebra of each SIMD slot. Homomorphic extension field operations such as addition, multiplication, and linearized polynomials are exploited to work with encrypted vectors from $(\mathbb{F}_q)^k$ within each plaintext slot, where $q^w \leq t^d$. With RMFE, we use "extension field instructions" to process data, yielding a *field instruction multiple data* (FIMD) system.

Throughout this section, let (ϕ, ψ) be a $(k, w)_q$-RMFE. We first describe the core encoding and decoding functionality of FIMD.

- $\mu \in \mathbb{F}_{t^d} \leftarrow \mathsf{FIMD.Encode}(\mathbf{x} = (x_1, ..., x_{\ell \cdot k}) \in (\mathbb{F}_q)^{\ell \cdot k})$: For $i = 1, ..., \ell$, let $\mathbf{x}_i = (x_{(i-1)k+1}, ..., x_{i \cdot k}) \in (\mathbb{F}_q)^k$.
  1. Embed each $\mathbf{x}_i$ into $\mathbb{F}_{q^w}$ with ϕ to obtain $\hat{\mathbf{x}} = (\phi(\mathbf{x}_1), ..., \phi(\mathbf{x}_\ell)) \in (\mathbb{F}_{q^w})^\ell$;
  2. Encode $\hat{\mathbf{x}}$ into $\mu \in \mathcal{P}$ with the SIMD isomorphism.
- $\mathbf{m} \in (\mathbb{F}_q)^{\ell \cdot k} \leftarrow \mathsf{FIMD.Decode}(\mu \in \mathcal{P})$:
  1. Decode μ to the SIMD plaintext vector $\hat{\mathbf{m}} = (\mu_1, ..., \mu_\ell) \in (\mathbb{F}_{q^w})^\ell$;
  2. Apply ψ to the components of $\hat{\mathbf{m}}$ separately to compute the final output $\mathbf{m} = (\psi(\mu_1), ..., \psi(\mu_\ell)) \in (\mathbb{F}_q)^{\ell \cdot k}$.

The μ from $\mathsf{FIMD.Encode}$ is then encrypted with the FHE scheme into $\overline{\mu}$ for use in encrypted processing. Similarly, the input to $\mathsf{FIMD.Decode}$ comes from decrypted FHE ciphertexts that contain RMFE-encoded vectors.

Arithmetic Operations. The main operations in FHE are homomorphic addition and multiplication. Addition is straightforward in FIMD but multiplication requires a little more work to achieve. Because RMFE only supports one multiplication after embedding, the resulting data cannot be used without first decoding and re-encoding it. A re-encoding protocol was proposed by Cascudo et al. [7] to refresh secret-shared RMFE-encoded field elements and we require a similar operation with FHE. Crucially, the $\mathbb{F}_q$-linear nature of ϕ and ψ means that they can be composed to obtain a recode map, $\pi := \phi \circ \psi$.

Evaluating π on encrypted RMFE-encoded data is done by homomorphically evaluating the q-linearized polynomial f_π from applying Lemma 1 to π. Let evk denote the evaluation keys for the BGV/BFV FHE scheme and μ_1, μ_2 obtained from $\mathsf{FIMD.Encode}$, then the basic homomorphic FIMD operations are as follows.

- $c' \leftarrow \mathsf{FIMD.Recode}(evk, c)$: Output $c' = \mathsf{FHE.EvalLinearMap}(evk, f_\pi, c)$.
- $c^+ \leftarrow \mathsf{FIMD.EvalAdd}(evk, \overline{\mu_1}, \overline{\mu_2})$: Output $c^+ = \mathsf{FHE.EvalAdd}(evk, \overline{\mu_1}, \overline{\mu_2})$.
- $c^\times \leftarrow \mathsf{FIMD.EvalMul}(evk, \overline{\mu_1}, \overline{\mu_2})$:
  1. Compute $c = \mathsf{FHE.EvalMul}(evk, \overline{\mu_1}, \overline{\mu_2})$;
  2. Output $c^\times = \mathsf{FIMD.Recode}(evk, c)$.

Moving Data within Encrypted RMFE-Encoded Vectors. With SIMD, data in the various slots can be moved around using the automorphisms $\kappa \in \mathbb{Z}_m^*$. Rotations of the components are achieved with $\mathsf{EvalAut}$ using the appropriate automorphisms, κ, and shifts computed by first masking the irrelevant slots and rotating the result. Similar operations can be done with RMFE-encoded vectors and in fact, RMFE supports even more complex intra-vector manipulations.

This is possible due to the $\mathbb{F}_q$-linearity of ϕ and ψ. Any $\mathbb{F}_q$-linear map $\tau : (\mathbb{F}_q)^k \to (\mathbb{F}_q)^k$ can be applied on $\mathbf{x}$ of $\phi(\mathbf{x})$ by sandwiching it between ϕ and ψ, as $\phi \circ \tau \circ \psi$. This generalizes the recode operation, which has the identity map id between ϕ and ψ. In this way, arbitrary linear transformations on individual RMFE components in FIMD can be folded into homomorphic multiplication and thus done for free in many situations. Let f'_τ be the polynomial from Lemma 1 for $\phi \circ \tau \circ \psi$.

- $c^{\times\prime} \leftarrow \mathsf{FIMD.EvalMul'}(evk, \tau, \overline{m_1}, \overline{m_2})$:
  1. Compute $c = \mathsf{FHE.EvalMul}(evk, \overline{\mu_1}, \overline{\mu_2})$;
  2. Output $c^\times = \mathsf{FHE.EvalLinearMap}(evk, f'_\tau, c)$.

Rotations and Shifts for FIMD-Encoded Vectors. With the complete FIMD technique, where RMFE and SIMD are combined, we focus on how to compute rotations and shifts on $\mathcal{P}$, interpreted as the space of vectors $(\mathbb{F}_q)^{k\cdot\ell}$. For any plaintext $\mathbf{x} = (x_1, ..., x_{k\cdot\ell}) \in (\mathbb{F}_q)^{k\cdot\ell}$, FIMD encodes it into $\hat{\mathbf{x}} = (\phi(\mathbf{x}_1), ..., \phi(\mathbf{x}_\ell))$, where $\mathbf{x}_i = (x_{(i-1)k+1}, ..., x_{i\cdot k})$ for $1 \le i \le \ell$. To achieve rotations or shifts on the entire vector, we split the process into two steps.

First, we execute a set of RMFE-only data movement operations followed by a set of SIMD-only data movement operations. If we only move by small steps ($< k$), usually there would be one portion of the data that will move to an adjacent SIMD-slot and another that stays in the same SIMD-slot. More generally, each $\mathbf{x}_i$ can be partitioned into two parts, one that moves by z SIMD-slots and the other that moves by $z+1$ SIMD-slots to the left or right, for some $z = 0, ..., \ell - 1$. If any of $z, z+1$ goes beyond $\ell + 1$, those components should be wrapped around the other side of the vector for rotations and discarded for shifts. This is accomplished by using the appropriate SIMD data movement operation, $\mathsf{FHE.EvalRot}$ for the former and $\mathsf{FHE.EvalShift}$ for the latter.

For example, when rotating data one slot to the left, the first component of $\mathbf{x}_i$, $x_{(i-1)k+1}$ will be moved into last component of $\mathbf{x}_{i-1}$, whereas all other components remain in $\mathbf{x}_i$ but are moved 1 slot to the left, i.e. $(x_{(i-1)k+2}, ..., x_{i\cdot k}, y)$, where y would come from $\mathbf{x}_{i+1}$. On the other hand, moving by $k+1$ slots to the left means that $x_{(i-1)k+1}$ will be moved to slot $i-2$ and $(x_{(i-1)k+2}, ..., x_{i\cdot k}, y)$ moved to slot $i-1$.

- $c' \leftarrow \mathsf{FIMD.EvalRot}(evk, \rho, c)$: Let $|\rho| = \rho_{\mathsf{SIMD}} k + \rho_{\mathsf{RMFE}}$.
  - If $\rho_{\mathsf{RMFE}} = 0$, output $c' = \mathsf{FHE.EvalRot}(evk, \rho, c)$.
  - Otherwise, $\rho_{\mathsf{RMFE}} > 0$. Let τ_0 and τ_1 be $\mathbb{F}_q$-linear maps on $(\mathbb{F}_q)^k$ with $\tau_0(\mathbf{z}) = (z_{\rho_{\mathsf{RMFE}}+1}, ..., z_k, 0, ..., 0)$ and $\tau_1(\mathbf{z}) = (0, ..., 0, z_1, ..., z_{\rho_{\mathsf{RMFE}}})$ for $\mathbf{z} = (z_1, ..., z_k)$.
    1. Compute ciphertexts $c_{\tau_0} = \mathsf{FHE.EvalLinearMap}(evk, f_{\tau_0}, c)$ and $c_{\tau_1} = \mathsf{FHE.EvalLinearMap}(evk, f_{\tau_1}, c)$;
    2. Positive (negative) ρ mean rotation to the right (left). To move data correctly, for $i = 0, 1$,

    $\rho > 0$: compute $c'_{\tau_i} = \mathsf{FHE.EvalRot}(evk, \rho_{\mathsf{SIMD}} - i, c_{\tau_i})$;

    $\rho < 0$: compute $c'_{\tau_i} = \mathsf{FHE.EvalRot}(evk, -\rho_{\mathsf{SIMD}} - (1 - i), c_{\tau_i})$.

    3. Output $c' = \mathsf{FIMD.EvalAdd}(evk, c'_{\tau_0}, c'_{\tau_1})$.
- $c' \leftarrow \mathsf{FIMD.EvalShift}(evk, \rho, c)$: Follow the steps of $\mathsf{FIMD.EvalRot}(evk, \rho, c)$, replacing $\mathsf{FHE.EvalRot}$ with $\mathsf{FHE.EvalShift}$.

3.2 r-fold RMFE

Among the possible plaintext spaces $(\mathbb{F}_{t^d})^\ell$ available in the BGV/BFV FHE schemes, the possible set of (d, ℓ) can be very diverse. Many times, t is chosen so that $d = 1$ and ℓ is maximized. However, this requires large t because $d = 1 \Leftrightarrow t \mid (m - 1)$, where m refers to the m-th cyclotomic polynomial $\Phi_m(X)$ in $\mathcal{P} = \mathbb{Z}_t[X]/\Phi_m(X)$.

In the case for small $t = 2, 3, 5, 7, ...$, we can choose m to minimize d but it is not always possible. More specifically, most HE implementations support cyclotomic polynomials of the form $\Phi_m(X) = X^{m/2} + 1$, where $m = 2^N$ for some positive integer N. For such m, small primes tend to have very high d in the range of $m/4, ..., m/32$, which using $m = 32768$ as an example, would translate to $d \in \{512, 1024, 2048, 4096\}$. With such high d, using $\mathsf{FHE.EvalLinearMap}$ after each multiplication would be prohibitively expensive and alternatives are needed.

Instead of just 1 multiplication before decoding, we generalize RMFE to an embedding that allows r-fold multiplications before decoding is strictly necessary. This means that products of up to 2^r encoded vectors can be done before ψ is applied by multiplying the vectors pair-wise recursively.

Definition 2. *Let q be a prime power and $\mathbb{F}_q$ denote the finite field of q elements. For integers $k, w, r \geq 1$, a $(k, w, r)_q$-RMFE is a pair of $\mathbb{F}_q$-linear maps, (ϕ, ψ), where $\phi : (\mathbb{F}_q)^k \to \mathbb{F}_{q^w}$ and $\psi : \mathbb{F}_{q^w} \to (\mathbb{F}_q)^k$ such that for any 2^r vectors $\mathbf{x_1}, ..., \mathbf{x_{2^r}} \in (\mathbb{F}_q)^k$,*

$$\bigstar_{i=1}^{2^r} \mathbf{x_i} = \psi\left(\prod_{i=1}^{2^r} \phi(\mathbf{x_i})\right).$$

This allows us to effectively amortize the expensive recoding step over several multiplications instead, although this reduces the packing capacity as we will show. Besides that, in practice, linear map evaluations incur some noise increase

and reducing the number of recoding steps also reduces the noise overhead of FIMD multiplications.

With this new property, it is desirable to be able to add intermediate products of r-fold RMFE-encoded vectors, regardless of the number of multiplications these products have undergone. This way, we can easily perform degree-r multivariate polynomial evaluations simultaneously on all components of r-fold RMFE-encoded vectors. While it is not a strict requirement in Definition 2, we will focus on constructions that support this. To that end, we need an additional condition on algebraic function field RMFEs to ensure inter-operability between encoded vectors that go through a different number of multiplications.

Definition 3. *Let $K/\mathbb{F}_q$ be an algebraic function field. A divisor, $G = \sum a_P P$ in K, is called positive if $a_P \geq 0$ for all P.*

For a positive divisor G, $\mathcal{L}(G)$ is the set of functions that may have poles of order at most a_P at the places P with $a_P \neq 0$ in G. This means that $\mathcal{L}(xG) \subseteq \mathcal{L}(yG)$ for any $x \leq y$, since functions in $\mathcal{L}(xG)$ can only have poles of order less than $x \cdot a_P \leq y \cdot a_P$.

Theorem 4 (Extending Theorem 2). *Let $K/\mathbb{F}_q$ be an algebraic function field with genus g and k distinct rational places $P_1, ..., P_k$. Let G be a positive divisor of K such that $supp(G) \cap \{P_1, ..., P_k\} = \emptyset$ and $\ell(G) - \ell(G - \sum_{i=1}^{k} P_i) = k$. If there exists a place R with $w = \deg R > 2^r \deg G$, then there exists a $(k, w, r)_q$-RMFE.*

Proof. As before, we have the evaluation map from $\mathcal{L}(G)$ to the rational places,

$$\mathcal{E}_1 : \mathcal{L}(G) \to (\mathbb{F}_q)^k; f \mapsto (f(P_1), ..., f(P_k)).$$

We choose a k-dimensional subspace $W \subset \mathcal{L}(G)$ such that $\mathcal{E}_1$ restricted to W is an isomorphism between W and $(\mathbb{F}_q)^k$ for encoding. Then, with $f(R)$ denoting the evaluation of any $f \in K$ at R, the RMFE encode map is given by

$$\phi : \mathcal{E}_1(W) \cong (\mathbb{F}_q)^k \to \mathbb{F}_{q^w}; \quad (f(P_1), ..., f(P_k)) \mapsto f(R).$$

With a positive divisor, all $\mathcal{L}(xG) \subseteq \mathcal{L}(2^r G)$ and so we focus on the largest space, $\mathcal{L}(2^r G)$. We define the following injective $\mathbb{F}_q$-linear map (since $\deg R > \deg 2^r G$ as well),

$$\mathcal{E}_2 : \mathcal{L}(2^r G) \to \mathbb{F}_{q^w}; \quad f \mapsto f(R).$$

To obtain the RMFE decode map ψ, we first consider the map from the image of $\mathcal{E}_2$ to the input space,

$$\psi' : \mathrm{Im}(\mathcal{E}_2) \subseteq \mathbb{F}_{q^w} \to (\mathbb{F}_q)^k; \quad f(R) \mapsto (f(P_1), ..., f(P_k)).$$

Because $\mathcal{E}_2$ is injective, $f \in \mathcal{L}(2^r G)$ is uniquely determined by $f(R)$ and we linearly extend ψ' to all of $\mathbb{F}_{q^w}$ to get ψ.

The correctness of the construction follows for the same reasons in the proof of Theorem 2. With a positive divisor G, any RMFE-encoded vectors that undergone some number of multiplications would lie in $\mathcal{L}(xG) \subseteq \mathcal{L}(2^r G)$ for some $x \leq 2^r$ and thus can be added together in the "ambient" space $\mathcal{L}(2^r G)$.

Corollary 2 (Extending Corollary 1). *Let $K/\mathbb{F}_q$ be an algebraic function field of genus g and suppose that there are k distinct rational places $(P_1, ..., P_k)$ and a place of degree $w \geq 2^r k + 2^{r+1}g - 2^r + 1$. Then, there exists a $(k, w, r)_q$-RMFE.*

Proof. Like the proof of Corollary 1, we choose a divisor G of degree $k + 2g - 1$, whose support is disjoint from $(P_1, ..., P_k)$. Now, to apply Theorem 4 we require $w = \deg R > 2^r \deg G$, and therefore can get a $(k, w, r)_q$-RMFE as long as $w > 2^r(k + 2g - 1)$.

FIMD with r-Fold RMFE. We tag ciphertexts with a *RMFE level*, augmenting a standard BGV/BFV ciphertext c into (c, η), where η denotes the number of multiplications that have been done on c since the data was first RMFE-encoded. This is to keep track of how many multiplications a ciphertext can tolerate without recoding before being rendered useless by excessive multiplications.

Let (ϕ, ψ) be a $(k, w, r)_q$-RMFE and we use rFIMD to denote the combination of r-fold RMFE with SIMD packing methods. First, we highlight the modifications needed when encrypting and decrypting r-fold RMFE vectors.

- $(c, 0) \leftarrow \mathsf{rFIMD.Encrypt}(pk, \mathbf{x} = (x_1, ..., x_{\ell \cdot k}) \in (\mathbb{F}_t)^{\ell \cdot k})$:
  1. Compute $\hat{\mathbf{x}} = \mathsf{FIMD.Encode}(\mathbf{x} = (x_1, ..., x_{\ell \cdot k}))$.
  2. Encrypt $\hat{\mathbf{x}}$ and output $(c = \mathsf{FHE.Encrypt}(\mathsf{pk}, \hat{\mathbf{x}}), 0)$.
- $\mathbf{m} \leftarrow \mathsf{rFIMD.Decrypt}(sk, (c, \eta))$:
  1. If $\eta > r$, abort and output $\perp$. Otherwise, continue to Step 2.
  2. Decrypt the ciphertext to obtain $\mu = \mathsf{FHE.Decrypt}(sk, c)$.
  3. Decode μ and output $\mathbf{m} = \mathsf{FIMD.Decode}(\mu) \in (\mathbb{F}_t)^{\ell \cdot k}$.

Homomorphic operations remain mostly unchanged, especially for addition. The only difference is that the RMFE level of output ciphertexts has to be accounted for. As a side effect of shifts and rotations of RMFE-encoded data being modified recodings, any data movement operation in rFIMD with $\rho = \rho_{\mathsf{SIMD}} k + \rho_{\mathsf{RMFE}}$ with $\rho_{\mathsf{RMFE}} \neq 0$ would reset the RMFE level to zero.

- $(c^+, \eta) \leftarrow \mathsf{rFIMD.EvalAdd}(evk, (c_1, \eta_1), (c_2, \eta_2))$: Set $\eta = \max(\eta_1, \eta_2)$ and output $(c^+ = \mathsf{FIMD.EvalAdd}(evk, c_1, c_2), \eta)$.
- $(c^\times, \eta) \leftarrow \mathsf{rFIMD.EvalMul}(evk, (c_1, \eta_1), (c_2, \eta_2))$: Let $\eta' = \max(\eta_1, \eta_2) + 1$. We distinguish between two cases, $\eta' = r$ and $\eta' < r$.
  1. If $\eta' = r$, compute the recoded result $c^\times = \mathsf{FIMD.EvalMul}(evk, c_1, c_2)$ and output $(c^\times, 0)$.
  2. Otherwise, $\eta' < r$ and output $(c^\times = \mathsf{FHE.EvalMul}(evk, c_1, c_2), \eta')$.
- $(c', \eta') \leftarrow \mathsf{rFIMD.EvalShift}(evk, \rho, (c, \eta))$: Let $\rho = \rho_{\mathsf{SIMD}} k + \rho_{\mathsf{RMFE}}$.
  Return $(c' = \mathsf{FIMD.EvalShift}(evk, \rho, c), \eta')$, where $\eta' = 0$ if $\rho_{\mathsf{RMFE}} \neq 0$ and $\eta' = \eta$ otherwise.
- $(c', \eta) \leftarrow \mathsf{rFIMD.EvalRot}(evk, \rho, (c, \eta))$: Let $\rho = \rho_{\mathsf{SIMD}} k + \rho_{\mathsf{RMFE}}$.
  Return $(c' = \mathsf{FIMD.EvalRot}(evk, \rho, c), \eta')$, where $\eta' = 0$ if $\rho_{\mathsf{RMFE}} \neq 0$ and $\eta' = \eta$ otherwise.

3.3 Composite RMFE with FHE

Recall from Theorem 3 that composite RMFEs $(k_{\mathsf{in}}k_{\mathsf{out}}, w_{\mathsf{in}}w_{\mathsf{out}})_q$ are built from two component RMFEs, an "inner" $(k_{\mathsf{in}}, w_{\mathsf{in}})_q$-RMFE and "outer" $(k_{\mathsf{out}}, w_{\mathsf{out}})_{q^{w_{\mathsf{in}}}}$-RMFE with the maps $(\phi_{\mathsf{in}}, \psi_{\mathsf{in}})$ and $(\phi_{\mathsf{out}}, \psi_{\mathsf{out}})$ respectively. This allows us to design a three-stage method for recoding that leverages the simpler linear maps $(\phi_{\mathsf{in}}, \psi_{\mathsf{in}})$ and $(\phi_{\mathsf{out}}, \psi_{\mathsf{out}})$. However, it also presents complications for extending the r-fold property to composite RMFEs, which we describe and address at the end of the section by relaxing the recoding requirements for composite RMFEs.

Exploiting the Intermediate Extension. The key difference between standard and composite RMFEs is the tower of field extensions of $\mathbb{F}_q$ underlying composite RMFEs, $\mathbb{F}_q \subseteq \mathbb{E}_1 = \mathbb{F}_{q^{w_{\mathsf{in}}}} \subseteq \mathbb{E}_2 = \mathbb{F}_{q^{w_{\mathsf{in}}w_{\mathsf{out}}}}$. Furthermore, their respective extension degrees $[\mathbb{E}_1 : \mathbb{F}_q] = w_{\mathsf{in}}$ and $[\mathbb{E}_2 : \mathbb{E}_1] = w_{\mathsf{out}}$ are smaller than the direct extension $[\mathbb{E}_2 : \mathbb{F}_q] = w_{\mathsf{in}}w_{\mathsf{out}}$. This means that $\mathbb{E}_1$-linear maps on $\mathbb{E}_2$ and $\mathbb{F}_q$-linear maps on $\mathbb{E}_1$ correspond to $|\mathbb{E}_1|$- and $|\mathbb{F}_q|$-linearized polynomials of lower degrees and thus easier to evaluate.

We propose a three-stage recode process for composite RMFEs, exploiting the intermediate field $\mathbb{E}_1$. Let $\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_{k_{\mathsf{out}}}), \mathbf{y} = (\mathbf{y}_1, ..., \mathbf{y}_{k_{\mathsf{out}}}) \in (\mathbb{F}_q)^{k_{\mathsf{in}}k_{\mathsf{out}}}$ be vectors to be encoded, where $\mathbf{z}_i = (z_{(i-1)k_{\mathsf{in}}+1}, ..., z_{ik_{\mathsf{in}}}) \in (\mathbb{F}_q)^{k_{\mathsf{in}}}$ for $\mathbf{z} \in \{\mathbf{x}, \mathbf{y}\}$. Denoting with $\alpha \in \mathbb{E}_2$ the result of $\phi(\mathbf{x}) \cdot \phi(\mathbf{y})$, we perform recoding in the following manner,

1. Compute $\psi_{\mathsf{out}}(\alpha) = (\phi_{\mathsf{in}}(\mathbf{x}_1) \cdot \phi_{\mathsf{in}}(\mathbf{y}_1), ..., \phi_{\mathsf{in}}(\mathbf{x}_{k_{\mathsf{out}}}) \cdot \phi_{\mathsf{in}}(\mathbf{y}_{k_{\mathsf{out}}})) \in (\mathbb{E}_1)^{k_{\mathsf{out}}}$.
2. Apply the recode map $\pi_{\mathsf{in}} = \phi_{\mathsf{in}} \circ \psi_{\mathsf{in}}$ to each component of $\psi_{\mathsf{out}}(\alpha)$, recoding the intermediate field elements from the "inner" RMFE.

$$\Big(\pi_{\mathsf{in}}\big(\phi_{\mathsf{in}}(\mathbf{x}_i) \cdot \phi_{\mathsf{in}}(\mathbf{y}_i)\big)\Big)_{i=1}^{k_{\mathsf{out}}} = (\phi_{\mathsf{in}}(\mathbf{x}_1 * \mathbf{y}_1), ..., \phi_{\mathsf{in}}(\mathbf{x}_{k_{\mathsf{out}}} * \mathbf{y}_{k_{\mathsf{out}}})) \in (\mathbb{E}_1)^{k_{\mathsf{out}}}$$

3. Encode the resulting vector $(\phi_{\mathsf{in}}(\mathbf{x}_1 * \mathbf{y}_1), ..., \phi_{\mathsf{in}}(\mathbf{x}_{k_{\mathsf{out}}} * \mathbf{y}_{k_{\mathsf{out}}}))$ with ϕ_{out}, getting $\alpha' = \phi_{\mathsf{out}}((\phi_{\mathsf{in}}(\mathbf{x}_1 * \mathbf{y}_1), ..., \phi_{\mathsf{in}}(\mathbf{x}_{k_{\mathsf{out}}} * \mathbf{y}_{k_{\mathsf{out}}}))) \in \mathbb{E}_2$.

Three-Stage Recode for FIMD. As with standard recode, we evaluate linear maps in each stage. However, Stages 1 and 3 work over an $\mathbb{E}_1$-vector space while Stage 2 work over an $\mathbb{F}_q$-vector space. Stages 1 and 3 correspond to applying the "outer" RMFE decoding and encoding maps, $\psi_{\mathsf{out}}, \phi_{\mathsf{out}}$ respectively. These would correspond to evaluating two $q^{w_{\mathsf{in}}}$-linearized polynomials, one per map, following Lemma 1.

Let ϕ'_{in} and ψ'_{in} denote extensions of ϕ_{in} and ψ_{in} to $\mathbb{F}_q$-linear maps over $(\mathbb{E}_1)^{k_{\mathsf{out}}}$ that perform component-wise encoding and decoding of the $\mathbb{E}_1$ elements. In the second stage, notice that the map

$$\begin{aligned} \pi'_{\mathsf{in}} : (\mathbb{E}_1)^{k_{\mathsf{out}}} &\to (\mathbb{E}_1)^{k_{\mathsf{out}}} \\ (\alpha_1, \ldots, \alpha_{k_{\mathsf{out}}}) &\mapsto \phi'_{\mathsf{in}}\left(\psi'_{\mathsf{in}}\left((\alpha_1, \ldots, \alpha_{k_{\mathsf{out}}})\right)\right) \\ &= (\phi_{\mathsf{in}}(\psi_{\mathsf{in}}(\alpha_1)), ..., \phi_{\mathsf{in}}(\psi_{\mathsf{in}}(\alpha_{k_{\mathsf{out}}}))) \end{aligned}$$

is $\mathbb{F}_q$-linear since $(\mathbb{E}_1)^{k_{\mathsf{out}}}$ is the product of k_{out} copies of the $\mathbb{F}_q$-vector space $\mathbb{E}_1$ and π is the product of k_{out} copies of π_{in}. Thus, we can evaluate one q-linearized polynomial to achieve Stage 2.

In fact, as with the recode operation with standard RMFEs, we can view π'_{in} as a recode operation over the entire input vector space $\mathbb{F}_q^{k_{\mathsf{in}}k_{\mathsf{out}}}$. Then, we can similarly enhance the three-stage recode process to also evaluate arbitrary linear transformations τ over elements in $\mathbb{F}_q^{k_{\mathsf{in}}k_{\mathsf{out}}}$.

$$\begin{aligned}&\pi'_{\mathsf{in},\tau} : (\mathbb{E}_1)^{k_{\mathsf{out}}} \to (\mathbb{E}_1)^{k_{\mathsf{out}}}\\ &(\alpha_1, \ldots, \alpha_{k_{\mathsf{out}}}) \mapsto \phi'_{\mathsf{in}}\left(\tau\left(\psi'_{\mathsf{in}}\left((\alpha_1, \ldots, \alpha_{k_{\mathsf{out}}})\right)\right)\right)\end{aligned}$$

With this, we get an alternate FIMD multiplication algorithm for composite RMFEs. Similar to $\mathsf{FIMD.EvalMul}'$ in Section 3.1, we can evaluate arbitrary linear transformations τ during FIMD multiplication. This is achieved by replacing $f_{\pi'_{\mathsf{in}}}$ with the appropriate linearized polynomial for $\pi'_{\mathsf{in},\tau}$ in Stage 2 of $\mathsf{FIMD.Recode3S}$.

- $c' \leftarrow \mathsf{FIMD.Recode3S}(evk, c)$: Let $f_{\psi_{\mathsf{out}}}$, $f_{\pi'_{\mathsf{in}}}$, and $f_{\phi_{\mathsf{out}}}$ denote the linearized polynomials required in the three stage recoding process.
  1. Compute $c^{(1)} = \mathsf{FHE.EvalLinearMap}(evk, f_{\psi_{\mathsf{out}}}, c)$.
  2. Compute $c^{(2)} = \mathsf{FHE.EvalLinearMap}(evk, f_{\pi'_{\mathsf{in}}}, c^{(1)})$.
  3. Output $c' = \mathsf{FHE.EvalLinearMap}(evk, f_{\phi_{\mathsf{out}}}, c^{(2)})$.
- $c^{\times} \leftarrow \mathsf{FIMD.EvalMul^c}(evk, c_1, c_2)$:
  1. Compute $c = \mathsf{FHE.EvalMul}(evk, c_1, c_2)$.
  2. Output $c^{\times} = \mathsf{FIMD.Recode3S}(evk, c)$.

As with standard RMFEs, r-fold RMFEs can be composed just like the original RMFE.

Theorem 5 (Composite r-fold RMFE). *Let $(\phi_{\mathsf{in}}, \psi_{\mathsf{in}})$ be a $(k_{\mathsf{in}}, w_{\mathsf{in}}, r)_q$-RMFE and $(\phi_{\mathsf{out}}, \psi_{\mathsf{out}})$ be a $(k_{\mathsf{out}}, w_{\mathsf{out}}, r)_{q^{w_{\mathsf{in}}}}$-RMFE. Then, their composition in the manner of Theorem 3, denoted with (ϕ, ψ), is a $(k_{\mathsf{in}}k_{\mathsf{out}}, w_{\mathsf{in}}w_{\mathsf{out}}, r)_q$-RMFE.*

The proof of this is exactly the same as $(k, w)_q$-RMFE composition, since the composition of $\mathbb{F}_q$-linear maps are $\mathbb{F}_q$-linear and field elements decode to component-wise products of their respective encoded input vectors after each decode step. If both $(\phi_{\mathsf{in}}, \psi_{\mathsf{in}})$ and $(\phi_{\mathsf{out}}, \psi_{\mathsf{out}})$ allow mixing "intermediate" products, then the composed r-fold RMFE (ϕ, ψ) will also have this property.

Relaxing the r-fold Property for Composite RMFEs. Recode can be delayed up until 2^r encoded vectors are multiplied for any r-fold RMFEs. However, r-fold RMFEs from Theorem 5 is less space-efficient than r-fold RMFEs derived from Theorem 4. This is due to the fact that r-fold RMFEs have $w > 2^r \cdot k$ and so a composite r-fold RMFE from Theorem 5 would typically have $w_{\mathsf{in}}w_{\mathsf{out}} > (2^{2r}) \cdot k_{\mathsf{in}}k_{\mathsf{out}}$ per Corollary 2. With FHE, $w_{\mathsf{in}}w_{\mathsf{out}}$ is generally a dependent variable – t and m are the main parameters – rendering r-fold composite RMFEs almost unusable.

To remedy this situation, we relax the r-fold property such that for composite r-fold RMFEs, one can perform a less expensive "outer" recode after r'-fold multiplications for some $r' \mid r$ and only do a complete recode process after r-fold multiplications.

Theorem 6 (Composite r-fold RMFE, Relaxed Recode). *Let $(\phi_{\mathsf{in}}, \psi_{\mathsf{in}})$ be a $(k_{\mathsf{in}}, w_{\mathsf{in}}, r)_q$-RMFE and $(\phi_{\mathsf{out}}, \psi_{\mathsf{out}})$ be a $(k_{\mathsf{out}}, w_{\mathsf{out}}, r')_{q^{w_{\mathsf{in}}}}$-RMFE, for some $r' \mid r$. Then, their composition following Theorem 3, denoted with (ϕ, ψ), is a $(k_{\mathsf{in}}k_{\mathsf{out}}, w_{\mathsf{in}}w_{\mathsf{out}}, r)_q$-RMFE, provided $\pi_{\mathsf{out}} = \phi_{\mathsf{out}} \circ \psi_{\mathsf{out}}$ is evaluated on encoded elements after every r'-fold multiplications.*

Proof. Let ϕ'_{in} and ψ'_{in} denote extensions of the "inner" RMFE encode and decode maps to act component-wise on vectors in $(\mathbb{E}_1)^{k_{\mathsf{out}}}$. Suppose we have 2^r inputs, $\mathbf{x}_i = (x_{1,i}, ..., x_{k_{\mathsf{in}}k_{\mathsf{out}},i})$ for $1 \leq i \leq 2^r$ and let their respective encodings be $\alpha_i = \phi_{\mathsf{out}}(\phi'_{\mathsf{in}}(\mathbf{x}_i))$. Any intermediate result after r'-fold multiplications β would need to be refreshed with $\beta' = \pi_{\mathsf{out}}(\beta)$. Otherwise, further multiplications would fail to correctly decode with ψ_{out} and thus similarly fail to decode with ψ. Finally, observe that the map ψ_{in} tolerates up to $2^r \geq 2^{r'}$ multiplications. Therefore, any intermediate product γ would decode correctly with $\psi'_{\mathsf{in}}(\psi_{\mathsf{out}}(\gamma))$ as long as they have been "outer recoded" after every r'-fold multiplications.

Thus, the "outer" RMFE is no longer restricted to be r-fold and can even be a standard $(k_{\mathsf{out}}, w_{\mathsf{out}})_q$-RMFE if we are willing to perform an "outer" recode after each multiplication. In that case, the overhead between composite and standard r-fold RMFEs would be almost identical. Assuming we are using a composite r-fold RMFE with components $(\phi_{\mathsf{in}}, \psi_{\mathsf{in}})$ and $(\phi_{\mathsf{out}}, \psi_{\mathsf{out}})$ of $(k_{\mathsf{in}}, w_{\mathsf{in}}, r)_q$ and $(k_{\mathsf{out}}, w_{\mathsf{out}}, r')_q$-RMFEs respectively, we have

- $(c^\times, \rho) \leftarrow \mathsf{rFIMD.EvalMul^c}(evk, (c_1, \rho_1), (c_2, \rho_2))$: Let $\rho' = \max(\rho_1, \rho_2) + 1$ and $\pi_{\mathsf{out}} = \phi_{\mathsf{out}} \circ \psi_{\mathsf{out}}$.
  1. Compute $c = \mathsf{FHE.EvalMul}(evk, c_1, c_2)$.
  2. Then, we consider three cases based on ρ'.
     (a) If $\rho' = r$, output $(c^\times = \mathsf{FIMD.Recode3S}(evk, c), 0)$.
     (b) Else, if $\rho' \mid r'$, output $(c^\times = \mathsf{FHE.EvalLinearMap}(evk, \pi_{\mathsf{out}}, c), \rho')$.
     (c) Otherwise, output (c, ρ').

4 RMFE Parameter Selection with FHE

In this section, we describe how parameters should be chosen for RMFE with FHE. As introduced in Section 2.1, for chosen t and m, the FHE plaintext space is $(\mathbb{F}_{t^d})^\ell$, where $d \cdot \ell = \phi(m)$. Therefore, we are limited to $(k, w, r)_q$-RMFEs where $q^w \leq t^d$ and q is some power of t.

For the various forms of RMFE discussed in previous sections, the main parameter is the function field K used in Theorems 2 and 4. First, we introduce the Hasse-Weil bound, which gives an upper bound on the number of rational places of a function field. For every function field K, there is a unique non-singular projective curve C associated with it. It was shown that there is a one-to-one correspondence between the points on the curve C and the rational places of K.

Lemma 2 (Hasse-Weil Bound, [38, Theorem 5.2.3]). *Let $K/\mathbb{F}_q$ be an algebraic function field with genus g. The number of rational places of K, η, satisfies*

$$|\eta - (q+1)| \leq 2g\sqrt{q}.$$

Function Field	Base Field, $\mathbb{F}_q$	Genus, g	Max. k, η	$\deg G$	Min. $\deg R$, r-fold
Rational (P)	$\mathbb{F}_t$	0	$= t+1$	$k-1$	$2^r(k-1)+1$
Elliptic (E)	$\mathbb{F}_t$	1	$\leq t+1+2\sqrt{t}$	$k+1$	$2^r(k+1)+1$
Hermitian (H)	$\mathbb{F}_{t^2}$	$\frac{t(t-1)}{2}$	$= t^3+1$	$k+t^2-t-1$	$2^r(\deg G)+1$

Table 1. Possible $(k, \deg R, r)_q$-RMFE parameters

A collection of curves that satisfy the Hasse-Weil bound can be found at [23].

Packing Density. The packing density of an RMFE instantiation can be defined as w/k. Cascudo et al. showed that there existed families of RMFEs with good asymptotic packing density.

Theorem 7 ([7, Theorem 5]). *There exists a family of $(k, w)_t$-RMFE with $k \to \infty$ and $w = O(k)$. More concretely,*

$$\frac{w}{k} \to 2 + \frac{4}{A(t)},$$

where $A(t)$ is Ihara's constant of $\mathbb{F}_t$.

We extend the definition of packing density to FIMD by computing d/k and not w/k because the FHE plaintext space is fixed to extension degree d with the choice of t and m. A smaller number means that the FIMD instantiation can effectively use a larger portion of the underlying field.

Function Fields for Efficient RMFE for FHE. To make good use of the finite extension fields available from SIMD, we consider the following function fields that yield RMFEs with w/k close to 2. Details of the possible RMFEs enabled by these function fields are given in Table 1. The minimum degree of the place R is derived from Corollaries 1 and 2.

- Rational Function Field, $F_t(X)$: Corresponding to choosing the projective line as the underlying curve.
- "Elliptic" Function Fields, $F_t[X, Y]/C$: C is an appropriate elliptic curve that approaches the Hasse-Weil bound.
- Hermitian Function Field, $F_{t^2}[X, Y]/C$: $C = Y^t + Y - X^{t+1}$ is the Hermitian curve, and the function field satisfies the Hasse-Weil bound exactly.

When using Hermitian curves, note that the FHE slot degree d is effectively halved as the base field of the Hermitian function field is $\mathbb{F}_q = \mathbb{F}_{t^2}$.

Existence of Higher Degree Place R. Given a function field $K/\mathbb{F}_q$, the choice of w is dependent on whether K even admits a place R with $\deg R = w$. The following lemma gives conditions for the existence of places of a given degree.

Lemma 3 ([38, Corollary 5.2.10 (b), (c)]). *Let $K/\mathbb{F}_q$ be an algebraic function field with genus g.*

1. *If $g = 0$, then there exists a place of degree w, for all $w \geq 1$.*
2. *If $g \geq 1$, it is sufficient that w satisfies*

$$2g + 1 \leq q^{\frac{w-1}{2}}(q^{\frac{1}{2}} - 1),$$

for there to exist a place R of degree w.

The function fields from Table 1 have relatively small values of g, ensuring us a wide selection of w. More explicitly, let w_0 be the value derived from Lemma 3 such that for all $w \geq w_0$, Lemma 3 guarantees the existence of a place of degree w for a given function field. Table 2 records the values of w_0 for each of the function fields from Table 1.

Function Field	w_0
Rational (P)	1
Elliptic (E)	$2(\log_t 3 - \log_t(t^{1/2} - 1)) + 1$
Hermitian (H)	$\log_t \left(t + \frac{1}{t-1}\right) + 1$

Table 2. Lower bound on degree of places guaranteed to exist by Lemma 3

Composite RMFEs for FHE. For composite RMFEs, r-fold or otherwise, there are more considerations for the choices of component RMFEs. If we do not need the r-fold property, we could choose component RMFEs, $(k_{\mathsf{in}}, w_{\mathsf{in}})$ and $(k_{\mathsf{out}}, w_{\mathsf{out}})$ such that the complexity of each stage of FIMD.Recode3S is about the same. This entails balancing $w_{\mathsf{in}} \cdot k_{\mathsf{out}}$ with w_{out} as these determine the degree of the linear maps computed in Stage 2 and Stages 1 and 3 respectively.

For composite r-fold RMFEs, the main choices are in the sizes of r' and k_{out}. Larger r' means fewer recoding operations but reduces the potential packing efficiency, while k_{out} determines how expensive the "outer" recoding operation would be. An option is to have cheap "outer" recodes and a more expensive three-stage recode since the latter would be amortized over r-fold multiplications.

5 Experiment Results

In this section, we discuss the results of our experiments on the performance of standard and r-fold RMFEs, as well as the three-stage recode optimization for composite (r-fold) RMFEs. The experiment platform is an Intel® Xeon® Platinum 8170 with maximum turbo frequency of 3.7 GHz and 192 GB RAM. We do not use multi-threading for the experiments in this section.

Plaintext modulus, t	# SIMD Slots, ℓ	Extension Degree, d
3	2	2048
7	4	1024
17	8	512
31	16	256

Table 3. FHE plaintext spaces for various primes with $\Phi_{8192}(X)$

Throughout this section, we use $\Phi_{8192}(X) = X^{4096} + 1$ and various plaintext modulus t. Magma was used to implement the RMFEs and compute the necessary data to use with HElib. The capacity parameter in HElib is set to 99 and yields a maximum ciphertext bit-width of < 159. Estimations with the `lwe-estimator` of Albrecht et al. [1] shows the FHE instance achieving at least 80-bit security. Table 3 shows the decomposition of the plaintext space for $m = 8192$ with respect to various primes. We split our experiments into two main categories: FIMD with basic and composite (r-fold) RMFE.

rFIMD Implementation Details. The main component of rFIMD is the recode operation, which consists of evaluating one or more linear maps on FHE ciphertexts to refresh the RMFE encoding encrypted within them. To that end, we generated the key-switching matrices for all necessary automorphisms in the recode operation, which is at most, w matrices for a $(k, w, r)_q$-RMFE. This allows us to fully exploit the hoisting technique of Halevi and Shoup [28]. We evaluate each linear map in the recode operation by first hoisting the input ciphertext and then computing the required automorphisms one by one to minimize the number of ciphertexts in memory. Besides that, due to the large noise increases from the recode computation, we apply modulus switching to rescale the resulting ciphertext. This reduces the ciphertext modulus based on the current estimated noise levels and improves the performance of multiplications down the line.

5.1 Experimental Results for basic (r)FIMD

A list of the parameters used for basic RMFEs is shown in Table 4. These parameters are chosen by maximizing k for each function field and they also support r-fold variants for small $r \in \{1, 2, 4\}$. Note that Hermitian function fields were not considered for $t = 17, 31$, as $\deg G$ would exceed d. We also had to reduce k for $t = 31$ with higher r values for the same reason that $\deg G$ would exceed d. We denote an RMFE parameter set by t-*curve_type*, where *curve_type* $\in$ {P,E,H} indicates the rational (P for projective line), elliptic and Hermitian function fields respectively. For example, 17-P represents the case where $t = 17$ and the RMFE is instantiated with the rational function field.

The first set of experiments compared the performance and noise impact of (r)FIMD multiplication to FHE multiplication. We prepared one (r)FIMD ciphertext and one FHE ciphertext, which was repeatedly squared until their

t	d	K	Parameter (P) Set	Curve	k	d/k	w
3	2048	Projective	3-P	-	4	512	7
		Elliptic	3-E	$y^2 - x^3 - 2x - 1$	8	293	17
		Hermitian	3-H	$y^3 + y - x^4$	28	73.1	67
7	1024	Projective	7-P	-	8	128	15
		Elliptic	7-E	$y^2 - x^3 - 3$	13	78.8	29
		Hermitian	7-H	$y^7 + y - x^8$	214	4.79	511
17	512	Projective	17-P	-	18	28.4	35
		Elliptic	17-E	$y^2 - x^3 - 3x$	26	19.7	55
31	256	Projective	31-P	-	16.0	8.0	63
					32.0	16.0	31
		Elliptic	31-E	$y^2 - x^3 - 3$	14	18.3	89
					43	5.95	31

Table 4. Basic RMFE parameters

capacities were exhausted. The time taken to complete this process as well as the overall number of multiplications that were done were recorded. For better comparison against (r)FIMD multiplications, we took as many timings from the last few FHE multiplications onwards, so that the same number of multiplications are compared. This is because HElib implements the BGV scheme whose multiplications become cheaper due to the use of modulus switching after each multiplication for noise control. The complete set of basic (r)FIMD experiments are described in Table 11, furnished in Appendix A. As we observe a similar trend across the different parameter sets, a subcollection will be used to facilitate the discussion about the experiment results in Table 5.

In general, (r)FIMD multiplications take much longer to complete than FHE multiplications. We observe a trend of better amortized (r)FIMD multiplication speedup as r increases. The speedup is primarily attributed to the decrease in (r)FIMD multiplication time as the number of recodes performed is reduced. Sometimes, recoding is not necessary as no more operations can be executed after the maximum FIMD multiplications are achieved. Table 5 show that by suppressing the recodes for P Sets 3-H and 7-H, we are able to obtain an amortized speedup of more than 20× and 11× respectively. This shows that recode is indeed an expensive operation that should be used sparingly.

We also see that higher k values are needed to see benefits with (r)FIMD. These k values are dependent on the type of the function field, with RMFEs from Hermitian function fields yielding the highest k, for any fixed r. Hence, it is beneficial to perform fewer recodes while maximizing the value of k for a basic (r)FIMD instantiation. Note that the number of multiplications supported for any ciphertext varies slightly, due to variance in the noise generated in fresh ciphertexts.

P Set	k	r	Max # rFIMD Mult	Max # FHE Mult	rFIMD Mult (sec)	FHE Mult (sec)	Speedup v.s. k FHE Mult
3-H	28	1	3	5	3.59	0.0245	0.191×
		2	4	5	2.31	0.0355	0.431×
		4	4	5	0.900	0.0400	1.24×
		4⋆	4	5	0.0541	0.0389	20.1×
7-P	8	1	3	5	3.74	0.0242	0.0516×
		2	4	5	2.40	0.0359	0.120×
		4	4	5	1.16	0.0378	0.261×
		4⋆	4	5	0.0431	0.0404	11.0×
7-H	214	1	3	5	1.83	0.0238	2.79×
17-P	18	1	2	5	1.16	0.0245	0.380×
		2	4	5	1.12	0.0348	0.558×
		4	4	5	0.428	0.0381	1.61×
31-E	43	1	2	4	0.578	0.0236	1.76×
	43	2	3	4	0.382	0.0251	2.82×
	14	4⋆	3	4	0.0258	0.0263	14.3×

⋆No recodes were performed

Table 5. Selected experiments comparing (r)FIMD and FHE multiplication performance for basic RMFEs

Furthermore, our implementation of homomorphic linear map evaluation computes one monomial at a time and consumes it immediately; leaving $0.5k\times$ fewer ciphertexts (during peak operation) in memory compared to FHE. This can be adjusted to trade off improved FIMD multiplication speeds by computing several monomials at once using multiple cores. We observed that the recode operation roughly consumes the noise budget of one multiplication, implying that the standard RMFEs defined in Section 2.3 would yield about 1/2 the number of FHE multiplications. Overall, r-fold RMFEs have an important role in balancing FIMD multiplication performance and retaining a sizable proportion of multiplications for any given capacity.

5.2 Experimental Results for composite RMFE

As Table 3 shows, it is very difficult to work with small primes as the extension degrees of their plaintext slot algebra are exceedingly high (> 1000). Therefore, we investigated the effectiveness of composite RMFEs for FIMD in such cases. For our choice of $m = 8192$, it would not be meaningful to use composite RMFEs for $t = 17, 31$ and we focus on $t = 3, 7$ in this section. Due to the high degree, it is very expensive to generate the recode map π and so we focus on the three-stage recode process described in Section 3.3.

Just like the previous section, we consider the packing density of a composite RMFE instantiation with d/k, where $k = k_{\mathsf{in}} \cdot k_{\mathsf{out}}$. The degree $d' = [\mathbb{E}_1 : \mathbb{F}_q]$

is chosen as the next largest power-of-two from w_{in}. For larger r-fold values, we can adjust the intermediate degree accordingly. We identify a composite RMFE parameter set by a prefix C, e.g. C7-E, and present the parameters used in the experiments to follow in Table 6.

t	d	K	P Set	Curve	$(k_{\mathsf{total}}, d/k_{\mathsf{total}})$
3	2048	Projective	C3-P	-	$(8, 256)$, $(16, 128)$, $(32, 64)$, $(64, 32)$, $(128, 16)$, $(256, 8)$, $(512, 4)$
		Elliptic	C3-E	$y^2 - x^3 - 2x - 1$	$(24, 85.3)$, $(48, 42.7)$, $(64, 32)$, $(96, 21.3)$, $(128, 16)$, $(192, 10.7)$, $(384, 5.33)$
		Hermitian	C3-H	$y^3 + y - x^4$	$(24, 85.3)$, $(44, 46.5)$, $(48, 42.7)$, $(88, 23.2)$, $(96, 21.3)$, $(108, 19.0)$, $(276, 11.6)$, $(216, 216)$
7	1024	Projective	C7-P	-	$(32, 32)$, $(64, 16)$, $(128, 8)$, $(256, 4)$
		Elliptic	C7-E	$y^2 - x^3 - 3$	$(26, 39.4)$, $(48, 21.3)$, $(52, 19.7)$, $(96, 10.7)$, $(104, 9.85)$, $(208, 4.92)$
		Hermitian	C7-H	$y^7 + y - x^8$	$(64, 16)$

Table 6. Composite RMFE parameters

In this second set of experiments, we compared the performance and noise impact of FIMD multiplication with composite RMFEs to FHE multiplication. Similar to the previous experiment, we repeatedly squared FIMD and FHE ciphertext until their capacities were exhausted and recorded the time taken as well as how many squarings could be done.

As illustrated in Table 12 in Appendix A, composite RMFEs are more expensive than basic RMFE and standard FHE. Composite RMFEs, however, offer a greater amortized speedup than basic RMFEs over standard FHE multiplications due to the increase in packing capacity (i.e. lower d/k_{total}). We make a few observations on some of the trends in the results, presenting them with tables featuring appropriate subcollections of Table 12 below.

Our first observation is that there is a slight advantage in choosing an intermediate field such that its extension degree $d' = [\mathbb{E}_1 : \mathbb{F}_q] > [\mathbb{E}_2 : \mathbb{E}_1]$. Considering the results presented in Table 7, we see that a larger d' value, over the same k_{total}, resulted in more than 10% savings in (r)FIMD multiplication time. This is supported by the fact that the 3-stage recode process requires evaluating two $\mathbb{E}_1$-linear maps in Steps 1 and 3 of FIMD.Recode3S and only one $\mathbb{F}_q$-linear map in Step 2. Using larger $\mathbb{E}_1$ reduces the computation time in Steps 1 and 3 while increasing the computation time in Step 2 and we expect that a ratio close to $2:1$ for $[\mathbb{E}_1 : \mathbb{F}_q] : [\mathbb{E}_2 : \mathbb{E}_1]$ would be best for three-stage recode performance in our implementation.

We consider the effect of fixing either r_{in} and r_{out}, while fixing d', on composite RMFEs. A subcollection of the experiments where we fixed r_{out} and d', while varying r_{in}, that supports our observation can be found in Table 8. We observe

P Set	k_{total}	$(k_{\mathsf{in}}, r_{\mathsf{in}})_q$	$(k_{\mathsf{out}}, r_{\mathsf{out}})_{q^{d'}}$	rFIMD Mult (sec)	Speedup v.s. FHE Mult
C7-P	64	$(4,2)_7$	$(16,2)_{7^{16}}$	0.4663130	3.396230
		$(8,2)_7$	$(8,2)_{7^{32}}$	0.4076150	3.734270

Table 7. Effect of Intermediate Field Size on Composite RMFEs

that similar to basic RMFEs, (r)FIMD multiplication timings generally decrease with larger r_{in}. In our implementation, after each (r)FIMD multiplication, the ciphertext capacity drops by similar amounts regardless of recoding type (outer or full). With larger r_{in}, some full recodes are replaced by outer recodes, thereby reducing the time taken. However, due to the smaller k_{total} that accompanies this increase, overall amortized speedup against FHE multiplication actually decreased.

P Set	$(k_{\mathsf{in}}, r_{\mathsf{in}})_q$	$(k_{\mathsf{out}}, r_{\mathsf{out}})_{q^{d'}}$	k_{total}	Max rFIMD Mult	rFIMD Mult (sec)	Speedup v.s. k_{total} FHE Mult
C3-E	$(2,2)_3$	$(64,1)_{3^{16}}$	128	2	1.4284200	1.138740
	$(6,1)_3$	$(64,1)_{3^{16}}$	384	2	2.4344100	1.954400
C3-H	$(3,2)_3$	$(16,1)_{3^{64}}$	48	2	0.4838210	1.189630
	$(11,1)_3$	$(16,1)_{3^{64}}$	176	2	1.1126300	1.891190
	$(11,2)_3$	$(8,1)_{3^{128}}$	88	2	0.4557780	2.271160
	$(27,1)_3$	$(8,1)_{3^{128}}$	216	2	1.0570600	2.641760

Table 8. Effect of r_{in} for Composite RMFEs, Keeping r_{out} and d' fixed

On the other hand, Table 9 features a subcollection of experiments fixing r_{in} and d' while varying r_{out}. We generally get a decrease in (r)FIMD multiplication times as r_{out} increases. This is consistent with earlier trends seen in basic RMFEs and composite RMFEs with fixed r_{out}.

Looking at the parameter sets 7-H and C7-P, we also conclude that three-stage recode is more efficient than direct recode for composite RMFEs. Although we could not compute the direct recode map for C7-P, we approximate its performance by extending from 7-H in Table 10. The theoretical w_{total} for C7-P is $16 \cdot 63 = 1008$, which roughly corresponds to the number of monomials in its direct recode map. We extrapolate the FIMD multiplication timing for C7-P with direct recode by adjusting the multiplication time for 7-H by a factor of $1008/511 \approx 1.97$ as $w = 511$ for 7-H. 7-H took 1.83 seconds for 3 multiplications which give an average of 0.610 seconds per multiplication. One multiplication

P Set	$(k_{\mathsf{in}}, r_{\mathsf{in}})_q$	$(k_{\mathsf{out}}, r_{\mathsf{out}})_{q^{d'}}$	Max rFIMD Mult	rFIMD Mult (sec)	Speedup v.s. FHE Mult
C3-P	$(4,3)_3$	$(16,2)_{3^{32}}$	3	0.5992640	2.602520
	$(4,3)_3$	$(32,1)_{3^{32}}$	2	0.2796210	11.709300
C3-E	$(2,2)_3$	$(32,2)_{3^{16}}$	3	0.8984420	1.212720
	$(2,2)_3$	$(64,1)_{3^{16}}$	2	1.4284200	1.138740
	$(6,1)_3$	$(32,2)_{3^{16}}$	2	1.3726500	2.065650
	$(6,1)_3$	$(64,1)_{3^{16}}$	2	2.4344100	1.954400
C3-H	$(3,1)_3$	$(16,2)_{3^{32}}$	2	0.7272080	0.935283
	$(3,1)_3$	$(32,1)_{3^{32}}$	2	1.2001000	1.107840
	$(11,1)_3$	$(8,2)_{3^{64}}$	2	0.6007660	1.847700
	$(11,1)_3$	$(16,1)_{3^{64}}$	2	1.1126300	1.891190
	$(27,1)_3$	$(4,2)_{3^{128}}$	2	0.5804140	2.261520
	$(27,1)_3$	$(8,1)_{3^{128}}$	2	1.0570600	2.641760
C7-P	$(8,1)_7$	$(16,2)_{7^{16}}$	1	0.5346350	3.126460
	$(8,1)_7$	$(32,1)_{7^{16}}$	1	0.7542980	4.484790

Table 9. Effect of r_{out} for Composite RMFEs, Keeping r_{in} and d' fixed

in C7-P took an average of 0.754 seconds which is almost twice as fast as the adjusted time of $0.610 \cdot 1.97 \approx 1.20$ seconds.

P Set	k	w	r	Max rFIMD Mult	rFIMD Mult (sec)	1 rFIMD Mult (sec)
7-H	214	511	1	3	1.83	0.610
C7-P	$8 \cdot 32 = 256$	$16 \cdot 63 = 1008$	$(1,1)$	1	0.754	0.754

Table 10. Comparing Three-Stage Recode and Direct Recode

6 Conclusion and Future Work

In this work, we present a method that allows small primes to be used with the BGV and BFV FHE schemes without compromising on the amount of data that can be packed into a ciphertext. Specifically, we adapted reverse multiplication friendly embedding (RMFE) to FHE. To that end, we introduced an FHE-specific technique to compute a linear transformation on encoded vectors for free

during the recode process. Additionally, we proposed two extensions to RMFE targeting FHE plaintext spaces with high extension degree fields, namely r-fold RMFE and a three-stage recode process for composite RMFE. r-fold RMFE supports correct decoding of products of up to 2^r encoded vectors at the expense of requiring a higher field degree w for the embedding, capitalizing on the fact that the fixed d of FHE is often too high for small primes to fully utilize for standard RMFEs. Composite RMFEs, on the other hand, let us "split" a large extension field into a tower of smaller fields. This tower of fields is exploited in a three-stage recode, where each stage goes between pairs of fields that are smaller extensions and thereby use operations of lower complexity.

Our experiments show that FIMD multiplication is noisier than FHE multiplications, typically using the capacity of two FHE multiplications. On the other hand, FIMD multiplications have a lower amortized time and need only two ciphertexts to multiply more data. We also find that composite RMFEs, while applicable to high-degree (> 1000) extension fields, are difficult to use in practice. Generating the direct recode map is very time-consuming but the three-stage recode process requires as much capacity as almost 3 FHE multiplications. That said, we approximated the performance of direct recode and found that three-stage recode does improve multiplication times, but significantly increased noise consumption. A middle ground needs to be found for using composite RMFEs, which would entail using "inner" and "outer" RMFEs with some amount of r and r'-fold respectively.

This paper represents the beginning of applying RMFE to FHE, and much work remains to be done. A first direction would be adapting the methods of [13] to FIMD and potentially improve downstream applications of FHE. Another important task is to adapt RMFE for Galois rings, which was explored by Cramer et al. [17] for MPC, to FHE. Crucially, the bootstrapping techniques of Gentry et al. [25], Halevi and Shoup [27] and Chen and Han [9] for BGV and BFV demand plaintext algebras that are Galois rings. Finally, developing RMFEs from other classes of algebraic function fields is necessary to better understand how best to perform homomorphic computation with high-degree extension fields.

References

1. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. J. Mathematical Cryptology **9**(3) (2015)
2. 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. Springer, Heidelberg (2018)
3. Bootland, C., Castryck, W., Iliashenko, I., Vercauteren, F.: Efficiently processing complex-valued data in homomorphic encryption. Journal of Mathematical Cryptology **14**(1), 55–65 (2020)
4. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer, Heidelberg (2012)
5. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: ITCS 2012. pp. 309–325. ACM (2012)

6. Cascudo, I.: On asymptotically good strongly multiplicative linear secret sharing. Ph.D. thesis, Universidad de Oviedo (Jul 2010)
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, Heidelberg (2018)
8. Castryck, W., Iliashenko, I., Vercauteren, F.: Homomorphic $\mathrm{SIM}^2\mathrm{D}$ operations: Single instruction much more data. In: EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 338–359. Springer, Heidelberg (Apr / May 2018)
9. Chen, H., Han, K.: Homomorphic lower digits removal and improved FHE bootstrapping. In: EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 315–337. Springer, Heidelberg (2018)
10. Chen, H., Laine, K., Player, R., Xia, Y.: High-precision arithmetic in homomorphic encryption. In: CT-RSA 2018. LNCS, vol. 10808, pp. 116–136. Springer, Heidelberg (Apr 2018)
11. 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. Springer, Heidelberg (2017)
12. Cheon, J.H., Kim, D., Lee, K.: Mhz2k: Mpc from he over $\mathbb{Z}_{2^k}$ with new packing, simpler reshare, and better zkp. In: CRYPTO 2021, Part II. LNCS, vol. 12826, pp. 426–456. Springer, Heidelberg (Aug 2021)
13. Cheon, J.H., Kim, M., Kim, M.: Search-and-compute on encrypted data. In: FC 2015 Workshops. LNCS, vol. 8976, pp. 142–159. Springer, Heidelberg (2015)
14. 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. Springer, Heidelberg (2016)
15. 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. Springer, Heidelberg (2017)
16. Costache, A., Smart, N.P., Vivek, S., Waller, A.: Fixed-point arithmetic in SHE schemes. In: SAC 2016. LNCS, vol. 10532, pp. 401–422. Springer, Heidelberg (2016)
17. 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, Heidelberg, Virtual Event (Aug 2021)
18. Dalskov, A.P.K., Lee, E., Soria-Vazquez, E.: Circuit amortization friendly encodingsand their application to statistically secure multiparty computation. In: ASIACRYPT 2020, Part III. LNCS, vol. 12493, pp. 213–243. Springer, Heidelberg (Dec 2020)
19. Dowlin, N., Gilad-Bachrach, R., Laine, K., Lauter, K., Naehrig, M., Wernsing, J.: Manual for using homomorphic encryption for bioinformatics. Tech. Rep. MSR-TR-2015-87, Microsoft Research (November 2015), https://www.microsoft.com/en-us/research/publication/manual-for-using-homomorphic-encryption-for-bioinformatics/, accessed 27 September 2021
20. Dowlin, N., Gilad-Bachrach, R., Laine, K., Lauter, K., Naehrig, M., Wernsing, J.: Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. Tech. rep., Microsoft (February 2016)
21. Ducas, L., Micciancio, D.: FHEW: Bootstrapping homomorphic encryption in less than a second. In: EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 617–640. Springer, Heidelberg (2015)
22. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012)

23. van der Geer, G., Howe, E.W., Lauter, K.E., Ritzenthaler, C.: Tables of curves with many points (2009), http://www.manypoints.org, retrieved 20 Sept 2021

24. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing. p. 169–178. STOC '09, Association for Computing Machinery (2009)

25. Gentry, C., Halevi, S., Smart, N.P.: Fully homomorphic encryption with polylog overhead. In: EUROCRYPT 2012. LNCS, vol. 7237, pp. 465–482. Springer, Heidelberg (2012)

26. Goss, D.: Basic Structures of Function Field Arithmetic. Springer, Berlin (1998)

27. Halevi, S., Shoup, V.: Bootstrapping for helib. In: Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I. pp. 641–670 (2015)

28. Halevi, S., Shoup, V.: Faster homomorphic linear transformations in HElib. In: CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 93–120. Springer, Heidelberg (2018)

29. Halevi, S., Shoup, V.: Design and implementation of helib: a homomorphic encryption library. Cryptology ePrint Archive, Report 2020/1481 (2020), https://eprint.iacr.org/2020/1481

30. Iliashenko, I., Zucca, V.: Faster homomorphic comparison operations for BGV and BFV. PoPETs **2021**(3), 246–264 (Jul 2021). https://doi.org/10.2478/popets-2021-0046

31. Jäschke, A., Armknecht, F.: (Finite) field work: Choosing the best encoding of numbers for FHE computation. In: CANS 17. LNCS, vol. 11261, pp. 482–492. Springer, Heidelberg (Nov / Dec 2017)

32. Kim, M., Lee, H.T., Ling, S., Wang, H.: On the efficiency of FHE-based private queries. IEEE Transactions on Dependable and Secure Computing **15**(2), 357–363 (2018)

33. Lattigo v2.2.0. Online: http://github.com/ldsec/lattigo (Jul 2021), ePFL-LDS

34. Orsini, E., Smart, N.P., Vercauteren, F.: Overdrive2k: Efficient secure mpc over $\mathbb{Z}_{2^k}$ from somewhat homomorphic encryption. In: CT-RSA 2020. LNCS, vol. 12006, pp. 254–283. Springer, Heidelberg (Feb 2020)

35. PALISADE lattice cryptography library (release 1.11.5). https://gitlab.com/palisade/palisade-release (Sep 2021), pALISADE Project

36. Microsoft SEAL (release 3.6). https://github.com/Microsoft/SEAL (Nov 2020), microsoft Research, Redmond, WA.

37. Smart, N.P., Vercauteren, F.: Fully homomorphic simd operations. Designs, Codes and Cryptography **71**(1), 57–81 (2014)

38. Stichtenoth, H.: Algebraic Function Fields and Codes. Springer Publishing Company, Incorporated, 2nd edn. (2008)

39. Tan, B.H.M., Lee, H.T., Wang, H., Ren, S.Q., Aung, K.M.M.: Efficient private comparison queries over encrypted databases using fully homomorphic encryption with finite fields. IEEE Transactions on Dependable and Secure Computing pp. 1–1 (2020)

A Complete Experiment Results

A.1 Basic RMFE

P Set	k	r	Max # rFIMD Mult	Max # FHE Mult	rFIMD Mult (sec)	FHE Mult (sec)	Speedup v.s. k FHE Mult
3-P	4	1	3	6	7.84	0.0170	0.00868×
		2	4	5	4.54	0.0337	0.0297×
		4	4	5	1.75	0.0369	0.0845×
		4⋆	4	5	0.0475	0.0370	3.12×
3-E	7	1	3	6	7.16	0.0180	0.0176×
		2	4	6	4.58	0.0276	0.0421×
		4	4	5	1.70	0.0478	0.197×
		4⋆	4	5	0.0469	0.0373	5.57×
3-H	28	1	3	5	3.59	0.0245	0.191×
		2	4	5	2.31	0.0355	0.431×
		4	4	5	0.900	0.0400	1.24×
		4⋆	4	5	0.0541	0.0389	20.1×
7-P	8	1	3	5	3.74	0.0242	0.0516×
		2	4	5	2.40	0.0359	0.120×
		4	4	5	1.16	0.0378	0.261×
		4⋆	4	5	0.0431	0.0404	11.0×
7-E	13	1	3	5	3.71	0.0244	0.0853×
		2	4	5	2.31	0.0359	0.202×
		4	4	5	0.931	0.0384	0.536×
		4⋆	4	5	0.0478	0.0404	20.1×
7-H	214	1	3	5	1.83	0.0238	2.79×
17-P	18	1	2	5	1.16	0.0245	0.380×
		2	4	5	1.12	0.0348	0.558×
		4	4	5	0.428	0.0381	1.61×
17-E	26	1	2	5	1.16	0.0252	0.565×
		2	4	5	0.961	0.0434	1.17×
		4	4	5	0.421	0.0389	2.40×
31-P	32	1	2	4	0.579	0.0231	1.28×
	32	2	3	4	0.384	0.0259	2.16×
	16	4	3	4	0.0262	0.0254	15.5×
31-E	43	1	2	4	0.578	0.0236	1.76×
	43	2	3	4	0.382	0.0251	2.82×
	14	4	3	4	0.0258	0.0263	14.3×

⋆No recodes were performed

Table 11. Comparison of FIMD and FHE multiplication performance for basic RMFEs

A.2 Composite RMFE

P Set	$(k_{\mathsf{in}}, r_{\mathsf{in}})_q$	$(k_{\mathsf{out}}, r_{\mathsf{out}})_{q^{d'}}$	Max rFIMD Mult	Max FHE Mult	rFIMD Mult (sec)	FHE Mult (sec)	Speedup v.s. FHE Mult
C3-P	$(2,2,3)_3$	$(28,3,55)_{3^8}$	3	6	0.7074590	0.0174873	1.384240
	$(4,1,7)_3$	$(64,2,127)_{3^8}$	2	5	1.6718600	0.0129312	1.980060
	$(4,1,7)_3$	$(128,1,255)_{3^8}$	1	5	1.7500500	0.0124286	3.636160
	$(4,2,7)_3$	$(64,1,127)_{3^{16}}$	2	5	1.1955500	0.0126123	2.700630
	$(4,3,7)_3$	$(16,2,31)_{3^{32}}$	3	5	0.1890470	0.0350566	11.86810
	$(4,3,7)_3$	$(32,1,63)_{3^{32}}$	2	5	0.2501270	0.0248627	12.72320
	$(4,4,7)_3$	$(8,2,15)_{3^{64}}$	3	5	0.0966742	0.0340547	11.272400
	$(4,4,7)_3$	$(16,1,31)_{3^{64}}$	2	5	0.1409130	0.0233469	10.603700
	$(4,6,7)_3$	$(2,2,3)_{3^{256}}$	4	5	0.0882771	0.0365346	3.310900
	$(4,6,7)_3$	$(4,1,7)_{3^{256}}$	3	6	0.1059930	0.0175280	2.645920
C3-E	$(2,2,7)_3$	$(32,2,63)_{3^{16}}$	3	6	0.8984420	0.0170243	1.212720
	$(2,2,7)_3$	$(64,1,127)_{3^{16}}$	2	5	1.4284200	0.0127079	1.138740
	$(6,1,15)_3$	$(32,2,63)_{3^{16}}$	2	5	1.3726500	0.0147678	2.065650
	$(6,1,15)_3$	$(64,1,127)_{3^{16}}$	2	6	2.4344100	0.0123901	1.954400
	$(6,2,15)_3$	$(16,2,31)_{3^{32}}$	3	5	0.8059610	0.0338111	4.027320
	$(6,2,15)_3$	$(32,1,63)_{3^{32}}$	2	5	1.0435400	0.0129588	2.384290
	$(6,3,15)_3$	$(8,2,15)_{3^{64}}$	3	6	0.5079130	0.0164912	1.558490
	$(6,3,15)_3$	$(16,1,31)_{3^{64}}$	2	6	0.1632760	0.0168873	9.929100
	$(6,4,15)_3$	$(4,2,7)_{3^{128}}$	3	5	0.0692145	0.0332331	11.523500
	$(6,4,15)_3$	$(8,1,15)_{3^{128}}$	2	5	0.0893559	0.0248044	13.324400
C3-H	$(3,1,17)_3$	$(16,2,31)_{3^{32}}$	2	6	0.7272080	0.0141697	0.935283
	$(3,1,17)_3$	$(32,1,63)_{3^{32}}$	2	6	1.2001000	0.0138491	1.107840
	$(3,2,17)_3$	$(8,2,15)_{3^{64}}$	3	6	0.4109670	0.0276301	1.613570
	$(3,2,17)_3$	$(16,1,31)_{3^{64}}$	2	5	0.4838210	0.0119909	1.189630
	$(11,1,33)_3$	$(8,2,15)_{3^{64}}$	2	6	0.6007660	0.0126140	1.847700
	$(11,1,33)_3$	$(16,1,31)_{3^{64}}$	2	6	1.1126300	0.119557	1.891190
	$(11,2,33)_3$	$(4,2,7)_{3^{128}}$	3	6	0.3776890	0.0258704	3.013850
	$(11,2,33)_3$	$(8,1,15)_{3^{128}}$	2	5	0.4557780	0.117630	2.271160
	$(27,1,65)_3$	$(4,2,7)_{3^{128}}$	2	5	0.5804140	0.0121539	2.261520
	$(27,1,65)_3$	$(8,1,15)_{3^{128}}$	2	5	1.0570600	0.0129282	2.641760
C7-P	$(4,2,7)_7$	$(16,2,31)_{7^{16}}$	3	5	0.4663130	0.0247454	3.396230
	$(4,2,7)_7$	$(32,1,63)_{7^{16}}$	2	5	0.5249380	0.0129875	3.166860
	$(8,1,15)_7$	$(16,2,31)_{7^{16}}$	1	5	0.5346350	0.0130587	3.126460
	$(8,1,15)_7$	$(32,1,63)_{7^{16}}$	1	5	0.7542980	0.0132143	4.484790
	$(8,2,15)_7$	$(8,2,15)_{7^{32}}$	3	5	0.4076150	0.0237835	3.734270
	$(8,2,15)_7$	$(16,1,31)_{7^{32}}$	2	5	0.5056360	0.0129546	3.279410
	$(8,3,15)_7$	$(4,2,7)_{7^{64}}$	3	5	0.2652440	0.0235409	2.840060
	$(8,3,15)_7$	$(8,1,15)_{7^{64}}$	2	5	0.0908312	0.0126516	8.914370
C7-E	$(6,2,15)_7$	$(8,2,15)_{7^{32}}$	3	5	0.4115960	0.0246697	2.876960
	$(6,2,15)_7$	$(16,1,31)_{7^{32}}$	2	6	0.5140520	0.0129425	2.417030
	$(13,1,29)_7$	$(8,2,15)_{7^{32}}$	2	5	0.5879160	0.0124884	2.209150
	$(13,1,29)_7$	$(16,1,31)_{7^{32}}$	1	5	0.7183080	0.0126397	3.660070
	$(13,3,29)_7$	$(2,2,3)_{7^{128}}$	3	5	0.2556120	0.0242098	2.462540
	$(13,3,29)_7$	$(4,1,7)_{7^{128}}$	2	5	0.0623321	0.0255382	21.305100
C7-H	$(32,1,147)_7$	$(2,1,3)_{7^{256}}$	2	6	0.5453260	0.0135578	1.591150

Table 12. Comparison of FIMD and FHE multiplication performance for composite RMFEs, with the three-stage recode process of Section 3.3