Somewhat Homomorphic Encryption based on Random Codes

Carlos Aguilar-Melchor[1], Victor Dyseryn[2], Philippe Gaborit[2]

[1] Sandbox AQ
[2] XLIM, Université de Limoges, France

Abstract. We present a secret-key encryption scheme based on random rank metric ideal linear codes with a simple decryption circuit. It supports unlimited homomorphic additions and plaintext absorptions as well as a fixed arbitrary number of homomorphic multiplications.

We study a candidate bootstrapping algorithm that requires no multiplication but additions and plaintext absorptions only. This latter operation is therefore very efficient in our scheme, whereas bootstrapping is usually the main reason which penalizes the performance of other fully homomorphic encryption schemes. However, the security reduction of our scheme restricts the number of independent ciphertexts that can be published. In particular, this prevents to securely evaluate the bootstrapping algorithm as the number of ciphertexts in the key switching material is too large.

Our scheme is nonetheless the first somewhat homomorphic encryption scheme based on random ideal codes and a first step towards full homomorphism. Random ideal codes give stronger security guarantees as opposed to existing constructions based on highly structured codes. We give concrete parameters for our scheme that shows that it achieves competitive sizes and performance, with a key size of 3.7 kB and a ciphertext size of 0.9 kB when a single multiplication is allowed.

1 Introduction

Homomorphic encryption. A homomorphic encryption scheme allows to perform operations on plaintexts which are still in their encrypted form. Because it enables computations in a public cloud while keeping the data private, homomorphic encryption has numerous applications, especially in the medical or banking sector.

In the early years of homomorphic encryption after it was introduced by Rivest, Adleman and Dertouzos in 1978 [28], some schemes were designed [24, 19, 27] but they only supported a single type of operation, either an addition or a multiplication. The scheme by Boneh, Goh and Nissim [13] was the first to support unlimited additions and a single multiplication. A long-standing problem was finally solved when Gentry designed in 2009 a fully homomorphic encryption (FHE) scheme [23] able to perform unlimited additions and multiplications on encrypted data.

Many improvements have been made [14, 18, 15] since the initial proposal by Gentry in order to make fully homomorphic encryption efficient. These efficient schemes were built with structured lattices, making them all highly at risk should an attack be found on structured lattice difficult problems.

Bootstrapping. A fundamental technique in achieving fully homomorphic encryption is called bootstrapping [23]. In most systems, after some homomorphic operations, the ciphertext suffers from a large amount of noise that prevent any further operation. The bootstrapping technique consists in homomorphically applying the decryption circuit, generating a new ciphertext under a new key with a reduced amount of noise. This allows to continue with additional homomorphic operations. However, in existing systems, the bootstrapping procedure is very costly, making homomorphic encryption inpractical for generic applications.

Homomorphic encryption based on codes. The question of homomorphic encryption based on codes was first addressed in 2011: the authors of [9] present a symmetric scheme supporting additions and a limited number of multiplications. Their construction relies on a class of codes called *special evaluation codes* whose codewords have natural multiplicative properties. They instantiate their scheme with Reed-Muller codes. They do not investigate bootstrapping further than showing its impossibility.

In the same year, a public-key homomorphic encryption scheme based on Reed-Solomon codes was proposed [12] but was broken shortly after its publication [22].

More recently, homomorphic computations in Reed-Muller codes were investigated [16]. The authors present an operation on Reed-Muller codewords so that the result represents an encoding of the multiplication of the messages. However, they did not study their techniques in the presence of noise, nor did they propose an encryption scheme. Therefore their work is not related to any notion of cryptographic security.

All existing code-based homomorphic constructions thus rely on highly structured codes, which turned out in the past to be a source of numerous attacks [30, 26].

Rank metric. Rank metric is an alternative to the usual Hamming metric in coding theory. In rank metric, vectors in the word space can be seen as matrices and their weight is defined as the rank of that matrix. Rank metric encryption [7, 4, 1, 6, 2] and signature [5, 11] schemes have been proposed and in several cases acheive better performance than Hamming metric cryptosystems. No other advanced primitives were designed in the rank metric, other than an identity-based encryption based on rank metric codes [20] that was broken shortly after [17]. Contrary to the Hamming metric, for a given support, in rank metric the number of possible error vectors depends not only on the size of the field but also on the length of the code. This additionnal degree of liberty could open a possibility for an efficient fully homomorphic encryption scheme.

Our contribution. We propose the first code-based somewhat homomorphic encryption scheme relying on random ideal codes. It has therefore a stronger security reduction than existing approaches based on highly structured codes. Our construction is symmetric and supports addition, multiplication and plaintext absorption.

We also propose the first candidate bootstrapping algorithm for a code-based homomorphic scheme that homomorphically decrypts ciphertexts produced from another secret key. Remarkable for its simplicity, our algorithm enjoys no multiplicative depth, as it requires additions and plaintext multiplications only.

However, our scheme suffers from two major limitations that hamper its categorization as fully homomorphic. First, the number of multiplications is limited because each operation increases the length of the ciphertext as well as the dimension of the noise space. Second, and most importantly, there is an upper bound to the number of independent ciphertexts that can be published without a polynomial key recovery attack. In particular, the number of ciphertexts required for our bootstrapping algorithm is larger than the maximal number of publishable independent ciphertexts. To address these problems, we propose a refinement of our homomorphic decryption algorithm by introducing the notion of ciphertext packing. It reduces the number of bootstrapping ciphertexts very close yet still above the maximal limit.

Still, we give concrete parameters for our scheme that shows its efficiency as a somewhat homomorphic encryption scheme and a strong potential to be refined into a FHE scheme. For a single multiplication, the key size is 3.7 kB and the ciphertext size is only 0.9 kB, with competitive running times estimated to be a few microseconds for addition and 0.5 millisecond for multiplication. Other parameters could be found to support an arbitrary fixed number of multiplications.

Overview of our construction. Our scheme is an Aleknovich-inspired [3] construction. It can be seen as a secret key version of the NIST Round 2 candidate RQC [1], with important differences (see below). The ciphertext is a pair $(\boldsymbol{u}, \boldsymbol{v})$ of vectors in $\mathbb{F}_{q^m}^n$ where $\boldsymbol{v}$ is the noisy version of the multiplication $\boldsymbol{u} \cdot \boldsymbol{s}$ of the first component of the ciphertext times the secret key $\boldsymbol{s}$. The noise is taken in a secret space and contains an encoding of the message $\boldsymbol{m}$ being a vector in $\mathbb{F}_q^n$. The ciphertext space enjoys an $\mathbb{F}_q^n$-module structure which makes addition and plaintext multiplication completely straightforward.

Contrary to RQC in which the encoded message is a codeword of a public Gabidulin code that can be recovered from the noise using a decoding algorithm, in our construction the message is encoded into a vector space orthogonal to the error vector. The decryption algorithm is thus quite simple since it consists in a secret orthogonal projection of the noise term (i.e. a scalar product with a secret basis). Consequently, a natural homomorphic decryption algorithm can be designed. The key switching material consists in encrypted coordinates of the previous key and projection vector, split against a public basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. By splitting the ciphertext onto the same basis and plaintext multplying each component to the key switching material, one obtains a fresh ciphertext under a new key with no multiplication, only with additions and plaintext multiplications.

The security of our scheme can be reduced to the well-studied ideal rank syndrome decoding problem (IRSD). The rate of the code tends towards 0 as the number of independent ciphertexts increases, giving an upper bound of $2w$ (where w is the rank weight of the error term in the ciphertext) to the number of ciphertexts than can be safely published. This prevents from using safely the homomorphic decryption algorithm which requires $2m$ ciphertexts: one for each of the m components against the $\mathbb{F}_{q^m}$-basis of the secret key and the projection vector.

Aiming at reducing the number of ciphertexts necessary to a homomorphic decryption, we finally present a way to pack several plaintexts in a single ciphertext. Instead of having the message encoded in a single dimension orthogonal to the error, the idea is to increase the dimension of the encoded message, which is now a matrix with components in $\mathbb{F}_q$. Rows of the matrix can be rotated using a public operation that allows to perform homomorphic linear combinations on the rows of the plaintext matrix. Because the ciphertext now contains more information, the size of the bootstrapping material is reduced to $2(w+1)$. However, this is still higher than the secure upper bound $2w$.

Outline. This paper is organized as follows: Section 2 contains basic definitions. Section 3 presents our encryption scheme and homomorphic addition. Section 4 adds a homomorphic multiplication and builds our somewhat homomorphic encryption scheme. Section 5 defines our homomorphic decryption algorithms. The security of the scheme is studied in Section 6. The idea of packing is presented in Section 7 in an attempt to reduce the number of ciphertexts required for bootstrapping. Finally, concrete parameters for our scheme are presented in Section 8.

2 Preliminaries

In this section we first present the rings, fields and vector spaces we will work with as well as an associated metric, and then we will define error-correcting codes associated with this metric.

2.1 Basic notations

For a finite set S, $x \overset{\$}{\leftarrow} S$ corresponds to a uniform sampling from the set S. For a probabilistic algorithm $\mathrm{Alg}()$, $x \overset{\$}{\leftarrow} \mathrm{Alg}()$ corresponds to sampling following the algorithm's output distribution. We also use the notation $x \in \{\mathrm{Alg}()\}$ to indicate that x is in the set of potential outputs of Alg.

Vectors will be represented with lowercase bold letters and matrices with uppercase bold letters. Vectors are assumed to be row vectors unless stated otherwise. For a field $\mathbb{F}$, $\mathcal{M}_{n,m}(\mathbb{F})$ represents the set of matrices with n rows and m columns of elements in $\mathbb{F}$. When n equals m this set, together with classical matrix sum and product, forms a ring that we denote $\mathcal{M}_n(\mathbb{F})$. For a given vector $\boldsymbol{b}$, we will note $\boldsymbol{b}^{(i)}$ its i-th coordinate, and for a given matrix $\boldsymbol{B}$, we will note $\boldsymbol{B}^{(i)}$ its i-th *column* vector. Finally, for $\boldsymbol{b}$ a vector over a field F, and an element $k \in \mathbb{F}$, we note the usual scalar multiplication $k \star \boldsymbol{b}$.

2.2 Finite Fields

In the following we let q be a prime power and m, n two positive integers. We will work with the finite fields of order q, and q^m: $\mathbb{F}_q, \mathbb{F}_{q^m}$. Of course there are multiple isomorphic fields of a given order, with multiple representations and leading to different calculation algorithms.

Finite field $\mathbb{F}_q$. Generally, q will be a prime and therefore elements and computations in $\mathbb{F}_q$ are associated with elements and computations in the modular ring $\mathbb{Z}/q\mathbb{Z}$. But it is important to notice that this is not mandatory and one may perfectly choose q to be a prime power and define a proper representation and calculation rules for this setting.

Extension of finite field $\mathbb{F}_{q^m}$. As usual, elements in extensions of the base field $\mathbb{F}_q$ will be represented using quotients over the polynomial ring $\mathbb{F}_q[X]$. Thus elements and computations in $\mathbb{F}_{q^m}$ are associated with polynomial representations and computations over $\mathbb{F}_q[X]/\langle P\rangle$ for $P \in \mathbb{F}_q[X]$ an irreducible monic polynomial of degree m. We fix such a polynomial P for the rest of the article.

Element-vector transformation. An element $f \in \mathbb{F}_{q^m}$ can be associated to a *column* vector of $\mathbb{F}_q^m$ using the coefficients of the polynomial representation of f. It is obviously an $\mathbb{F}_q$-vector space isomorphism that we denote $\mathbf{vec}()$:

$$\begin{aligned} \mathbf{vec} : \mathbb{F}_{q^m} &\longrightarrow \mathbb{F}_q^m \\ f = \sum_{i=0}^{m-1} f_i X^i &\longmapsto \begin{pmatrix} f_1 \\ \vdots \\ f_m \end{pmatrix} \end{aligned}$$

Vector-matrix transformation. Similarly, a vector $\boldsymbol{v} \in \mathbb{F}_{q^m}^n$ can be associated to an $m \times n$ matrix $\mathbf{Mat}(\boldsymbol{v}) \in \mathcal{M}_{m,n}(\mathbb{F}_q)$ whose i-th column is $\mathbf{vec}(\boldsymbol{v}^{(i)})$. (The bold capital 'M' in this transformation's name being a reminder that it outputs a matrix).

2.3 Ideal matrices and vector-vector product

We would like to give a field structure to the set of vectors in $\mathbb{F}_{q^m}^n$. To do so, we associate a vector $\boldsymbol{v}$ of $\mathbb{F}_{q^m}^n$ to a polynomial $\text{poly}(\boldsymbol{v})$ in the ideal ring $\mathbb{F}_{q^m}[X]/\langle Q\rangle$ for $Q \in \mathbb{F}_{q^m}[X]$ a monic irreducible polynomial of degree n whose coefficients are in the subfield $\mathbb{F}_q$[1]. We fix such a polynomial Q for the rest of the article. This transformation is an isopmorphism and can be seen as the equivalent to $\mathbf{vec}^{-1}$ (with the difference that poly operates on *row* vectors of length n).

We can now define the multiplication of two vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{F}_{q^m}^n$ as

$$\boldsymbol{u} \cdot \boldsymbol{v} = \text{poly}^{-1}(\text{poly}(\boldsymbol{u})\text{poly}(\boldsymbol{v})).$$

Note that the product of polynomials is calculated *modulo Q*.

We can alternatively define the product of two vectors as a matrix-vector product thanks to the definition of ideal matrices.

Definition 1 (Ideal Matrices). *Let $Q \in \mathbb{F}_{q^m}[X]$ be a polynomial of degree n and $\boldsymbol{v} \in \mathbb{F}_{q^m}^n$. The ideal matrix generated by $\boldsymbol{v}$ modulo Q is the matrix denoted*

[1] This is to ensure $\mathbb{F}_q^n$ is stable under multiplication.

$\mathcal{IM}_Q(\boldsymbol{v}) \in \mathcal{M}_n(\mathbb{F}_{q^m})$ *of the form:*

$$\mathcal{IM}_Q(\boldsymbol{v}) = \begin{pmatrix} \boldsymbol{v} \\ X\text{poly}(\boldsymbol{v}) \\ \vdots \\ X^{n-1}\text{poly}(\boldsymbol{v}) \end{pmatrix}.$$

The products of polynomials are also computed *modulo* Q and the transformation poly^{-1} has been omitted for simplicity.

The multiplication of two vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{F}_{q^m}^n$ can be then computed with

$$\boldsymbol{u} \cdot \boldsymbol{v} = \boldsymbol{u}\, \mathcal{IM}_Q(\boldsymbol{v}).$$

2.4 Metric and support

Definition 2 (Rank metric over $\mathbb{F}_{q^m}^n$). *Let $\boldsymbol{e} = (e_1, \dots, e_n)$ be an element of $\mathbb{F}_{q^m}^n$. The rank weight of $\boldsymbol{e}$, denoted by $\mathsf{rw}(\boldsymbol{e})$, is defined as*

$$\mathsf{rw}(\boldsymbol{e}) = \mathsf{rank}(\mathbf{Mat}(\boldsymbol{e})).$$

Note that the rank weight is independent from the choice of the irreducible monic polynomial P of degree m that was chosen to define $\mathbf{Mat}()$. The rank distance between two vectors $\boldsymbol{e}, \boldsymbol{f} \in \mathbb{F}_{q^m}^n$ is defined by $\mathsf{rw}(\boldsymbol{e} - \boldsymbol{f})$.

For $\boldsymbol{e} = (e_1, \dots, e_n) \in \mathbb{F}_{q^m}^n$, the *support* E of $\boldsymbol{e}$, denoted $\mathsf{supp}(\boldsymbol{e})$, is the $\mathbb{F}_q$-linear subspace of $\mathbb{F}_{q^m}$ generated by the coordinates of $\boldsymbol{e}$:

$$E = \langle e_1, \dots, e_n \rangle_{\mathbb{F}_q}.$$

Note that $\mathsf{dim}(E) = \mathsf{rw}(\boldsymbol{e})$.

$\mathcal{S}_w^n(\mathbb{F}_{q^m})$ stands for the set of vectors of length n and rank weight w over $\mathbb{F}_{q^m}$:

$$\mathcal{S}_w^n(\mathbb{F}_{q^m}) = \{\boldsymbol{e} \in \mathbb{F}_{q^m}^n : \mathsf{rw}(\boldsymbol{e}) = w\}$$

2.5 Codes and ideal codes

Definition 3 ($\mathbb{F}_{q^m}$-linear code). *An $\mathbb{F}_{q^m}$-linear code $\mathcal{C}$ of dimension k and length n is a subspace of dimension k of $\mathbb{F}_{q^m}^n$ seen as an $\mathbb{F}_{q^m}$-linear space. The notation $[n, k]_{q^m}$ is used to denote its parameters.*

The code $\mathcal{C}$ can be represented by two equivalent ways:

- *by a generator matrix $\boldsymbol{G} \in \mathbb{F}_{q^m}^{k \times n}$. Each row of $\boldsymbol{G}$ is an element of a basis of $\mathcal{C}$,*

$$\mathcal{C} = \{\boldsymbol{x}\boldsymbol{G}, \boldsymbol{x} \in \mathbb{F}_{q^m}^k\}.$$

- *by a parity-check matrix $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k) \times n}$. Each row of $\boldsymbol{H}$ determines a parity-check equation verified by the elements of $\mathcal{C}$:*

$$\mathcal{C} = \{\boldsymbol{x} \in \mathbb{F}_{q^m}^n : \boldsymbol{H}\boldsymbol{x}^T = \boldsymbol{0}\}.$$

We say that $\boldsymbol{G}$ (respectively $\boldsymbol{H}$) is under systematic form if and only if it is of the form $(\boldsymbol{I}_k|\boldsymbol{A})$ (respectively $(\boldsymbol{I}_{n-k}|\boldsymbol{B})$).

To describe an $[n,k]_{q^m}$ linear code, we can give a systematic generator matrix or a systematic parity-check matrix. In both cases, the number of bits needed to represent such a matrix is $k(n-k)m\lceil\log_2 q\rceil$. To reduce the size of a representation of a code, we introduce ideal codes. They are a generalization of double circulant codes by choosing a polynomial P to define the quotient-ring $\mathbb{F}_{q^m}[X]/(P)$. More details about this construction can be found in [7].

Definition 4 (Ideal codes). *Let $Q \in \mathbb{F}_q[X]$ be a polynomial of degree n. For $s \geq 2$, an s-ideal code is an $[sn,n]_{q^m}$-code $\mathcal{C}$ with a generator matrix of the form $\boldsymbol{G} = \begin{pmatrix}\boldsymbol{I}_n & \mathcal{IM}_Q(\boldsymbol{g}_1) & \dots & \mathcal{IM}_Q(\boldsymbol{g}_{s-1})\end{pmatrix} \in \mathbb{F}_{q^m}^{n\times ns}$, where $\boldsymbol{g}_i \in \mathbb{F}_{q^m}^n$ for $1 \leq i \leq s-1$. The vectors $(\boldsymbol{g}_1,\cdots,\boldsymbol{g}_{s-1})$ are said to be the generators of the ideal code.*
Similarly, $\mathcal{C}$ is an ideal code if it admits a parity-check matrix of the form

$$\boldsymbol{H} = \begin{pmatrix} & \mathcal{IM}_Q(\boldsymbol{h}_1)^\top \\ \boldsymbol{I}_{n(s-1)} & \vdots \\ & \mathcal{IM}_Q(\boldsymbol{h}_{s-1})^\top \end{pmatrix} \in \mathbb{F}_{q^m}^{n(s-1)\times ns}$$

where $\boldsymbol{h}_i \in \mathbb{F}_{q^m}^n$ for $1 \leq i \leq s-1$.

2.6 Difficult Problems for Cryptography

Rank metric code-based cryptography relies on the rank syndrome decoding problem (RSD) and variants.

Definition 5 (Search RSD problem). *On input $(\mathbf{H},\mathbf{s}^\top) \in \mathbb{F}_{q^m}^{(n-k)\times n} \times \mathbb{F}_{q^m}^{(n-k)}$ from the RSD distribution, the* syndrome decoding problem *$\mathsf{RSD}_{n,k,w}$ asks to find $\mathbf{x} \in \mathcal{S}_w^n(\mathbb{F}_{q^m})$ such that $\mathbf{H}\mathbf{x}^\top = \mathbf{s}^\top$.*

The RSD problem has a decision version, which asks to decide whether the given sample came from the RSD distribution or uniform distribution:

Definition 6 (RSD Distribution). *Let $n,k,w \in \mathbb{N}^*$, the $\mathsf{RSD}_{n,k,w}$ Distribution chooses $\mathbf{H} \stackrel{\$}{\leftarrow} \mathbb{F}_{q^m}^{(n-k)\times n}$ and $\mathbf{x} \stackrel{\$}{\leftarrow} \mathcal{S}_w^n(\mathbb{F}_{q^m})$, and outputs $(\mathbf{H},\mathbf{H}\mathbf{x}^\top)$.*

Definition 7 (Decision RSD problem). *Given $(\mathbf{H},\mathbf{s}^\top) \stackrel{\$}{\leftarrow} \mathbb{F}_{q^m}^{(n-k)\times n} \times \mathbb{F}_{q^m}^{(n-k)}$, the decision RSD problem $\mathsf{DRSD}_{n,k,w}$ asks to decide with non-negligible advantage whether $(\mathbf{H},\mathbf{s}^\top)$ came from the $\mathsf{RSD}_{n,k,w}$ distribution or the uniform distribution over $\mathbb{F}_{q^m}^{(n-k)\times n} \times \mathbb{F}_{q^m}^{(n-k)}$.*

In order to propose reasonable key sizes, we base our proposition on s-ideal codes. We adapt the previous problems to this configuration.

Definition 8 (Search IRSD problem). *For positive integers n, w, s, a random parity check matrix $\mathbf{H}$ of an s-ideal code $\mathcal{C}$ and $\mathbf{y} \stackrel{\$}{\leftarrow} \mathbb{F}_{q^m}^{sn-n}$, the* Search s-ideal RSD problem $\mathsf{IRSD}_{n,s,w}$ *asks to find $\mathbf{x} = (\mathbf{x}_1,\dots,\mathbf{x}_s) \in \mathbb{F}_{q^m}^{sn}$ such that $\omega(\mathbf{x}) = w$, and $\mathbf{y} = \mathbf{x}\mathbf{H}^\top$.*

Although there is no general complexity result for ideal codes, decoding these codes is considered hard by the community. There exist general attacks which use the cyclic structure of the code [29, 25] but these attacks have only a very limited impact on the practical complexity of the problem. The conclusion is that in practice, the best attacks are the same as those for non-ideal codes up to a small factor. IRSD is also at the core of the security of other encryption schemes such as NIST Round 2 candidates ROLLO [4] or RQC [1].

We also define the decision version of the IRSD problem:

Definition 9 (IRSD Distribution). *For positive integers n, w and s, the IRSD$_{n,s,w}$ distribution chooses uniformly at random a parity check matrix $\mathbf{H} \overset{\$}{\leftarrow} \mathbb{F}_{q^m}^{(sn-n)\times sn}$ of an s-ideal code $\mathcal{C}$ (see definition 4) together with a vector $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_s) \overset{\$}{\leftarrow} \mathbb{F}_{q^m}^{sn}$ such that $\omega(\boldsymbol{x}) = w$, and outputs $(\mathbf{H}, \mathbf{H}\mathbf{x}^\top)$.*

Definition 10 (Decision IRSD problem). *For positive integers n, w, s, a bit b, a random parity check matrix $\mathbf{H}$ of an s-ideal code $\mathcal{C}$ and $\mathbf{y} \overset{\$}{\leftarrow} \mathbb{F}_{q^m}^{sn-n}$, the* decision s-ideal RSD problem DIRSD$_{n,s,w}$ *asks to decide with non-negligible advantage whether $(\mathbf{H}, \mathbf{y}^\top)$ came from the IRSD$_{n,s,w}$ distribution or the uniform distribution over vectors of parity b in $\mathbb{F}_{q^m}^{(sn-n)\times sn} \times \mathbb{F}_{q^m}^{(sn-n)}$.*

3 Additive Scheme

In this section we present an additive secret key encryption scheme. It can also multiplicatively absorb a plaintext.

3.1 Fundamental Algorithms

The three polynomial-time algorithms constituting our the additive scheme AHE (for Additively-Homomorphic Encryption) are depicted in Fig. 1. The scheme is parametrized by:

- q, the base field cardinality;
- m, the dimension of the field extension;
- n, the length of the vectors;
- w, the rank weight of the error; it must be that $w < m$.

Remark 1. Encrypt$_{\mathsf{AHE}}$ and Decrypt$_{\mathsf{AHE}}$ are functions, not randomized algorithms. In general we will hide the randomness in the encryption function: Encrypt$_{\mathsf{AHE}}(\boldsymbol{sk}, \boldsymbol{m})$ being the randomized algorithm that samples $\boldsymbol{r} \overset{\$}{\leftarrow} \mathbb{F}_{q^m}^n \times \mathcal{M}_{w,n}(\mathbb{F}_q)$ and returns Encrypt$_{\mathsf{AHE}}(\boldsymbol{sk}, \boldsymbol{m}, \boldsymbol{r})$.

Remark 2. In Encrypt$_{\mathsf{AHE}}$, having $\boldsymbol{e} = \boldsymbol{f}\boldsymbol{R}_2$ with $\boldsymbol{R}_2 \overset{\$}{\leftarrow} \mathcal{M}_{w,n}(\mathbb{F}_q)$ is equivalent to having $\boldsymbol{e} \overset{\$}{\leftarrow} F^n$. As for $\hat{\boldsymbol{m}}$, it belongs to $(\langle \boldsymbol{g}^{(1)} \rangle_{\mathbb{F}_q})^n$.

- $\mathsf{KeyGen}_{\mathsf{AHE}}()$:
  - samples $\boldsymbol{f} = (f_1, \ldots, f_w) \overset{\$}{\leftarrow} \mathcal{S}_w^w(\mathbb{F}_{q^m})$
  - extends $\boldsymbol{f}$ into a basis $\boldsymbol{b} = (f_1, \ldots, f_w, g_1, \ldots, g_{m-w}) \in \mathcal{S}_m^m(\mathbb{F}_{q^m})$
  - defines $\boldsymbol{g} = (g_1, \ldots, g_{m-w})$
  - computes the matrix $\boldsymbol{B} = \mathbf{Mat}(\boldsymbol{b})$
  - defines $\boldsymbol{D}$ as the last $m - w$ columns of its transposed inverse $(\boldsymbol{B}^{-1})^T$
  - samples $\boldsymbol{s} \overset{\$}{\leftarrow} F^n$ with $F = \mathsf{supp}(\boldsymbol{f})$
  - returns $\boldsymbol{sk} = (\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{D}, \boldsymbol{s})$.
- $\mathsf{Encrypt}_{\mathsf{AHE}}(\boldsymbol{sk} = (\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{D}, \boldsymbol{s}), \boldsymbol{m} \in \mathbb{F}_q^n, \boldsymbol{r} \in \mathbb{F}_{q^m}^n \times \mathcal{M}_{w,n}(\mathbb{F}_q))$: notes $(\boldsymbol{r}_1, \boldsymbol{R}_2) = \boldsymbol{r}$, defines $\boldsymbol{u} = \boldsymbol{r}_1$, $\boldsymbol{e} = \boldsymbol{f}\boldsymbol{R}_2$ and sets $\boldsymbol{v} = \boldsymbol{s} \cdot \boldsymbol{u} + \boldsymbol{e} + \hat{\boldsymbol{m}}$ with $\hat{\boldsymbol{m}} = \boldsymbol{g}^{(1)} \star \boldsymbol{m} \in \mathbb{F}_{q^m}^n$. Returns $\boldsymbol{ct} = (\boldsymbol{u}, \boldsymbol{v})$.
- $\mathsf{Decrypt}_{\mathsf{AHE}}(\boldsymbol{sk} = (\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{D}, \boldsymbol{s}), \boldsymbol{ct} = (\boldsymbol{u}, \boldsymbol{v}))$: returns $\boldsymbol{d}^T \mathbf{Mat}(\boldsymbol{v} - \boldsymbol{s} \cdot \boldsymbol{u})$ with $\boldsymbol{d} = \boldsymbol{D}^{(1)}$.

Fig. 1. Description of the additive scheme.

Proposition 1 (Fresh Ciphertext Decryption Correctness). *For* $\boldsymbol{sk} \overset{\$}{\leftarrow} \mathsf{KeyGen}_{\mathsf{AHE}}()$*,* $\boldsymbol{m} \in \mathbb{F}_q^n$ *and* $\boldsymbol{r} \in \mathbb{F}_{q^m}^n \times \mathcal{M}_{w,n}(\mathbb{F}_q)$ *it holds that*

$$\mathsf{Decrypt}_{\mathsf{AHE}}\left(\boldsymbol{sk}, \mathsf{Encrypt}_{\mathsf{AHE}}\left(\boldsymbol{sk}, \boldsymbol{m}, \boldsymbol{r}\right)\right) = \boldsymbol{m}.$$

Proof. We suppose in this proof that the $\mathsf{KeyGen}_{\mathsf{AHE}}$ and $\mathsf{Encrypt}_{\mathsf{AHE}}$ protocols are well defined and can be executed properly. We just note that $\boldsymbol{B}^{-1}$ exists as $\boldsymbol{b}$ is a basis of $\mathbb{F}_{q^m}$ and thus $\boldsymbol{B}$ is of full rank.

As $\boldsymbol{d}$ is the $(w+1)$-th column vector of $(\boldsymbol{B}^{-1})^T$, we thus have that $\boldsymbol{d}^T \mathbf{vec}(\boldsymbol{g}^{(1)}) = 1 \in \mathbb{F}_q$ and $\boldsymbol{d}^T \mathbf{vec}(x) = 0$ for any $x \in \boldsymbol{b} \setminus \{\boldsymbol{g}^{(1)}\}$.

The correctness is thus straightforward as, noting $(\boldsymbol{u}, \boldsymbol{v}) = \mathsf{Encrypt}_{\mathsf{AHE}}(\boldsymbol{sk}, \boldsymbol{m}, \boldsymbol{r})$, from $\boldsymbol{v} = \boldsymbol{s} \cdot \boldsymbol{u} + \boldsymbol{e} + \hat{\boldsymbol{m}}$ we get that $\boldsymbol{v} - \boldsymbol{s} \cdot \boldsymbol{u} = \boldsymbol{e} + \hat{\boldsymbol{m}}$ and therefore

$$\boldsymbol{v} - \boldsymbol{s} \cdot \boldsymbol{u} = \sum_{1 \leq i \leq w} f_i \star \boldsymbol{e}_i + \boldsymbol{g}^{(1)} \star \boldsymbol{m}.$$

with $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_w, \boldsymbol{m} \in \mathbb{F}_q^n$. We can thus write,

$$\begin{aligned} \boldsymbol{d}^T \mathbf{Mat}(\boldsymbol{v} - \boldsymbol{s} \cdot \boldsymbol{u}) &= \sum_{1 \leq i \leq w} \boldsymbol{d}^T \mathbf{vec}(f_i) \star \boldsymbol{e}_i + \boldsymbol{d}^T \mathbf{vec}(\boldsymbol{g}^{(1)}) \star \boldsymbol{m} \\ &= \boldsymbol{m} \end{aligned}$$

using the fact that, as noted above, $\boldsymbol{d}^T \mathbf{vec}(\boldsymbol{g}^{(1)}) = 1 \in \mathbb{F}_q$ and $\boldsymbol{d}^T \mathbf{vec}(x) = 0$ for any $x \in \boldsymbol{b} \setminus \{\boldsymbol{g}^{(1)}\}$.

□

Proposition 2 (Ciphertext Distribution). *For* $\boldsymbol{sk} \overset{\$}{\leftarrow} \mathsf{KeyGen}_{\mathsf{AHE}}()$*, the set of ciphertexts of zero* $\{\mathsf{Encrypt}_{\mathsf{AHE}}(\boldsymbol{sk}, \boldsymbol{0})\}$ *is a subgroup of* $\mathbb{F}_{q^m}^n \times \mathbb{F}_{q^m}^n$*, and more generally the set of ciphertexts of a message* $\boldsymbol{m} \in \mathbb{F}_q^n$ *are the cosets* $\{\mathsf{Encrypt}_{\mathsf{AHE}}(\boldsymbol{sk}, \boldsymbol{0})\} + \boldsymbol{g}^{(1)} \star \boldsymbol{m}$*. The output probability distribution is uniform in the associated coset.*

Proof. The set $\{\mathsf{Encrypt}_{\mathsf{AHE}}(\boldsymbol{sk},\boldsymbol{0})\}$ is a subgroup as for any $(\boldsymbol{u},\boldsymbol{s}\cdot\boldsymbol{u}+\boldsymbol{e}),(\boldsymbol{u}',\boldsymbol{s}\cdot\boldsymbol{u}'+\boldsymbol{e}')\in\{\mathsf{Encrypt}_{\mathsf{AHE}}(\boldsymbol{sk},\boldsymbol{0})\}$ we have that $\boldsymbol{u}-\boldsymbol{u}'\in\mathbb{F}_{q^m}^n$ and $\boldsymbol{e}-\boldsymbol{e}'\in F^n$ and thus $(\boldsymbol{u}-\boldsymbol{u}',\boldsymbol{s}\cdot(\boldsymbol{u}-\boldsymbol{u}')+(\boldsymbol{e}-\boldsymbol{e}'))$ is in $\{\mathsf{Encrypt}_{\mathsf{AHE}}(\boldsymbol{sk},\boldsymbol{0})\}$. Moreover the output distribution of $\mathsf{Encrypt}_{\mathsf{AHE}}$ on this set is uniform as for a given $\boldsymbol{s}$ there is a one to one mapping between the pairs $(\boldsymbol{u},\boldsymbol{e})$ and $(\boldsymbol{u},\boldsymbol{s}\cdot\boldsymbol{u}+\boldsymbol{e})$ and the pairs $(\boldsymbol{u},\boldsymbol{e})$ are chosen uniformly on $\mathbb{F}_{q^m}^n\times F^n$. Proving the rest of the proposition is trivial as the encryption process consists exactly on generating a ciphertext of zero and adding $\boldsymbol{g}^{(1)}\star\boldsymbol{m}$ to it.

□

The security of this encryption scheme is reduced to the IRSD problem in Section 6; the problem of decrypting ℓ independent ciphertexts is equivalent to solving the syndrome decoding in an $(\ell+1)$-ideal code. For small values of ℓ, the IRSD is known to be hard and is at the core of the security of other encryption schemes such as ROLLO or RQC.

3.2 Additively Homomorphic Algorithms

Figure 2 presents the homomorphic algorithms of our additive scheme.

- $\mathsf{Add}_{\mathsf{AHE}}(\boldsymbol{ct}=(\boldsymbol{u},\boldsymbol{v}),\boldsymbol{ct}'=(\boldsymbol{u}',\boldsymbol{v}'))$: returns $(\boldsymbol{u}+\boldsymbol{u}',\boldsymbol{v}+\boldsymbol{v}')\in\mathbb{F}_{q^m}^n\times\mathbb{F}_{q^m}^n$.
- $\mathsf{PtxtMul}_{\mathsf{AHE}}(\boldsymbol{m}'\in\mathbb{F}_q^n,\boldsymbol{ct}=(\boldsymbol{u},\boldsymbol{v}))$: returns $(\boldsymbol{m}'\cdot\boldsymbol{u},\boldsymbol{m}'\cdot\boldsymbol{v})\in\mathbb{F}_{q^m}^n\times\mathbb{F}_{q^m}^n$.

Fig. 2. Additively Homomorphic Algorithms.

In the following proposition we consider $\mathbb{F}_q^n$ as the ring $(\mathbb{F}_q^n,+,\cdot)$, $+$ the natural addition in $\mathbb{F}_q^n$ and $\cdot$ the multiplication in $\mathbb{F}_q[X]/\langle Q\rangle$ (see subsection 2.3).

Proposition 3 (Encryption is an $\mathbb{F}_q^n$-module isomorphism). *For any properly generated key* $\boldsymbol{sk}$*, the function* $f=\mathsf{Encrypt}_{\mathsf{AHE}}(\boldsymbol{sk},\cdot,(\cdot,\cdot))$ *is an* $\mathbb{F}_q^n$*-module isomorphism between* $(Dom(f),+)$ *and* $(Im(f),+)$ *with* $Dom(f)=\mathbb{F}_q^n\times\mathbb{F}_{q^m}^n\times\mathcal{M}_{w,n}(\mathbb{F}_q)$ *and* $Im(f)\subset\mathbb{F}_{q^m}^n\times\mathbb{F}_{q^m}^n$.

Proof. First note that $(\mathrm{Im}(f),+)$ is an $\mathbb{F}_q^n$-module. It is a subgroup of $(\mathbb{F}_{q^m}^n\times\mathbb{F}_{q^m}^n,+)$ as for any two ciphertexts $\boldsymbol{ct}=(\boldsymbol{u},\boldsymbol{s}\cdot\boldsymbol{u}+\boldsymbol{e}+\boldsymbol{g}^{(1)}\star\boldsymbol{m})$, $\boldsymbol{ct}'=(\boldsymbol{u}',\boldsymbol{s}\cdot\boldsymbol{u}'+\boldsymbol{e}'+\boldsymbol{g}^{(1)}\star\boldsymbol{m}')$, we have $\boldsymbol{ct}-\boldsymbol{ct}'=(\boldsymbol{u}-\boldsymbol{u}',\boldsymbol{s}\cdot(\boldsymbol{u}-\boldsymbol{u}')+(\boldsymbol{e}-\boldsymbol{e}')+\boldsymbol{g}^{(1)}\star(\boldsymbol{m}-\boldsymbol{m}'))\in\{\mathsf{Encrypt}_{\mathsf{AHE}}(\boldsymbol{sk},\boldsymbol{m}-\boldsymbol{m}')\}\subset\mathrm{Im}(f)$. Moreover for any $\boldsymbol{m}'\in\mathbb{F}_q^n$ we have $\boldsymbol{m}'\cdot\boldsymbol{ct}=(\boldsymbol{m}'\cdot\boldsymbol{u},\boldsymbol{s}\cdot(\boldsymbol{m}'\cdot\boldsymbol{u})+\boldsymbol{m}'\cdot\boldsymbol{e}+\boldsymbol{g}^{(1)}\star(\boldsymbol{m}'\cdot\boldsymbol{m}))\in\{\mathsf{Encrypt}(\boldsymbol{sk},\boldsymbol{m}'\cdot\boldsymbol{m})\}\subset\mathrm{Im}(f)$ as $\boldsymbol{m}\cdot\boldsymbol{u}\in\mathbb{F}_{q^m}^n$ and $\boldsymbol{m}'\cdot\boldsymbol{e}\in F^n$ (F being an $\mathbb{F}_q$-linear span).

To prove that $(\mathrm{Dom}(f),+)$ with $\mathrm{Dom}(f)=\mathbb{F}_q^n\times\mathbb{F}_{q^m}^n\times\mathcal{M}_{w,n}(\mathbb{F}_q)$ is an $\mathbb{F}_q^n$-module we must define the external multiplication with an element in $\mathbb{F}_q^n$. We define it by multiplying coordinatewise. The first two coordinates correspond to multiplications in $\mathbb{F}_q[X]/\langle Q\rangle$ and $\mathbb{F}_{q^m}[X]/\langle Q\rangle$. For the last one we consider that the external multiplication is done with each of the w rows of the matrix over $\mathbb{F}_q[X]/\langle Q\rangle$. With this operation it is trivial to verify that we obtain an $\mathbb{F}_q^n$-module.

The identity element of $(\mathbb{F}_q^n \times \mathbb{F}_{q^m}^n \times \mathcal{M}_{w,n}(\mathbb{F}_q), +)$ is $(\mathbf{0}, \mathbf{0}, \mathbf{0})$ and $f(\mathbf{0}, \mathbf{0}, \mathbf{0}) = \mathsf{Encrypt}_{\mathsf{AHE}}(\mathbf{0}, (\mathbf{0}, \mathbf{0})) = (\mathbf{0}, \mathbf{0})$ the identity element of $\mathrm{Im}(f)$.

For $\boldsymbol{m}' \in \mathbb{F}_q^n$ and $(\boldsymbol{m}, \boldsymbol{r}_1, \boldsymbol{R}_2) \in \mathrm{Dom}(f)$, we have $f(\boldsymbol{m}' \cdot \boldsymbol{m}, \boldsymbol{m}' \cdot \boldsymbol{r}_1, \boldsymbol{m}' \cdot \boldsymbol{R}_2) = (\boldsymbol{m}' \cdot \boldsymbol{r}_1, \boldsymbol{s} \cdot (\boldsymbol{m}' \cdot \boldsymbol{r}_1) + \boldsymbol{f} \cdot \boldsymbol{m}' \cdot \boldsymbol{R}_2 + \boldsymbol{g}^{(1)} \star \boldsymbol{m}' \cdot \boldsymbol{m}) = \boldsymbol{m}' \cdot f(\boldsymbol{m}, \boldsymbol{r}_1, \boldsymbol{R}_2)$ using the commutative and associative properties of the involved polynomial operations, which concludes the proof.

□

Corollary 1 (Homomorphic Addition Distribution). *For any $\boldsymbol{m}, \boldsymbol{m}' \in \mathbb{F}_q^n$, properly generated key $\boldsymbol{sk}$, and $\boldsymbol{ct} \in \{\mathsf{Encrypt}_{\mathsf{AHE}}(\boldsymbol{sk}, \boldsymbol{m})\}$, let $\boldsymbol{ct}''$ be obtained by*

$$\boldsymbol{ct}' \overset{\$}{\leftarrow} \mathsf{Encrypt}_{\mathsf{AHE}}(\boldsymbol{sk}, \boldsymbol{m}')$$
$$\boldsymbol{ct}'' = \mathsf{Add}_{\mathsf{AHE}}(\boldsymbol{ct}, \boldsymbol{ct}')$$

and let $\boldsymbol{ct}'''$ be obtained by

$$\boldsymbol{ct}''' \overset{\$}{\leftarrow} \mathsf{Encrypt}_{\mathsf{AHE}}(\boldsymbol{sk}, \boldsymbol{m} + \boldsymbol{m}').$$

Then the distributions of $\boldsymbol{ct}''$ and $\boldsymbol{ct}'''$ are identical.

Proof. The proof is immediately derived from Proposition 3 as it proves that $\boldsymbol{ct}'' = \mathsf{Encrypt}_{\mathsf{AHE}}(\boldsymbol{sk}, \boldsymbol{m} + \boldsymbol{m}', \boldsymbol{r} + \boldsymbol{r}')$ with $\boldsymbol{r}'$ uniformly sampled and $\boldsymbol{r}$ independent from $\boldsymbol{r}'$, and thus $\boldsymbol{r} + \boldsymbol{r}'$ is uniform in $\mathbb{F}_{q^m}^n \times \mathcal{M}_{w,n}(\mathbb{F}_q)$.

Note that obtaining the same distribution as a fresh ciphertext of the sum is a much stronger property than decrypting to the sum. In practice it implies (among other things) that no information about a computation, besides the result, can leak from the output ciphertext if one of the input ciphertexts was generated with $\mathsf{Encrypt}_{\mathsf{AHE}}$ and unknown to the decrypter (which is definitely not naturally true with lattice-based schemes). It also implies that there is no bound on the amount of ciphertexts that can be added (which again is not naturally true for lattice-based schemes), but we delay the formalization of these properties into an associated corollary to make it more general so that it takes into account arbitrary linear combinations of ciphertexts. We prove thus first that multiplications of ciphertexts by plaintexts also lead to the same distribution as fresh ciphertexts.

Corollary 2 (Homorphic Plaintext Multiplication Distribution). *For any $\boldsymbol{m} \in \mathbb{F}_q^n$, $\boldsymbol{m}' \in \mathbb{F}_q^n\{\mathbf{0}\}$ and properly generated key $\boldsymbol{sk}$, let $\boldsymbol{ct}'$ be obtained by $\boldsymbol{ct} \overset{\$}{\leftarrow} \mathsf{Encrypt}_{\mathsf{AHE}}(\boldsymbol{sk}, \boldsymbol{m})$ and $\boldsymbol{ct}' = \boldsymbol{m}' \cdot \boldsymbol{ct}$. Let $\boldsymbol{ct}''$ be obtained by $\boldsymbol{ct}'' \overset{\$}{\leftarrow} \mathsf{Encrypt}_{\mathsf{AHE}}(\boldsymbol{sk}, \boldsymbol{m} \cdot \boldsymbol{m}')$. The distributions of $\boldsymbol{ct}'$ and $\boldsymbol{ct}''$ are identical.*

Proof. Again, the proof is immediately derived from 3 as it proves that $\boldsymbol{ct}' = \mathsf{Encrypt}_{\mathsf{AHE}}(\boldsymbol{sk}, \boldsymbol{m}' \cdot \boldsymbol{m}, \boldsymbol{m}' \cdot \boldsymbol{r})$ with $\boldsymbol{r}$ uniformly sampled and $\boldsymbol{m}'$ independent from $\boldsymbol{r}$, and thus $\boldsymbol{m}' \cdot \boldsymbol{r}$ is uniform in $\mathbb{F}_{q^m}^n \times \mathcal{M}_{w,n}(\mathbb{F}_q)$ as $\boldsymbol{m}'$ is invertible $(\mathbb{F}_q[X]/\langle Q \rangle$ being a field) and it therefore does not alter the uniform distribution.

□

We are now ready to prove the corollary summarizing the results of this section.

Corollary 3. *Any non-null linear combination, with coefficients in $\mathbb{F}_q^n$, of independent ciphertexts follows the same distribution as a fresh encryption of the same linear combination over the associated plaintexts. The resulting ciphertext decrypts correctly.*

Proof. The first result is obtained by removing first the null coefficients (not all are null as it is a non-null linear combination). Then we apply corollary 2 to each plaintext multiplication and corollary 1 iteratively. The second result is obtained using the first result and proposition 1.

□

The case in which some ciphertexts are inter-dependent, even maliciously, is more complex and beyond the scope of this paper. However it is important to note that with the properties described in this section it is quite manageable, unlike for lattice-based homomorphic encryption schemes for which this issue is quickly very complex.

Remark 3. Functions $\mathsf{Add}_{\mathsf{AHE}}$ and $\mathsf{PtxtMul}_{\mathsf{AHE}}$ corresponding to natural operations, from now on we will in general not call these functions explicitly replacing directly $\mathsf{Add}_{\mathsf{AHE}}(\boldsymbol{ct}, \boldsymbol{ct}')$ with operation $\boldsymbol{ct} + \boldsymbol{ct}'$ and $\mathsf{PtxtMul}_{\mathsf{AHE}}(\boldsymbol{m}, \boldsymbol{ct})$ with $\boldsymbol{m} \cdot \boldsymbol{ct}$ (where $\boldsymbol{m}$ multiplies each of the two coordinates of the vector $\boldsymbol{ct}$).

4 Somewhat Homomorphic Scheme

In this section we extend our additive scheme to a somewhat homomorphic scheme that can perform unlimited additions and one multiplication. The homomorphic multiplication operation transforms a two-component ciphertext into a three-component ciphertext, which can be decrypted with an alternative decryption algorithm.

4.1 Fundamental and Additively Homomorphic Algorithms

The fundamental algorithms constituting our scheme SHE, and the additively homomorphic algorithms directly inherited from AHE are depicted in Fig. 3. The scheme is parametrized by the same variables than the previous scheme, with a different condition on the weight w:

- q, the base field cardinality;
- m, the dimension of the field extension;
- n, the length of the vectors;
- w, the rank weight of the error; it must be that $\frac{w(w+3)}{2} + 1 < m$.

As for AHE we define a randomized algorithm $\mathsf{Encrypt}_{\mathsf{SHE}}(\boldsymbol{sk}, \boldsymbol{m})$ that samples $\boldsymbol{r} \overset{\$}{\leftarrow} \mathbb{F}_{q^m}^n \times \mathcal{M}_{w,n}(\mathbb{F}_q)$ and returns $\mathsf{Encrypt}_{\mathsf{SHE}}(\boldsymbol{sk}, \boldsymbol{m}, \boldsymbol{r})$. The SHE scheme does not change the encryption, decryption, addition and plaintext multiplication algorithms, it only gives a stronger constraint on w when defining the parameters and has a more complex key generation algorithm to ensure that noise can be separated from the message space even after a homomorphic multiplication.

- $\mathsf{KeyGen}_{\mathsf{SHE}}()$:
  - samples $g_1 \xleftarrow{\$} \mathbb{F}_{q^m}$, $\boldsymbol{f} = (f_1, \ldots, f_w) \xleftarrow{\$} \mathcal{S}_w^w(\mathbb{F}_{q^m})$
  - defines $F = \mathsf{supp}(f_1, \ldots, f_w)$ and $\tilde{F} = \mathsf{supp}(\boldsymbol{f}, g_1 \star \boldsymbol{f}, (f_i f_j)_{1 \leq i,j \leq w})$
  - computes $\tilde{\boldsymbol{f}} = (f_1, \ldots, f_{d_{\tilde{F}}}) \in \mathcal{S}_{d_{\tilde{F}}}^{d_{\tilde{F}}}(\mathbb{F}_{q^m})$ a basis of $\tilde{F}$ with $d_{\tilde{F}} = \dim(\tilde{F})$
  - defines $g_2 = g_1 g_1$
  - checks that $\mathsf{rw}(f_1, \ldots, f_{d_{\tilde{F}}}, g_1, g_2) = d_{\tilde{F}} + 2$ (if not it restarts)
  - extends this vector into a basis $\boldsymbol{b} = (f_1, \ldots, f_{d_{\tilde{F}}}, g_1, \ldots, g_{m-d_{\tilde{F}}}) \in \mathcal{S}_m^m(\mathbb{F}_{q^m})$
  - defines $\boldsymbol{g} = (g_1, \ldots, g_{m-d_{\tilde{F}}})$
  - computes the matrix $\boldsymbol{B} = \mathbf{Mat}(\boldsymbol{b})$
  - defines $\boldsymbol{D}$ as the last $m - d_{\tilde{F}}$ columns of $(\boldsymbol{B}^{-1})^T$
  - samples $\boldsymbol{s} \xleftarrow{\$} F^n$ with $F = \mathsf{supp}(\boldsymbol{f})$
  - returns $\boldsymbol{sk} = (\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{D}, \boldsymbol{s})$.

- $\mathsf{Encrypt}_{\mathsf{SHE}} = \mathsf{Encrypt}_{\mathsf{AHE}}$.
- $\mathsf{Decrypt}_{\mathsf{SHE}} = \mathsf{Decrypt}_{\mathsf{AHE}}$.
- $\mathsf{Add}_{\mathsf{SHE}} = \mathsf{Add}_{\mathsf{AHE}}$.
- $\mathsf{PtxtMul}_{\mathsf{SHE}} = \mathsf{PtxtMul}_{\mathsf{AHE}}$.

Fig. 3. Description of the fully homomorphic scheme.

Proposition 4 (Extension of AHE properties to SHE). *Propositions 2, 1, and 3 and Corollaries 1, 2 and 3 remain true when replacing* AHE *with* SHE.

Proof. The associated proofs only use properties of $\boldsymbol{D}$ and the definitions of $\mathsf{Encrypt}_{\mathsf{AHE}}$, $\mathsf{Decrypt}_{\mathsf{AHE}}$, $\mathsf{Add}_{\mathsf{AHE}}$ and $\mathsf{PtxtMul}_{\mathsf{AHE}}$. It can be easily checked that the used properties of $\boldsymbol{D}$ are maintained and that the definitions of $\mathsf{Encrypt}_{\mathsf{AHE}}$, $\mathsf{Decrypt}_{\mathsf{AHE}}$, $\mathsf{Add}_{\mathsf{AHE}}$ and $\mathsf{PtxtMul}_{\mathsf{AHE}}$ are unchanged.

□

4.2 Multiplicative Homomorphic Algorithms

- $\mathsf{Mul}(\boldsymbol{ct} = (\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{ct}' = (\boldsymbol{u}', \boldsymbol{v}'))$: returns $(\boldsymbol{v} \cdot \boldsymbol{v}', -(\boldsymbol{u} \cdot \boldsymbol{v}' + \boldsymbol{u}' \cdot \boldsymbol{v}), \boldsymbol{u} \cdot \boldsymbol{u}') \in \mathbb{F}_{q^m}^n \times \mathbb{F}_{q^m}^n \times \mathbb{F}_{q^m}^n$

- $\mathsf{DecryptMul}(\boldsymbol{sk} = (\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{D}, \boldsymbol{s}), (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}))$: computes $\boldsymbol{tmp} = \boldsymbol{a} + \boldsymbol{s} \cdot \boldsymbol{b} + \boldsymbol{s} \cdot \boldsymbol{s} \cdot \boldsymbol{c}$, and returns $\boldsymbol{d}^T \mathbf{Mat}(\boldsymbol{tmp})$ with $\boldsymbol{d} = \boldsymbol{D}^{(2)}$.

Fig. 4. Multiplicative Homomorphic Algorithms.

Proposition 5 (Homomorphic Multiplication Decryption Correctness). *For any* $\boldsymbol{m}, \boldsymbol{m}' \in \mathbb{F}_q^n$*, and properly generated key* $\boldsymbol{sk}$*, let* $\boldsymbol{ct} \in \{\mathsf{Encrypt}(\boldsymbol{sk}, \boldsymbol{m})\}$ *and* $\boldsymbol{ct}' \in \{\mathsf{Encrypt}(\boldsymbol{sk}, \boldsymbol{m}')\}$*. We have* $\mathsf{DecryptMul}(\boldsymbol{sk}, \mathsf{Mul}(\boldsymbol{ct}, \boldsymbol{ct}')) = \boldsymbol{m} \cdot \boldsymbol{m}'$.

Proof. Let's note $\boldsymbol{ct} = (\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{ct'} = (\boldsymbol{u'}, \boldsymbol{v'})$ and $\mathsf{Mul}(\boldsymbol{ct}, \boldsymbol{ct'}) = (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$. We then have

$$\begin{aligned}\boldsymbol{tmp} &= \boldsymbol{a} + \boldsymbol{s} \cdot \boldsymbol{b} + \boldsymbol{s} \cdot \boldsymbol{s} \cdot \boldsymbol{c} \\ &= \boldsymbol{v} \cdot \boldsymbol{v'} - \boldsymbol{s} \cdot (\boldsymbol{u} \cdot \boldsymbol{v'} + \boldsymbol{u'} \cdot \boldsymbol{v}) + \boldsymbol{s} \cdot \boldsymbol{s} \cdot \boldsymbol{u} \cdot \boldsymbol{u'} \\ &= (\boldsymbol{v} - \boldsymbol{s} \cdot \boldsymbol{u}) \cdot (\boldsymbol{v'} - \boldsymbol{s} \cdot \boldsymbol{u'}) \\ &= (\hat{\boldsymbol{m}} + \boldsymbol{e}) \cdot (\hat{\boldsymbol{m}}' + \boldsymbol{e'}) \\ &= \hat{\boldsymbol{m}} \cdot \hat{\boldsymbol{m}}' + \hat{\boldsymbol{m}}' \cdot \boldsymbol{e} + \hat{\boldsymbol{m}} \cdot \boldsymbol{e'} + \boldsymbol{e} \cdot \boldsymbol{e'}\end{aligned}$$

with $\hat{\boldsymbol{m}} \cdot \hat{\boldsymbol{m}}' = \boldsymbol{g}^{(1)}\boldsymbol{g}^{(1)} \star \boldsymbol{m} \cdot \boldsymbol{m'}$ and $\hat{\boldsymbol{m}}' \cdot \boldsymbol{e} + \hat{\boldsymbol{m}} \cdot \boldsymbol{e'} + \boldsymbol{e} \cdot \boldsymbol{e'} \in \tilde{F}^n$. We thus can write

$$\boldsymbol{tmp} = \boldsymbol{g}^{(1)}\boldsymbol{g}^{(1)} \star \boldsymbol{m} \cdot \boldsymbol{m'} + \sum_{1 \le i \le d_{\tilde{F}}} f_i \star \boldsymbol{e}_i \qquad \text{with } \boldsymbol{e}_i \in \mathbb{F}_q^n.$$

As $\boldsymbol{g}^{(1)}\boldsymbol{g}^{(1)} = \boldsymbol{g}^{(2)}$ and $\boldsymbol{d} = \boldsymbol{D}^{(2)}$, $\boldsymbol{d}^T\mathbf{vec}(\boldsymbol{g}^{(2)}) = 1$ and $\boldsymbol{d}^T\mathbf{vec}(f_i) = 0$ for $1 \le i \le d_{\tilde{F}}$, we thus have $\boldsymbol{d}^T\mathbf{Mat}(\boldsymbol{tmp}) = \boldsymbol{m} \cdot \boldsymbol{m'}$.

□

It is possible to define an encryption function that directly creates ciphertexts of the form $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ (with noise drawn from $\tilde{F}$) and show that the result of the multiplication of two fresh ciphertexts of the form $(\boldsymbol{u}, \boldsymbol{v})$ follows the same distribution as a fresh three-coordinate ciphertext associated with the product of plaintexts. It is also possible to show that non-null linear combinations of these three-coordinate ciphertexts have the same distributions as fresh three-coordinate ciphertexts. We do not delve into these proofs as in practice three-coordinate ciphertexts will be transformed back to two coordinate ciphertexts. However if for some reason (*e.g.* computing a degree two polynomial with a simple scheme) one wants to handle three-coordinate ciphertexts it is important to understand that the nice distributional properties of two-coordinate ciphertexts are maintained.

The Somewhat Homomorphic Encryption scheme described above can be adapted so as to perform the evaluation of a arbitrary polynomial of degree d. However this has two implications that we state informally. First, the ciphertext is expanded to $d + 1$ coordinates. Second, multiplications are expanding the noise space, meaning that the parameters must satisfy $w^d = \mathcal{O}(m)$. These two conditions require the choice of large and impractical parameters for high values of d.

5 (Insecure) Bootstrapping and Fully Homomorphic Encryption

In this section we present bootstrapping algorithms that homomorphically applies either $\mathsf{Decrypt}_{\mathsf{SHE}}$ or $\mathsf{DecryptMul}_{\mathsf{SHE}}$ on a ciphertext. Our bootstrapping algorithm has no multiplicative depth so it produces a two-component $(\boldsymbol{u}, \boldsymbol{v})$ fresh ciphertext with a new key. The simplicity of our construction gives a glimpse of a practical Fully Homomorphic Encryption scheme that would allow to compute arbitrary circuits.

However, this first bootstrapping construction is unsecure as the number of bootstrapping keys ($2m$ for the case of a homomorphic evaluation of $\mathsf{Decrypt}_{\mathsf{SHE}}$)

is higher than the upper bound on the number of independent ciphertexts allowed ($2w$, cf. Section 6). Therefore, at the moment, it cannot be used for a secure FHE.

We first present in Figure 5 a homomorphic decryption algorithm that works on two-component $(\boldsymbol{u}, \boldsymbol{v})$ ciphertexts, then present in Figure 6 a bootstrapping relinearization algorithm that is a homomorphic decryption on three-component $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ ciphertexts.

5.1 Homomorphic Decryption Algorithms

In this section we explicitly note $(\gamma_1, \ldots, \gamma_m)$ the public basis in which an element of $\mathbb{F}_q^m$ represents an element of $\mathbb{F}_{q^m}$.

```
– $\mathsf{GenKeySwitch}_{\mathsf{SHE}}(\boldsymbol{sk}_1 = (\boldsymbol{f}_1, \boldsymbol{g}_1, \boldsymbol{D}_1, \boldsymbol{s}_1))$:
    • generates a new basis $\boldsymbol{b}_2$ of $\mathbb{F}_{q^m}$, and $\boldsymbol{sk}_2 = (\boldsymbol{f}_2, \boldsymbol{g}_2, \boldsymbol{D}_2, \boldsymbol{s}_2)$ as in $\mathsf{KeyGen}_{\mathsf{SHE}}$
    • for $1 \leq i \leq m$ defines $\boldsymbol{s}_{1,i} = \boldsymbol{d}^T \mathbf{Mat}(\gamma_i \star \boldsymbol{s}_1)$ with $\boldsymbol{d} = \boldsymbol{D}_1^{(1)}$
    • defines $\mathsf{ksk} = (\mathsf{ksk}_1, \ldots, \mathsf{ksk}_m)$, with $\mathsf{ksk_i} \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{SHE}}(\boldsymbol{sk}_2, \boldsymbol{s}_{1,i})$
    • for $1 \leq i \leq m$ defines $\boldsymbol{p}_i = \boldsymbol{d}^{(i)} \star (1, 0, \ldots, 0) \in \mathbb{F}_{q^m}^n$
    • defines $\mathsf{projk} = (\mathsf{projk}_1, \ldots, \mathsf{projk}_m)$ with $\mathsf{projk}_i \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{SHE}}(\boldsymbol{sk}_2, \boldsymbol{p}_i)$
    • returns $(\boldsymbol{sk}_2, \mathsf{ksk}, \mathsf{projk})$
– $\mathsf{HomDecrypt}_{\mathsf{SHE}}(\mathsf{ksk}, \mathsf{projk}, \boldsymbol{ct} = (\boldsymbol{u}, \boldsymbol{v}))$:
    • defines $\boldsymbol{u}_i \in \mathbb{F}_q^n$ such that $\boldsymbol{u} = \sum_{1 \leq i \leq m} \gamma_i \star \boldsymbol{u}_i$
    • defines $\boldsymbol{v}_i \in \mathbb{F}_q^n$ such that $\boldsymbol{v} = \sum_{1 \leq i \leq m} \gamma_i \star \boldsymbol{v}_i$
    • computes $\boldsymbol{ct}_1 = \sum_{1 \leq i \leq m} \boldsymbol{v}_i \cdot \mathsf{projk}_i$
    • computes $\boldsymbol{ct}_2 = \sum_{1 \leq i \leq m} \boldsymbol{u}_i \cdot \mathsf{ksk}_i$
    • returns $\boldsymbol{ct}_1 - \boldsymbol{ct}_2$
```

Fig. 5. Homomorphic Decryption Algorithms.

Proposition 6 (Homomorphic Decryption Distribution). *For any properly generated key $\boldsymbol{sk}_1$, any $\boldsymbol{ct} \in (\mathbb{F}_{q^m}^n)^* \times (\mathbb{F}_{q^m}^n)^*$ such that* $\mathsf{Decrypt}_{\mathsf{SHE}}(\boldsymbol{sk}_1, \boldsymbol{ct}) = \boldsymbol{m} \in \mathbb{F}_q^n$*, and* $(\boldsymbol{sk}_2, \mathsf{ksk}, \mathsf{projk}) \xleftarrow{\$} \mathsf{GenKeySwitch}_{\mathsf{SHE}}(\boldsymbol{sk}_1)$*, let $\boldsymbol{ct}'$ be obtained by $\boldsymbol{ct}' =$* $\mathsf{HomDecrypt}_{\mathsf{SHE}}(\mathsf{ksk}, \mathsf{projk}, \boldsymbol{ct})$ *and let $\boldsymbol{ct}''$ be obtained by $\boldsymbol{ct}'' \xleftarrow{\$}$* $\mathsf{Encrypt}_{\mathsf{SHE}}(\boldsymbol{sk}_2, \boldsymbol{m})$*. The distributions of $\boldsymbol{ct}'$ and $\boldsymbol{ct}''$ are identical.*

Proof. As $\boldsymbol{ct}$ is non-null, $\boldsymbol{ct}_1 - \boldsymbol{ct}_2$ is a non-null linear combination of the ciphertexts projk_i and ksk_i, that have been generated independently. Thus, using Corollary 3 and Proposition 4, $\boldsymbol{ct}_1 - \boldsymbol{ct}_2$ follows the same distribution as $\boldsymbol{ct}''' \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{SHE}}(\boldsymbol{sk}_2, \sum_{1 \leq i \leq m} \boldsymbol{v}_i \cdot \boldsymbol{p}_i - \sum_{1 \leq i \leq m} \boldsymbol{u}_i \cdot \boldsymbol{s}_{1,i})$.

We thus only need to prove that $\sum_{1 \leq i \leq m} \boldsymbol{v}_i \cdot \boldsymbol{p}_i - \sum_{1 \leq i \leq m} \boldsymbol{u}_i \cdot \boldsymbol{s}_{1,i} = \boldsymbol{m}$. We have

$$\begin{aligned}\sum_{1\le i\le m} \boldsymbol{v}_i \cdot \boldsymbol{p}_i - \sum_{1\le i\le m} \boldsymbol{u}_i \cdot \boldsymbol{s}_{1,i} &= \sum_{1\le i\le m} \boldsymbol{v}_i \cdot \boldsymbol{d}^{(i)} \star (1,0,\ldots,0) - \sum_{1\le i\le m} \boldsymbol{u}_i \cdot \boldsymbol{s}_{1,i} \\ &= \sum_{1\le i\le m} \boldsymbol{d}^{(i)} \star \boldsymbol{v}_i - \sum_{1\le i\le m} \boldsymbol{u}_i \cdot \boldsymbol{s}_{1,i} \\ &= \boldsymbol{d}^T \mathbf{Mat}(\boldsymbol{v}) - \sum_{1\le i\le m} \boldsymbol{u}_i \cdot \boldsymbol{s}_{1,i} \\ &= \boldsymbol{d}^T \mathbf{Mat}(\boldsymbol{v}) - \sum_{1\le i\le m} \boldsymbol{u}_i \cdot \boldsymbol{d}^T \mathbf{Mat}(\gamma_i \star \boldsymbol{s}_1)\end{aligned}$$

noting that $\boldsymbol{d}^T \mathbf{Mat}(\gamma_i \star \boldsymbol{s}_1) = \sum_j \boldsymbol{d}^{(j)} \boldsymbol{\ell}_j$ with $\boldsymbol{\ell}_j$ the lines of $\mathbf{Mat}(\gamma_i \star \boldsymbol{s}_1)$ we can use the distributivity of the polynomial multiplication over the addition to get $\boldsymbol{u}_i \cdot \boldsymbol{d}^T \mathbf{Mat}(\gamma_i \star \boldsymbol{s}_1) = \boldsymbol{d}^T \mathbf{Mat}(\gamma_i \star \boldsymbol{u}_i \cdot \boldsymbol{s}_1)$. We thus obtain

$$\begin{aligned}\sum_{1\le i\le m} \boldsymbol{v}_i \cdot \boldsymbol{p}_i - \sum_{1\le i\le m} \boldsymbol{u}_i \cdot \boldsymbol{s}_{1,i} &= \boldsymbol{d}^T \mathbf{Mat}(\boldsymbol{v}) - \boldsymbol{d}^T \mathbf{Mat}(\sum_{1\le i\le m} \gamma_i \star \boldsymbol{u}_i \cdot \boldsymbol{s}_1) \\ &= \boldsymbol{d}^T \mathbf{Mat}(\boldsymbol{v} - \boldsymbol{u} \cdot \boldsymbol{s}_1) \\ &= \boldsymbol{m}\end{aligned}$$

which concludes the proof.

□

It is very important to note that Proposition 6 ensures that the output of $\mathsf{HomDecrypt}_{\mathsf{SHE}}$ is a well formed and well distributed ciphertext even if $\boldsymbol{ct}$ is not, as long as it decrypts to $\boldsymbol{m}$. There are many implications to Proposition 6. Among them we can highlight that it opens the path to fast and simple algorithms for: bootstrapping relinearization, and trans-encryption. It also allows to rerandomize a ciphertext (assuming that the key switching key is well-formed).

5.2 Bootstrapping relinearization

Proposition 7 (Relinearization Distribution). *For any properly generated key* $\boldsymbol{sk}_1$*, non-null* $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \in \mathbb{F}_{q^m}^n \times \mathbb{F}_{q^m}^n \times \mathbb{F}_{q^m}^n$ *such that* $\mathsf{DecryptMul}_{\mathsf{SHE}}(\boldsymbol{sk}_1, (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})) = \boldsymbol{m} \in \mathbb{F}_q^n$*, and* $(\boldsymbol{sk}_2, \mathsf{ksk}, \mathsf{ksksq}, \mathsf{projk}) \overset{\$}{\leftarrow} \mathsf{GenRelinKey}_{\mathsf{SHE}}(\boldsymbol{sk}_1)$*, let* $\boldsymbol{ct}'$ *be obtained by* $\boldsymbol{ct}' = \mathsf{Relinearize}_{\mathsf{SHE}}(\mathsf{ksk}, \mathsf{ksksq}, \mathsf{projk}, (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}))$ *and let* $\boldsymbol{ct}''$ *be obtained by* $\boldsymbol{ct}'' \overset{\$}{\leftarrow} \mathsf{Encrypt}_{\mathsf{SHE}}(\boldsymbol{sk}_2, \boldsymbol{m})$*. The distributions of* $\boldsymbol{ct}'$ *and* $\boldsymbol{ct}'''$ *are identical.*

Proof. The proof is very similar to the one of $\mathsf{HomDecrypt}_{\mathsf{SHE}}$. We use the fact that $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ is not null and the ciphertexts generated in $\mathsf{GenRelinKey}_{\mathsf{SHE}}$ are independently generated to show that the result is uniformly distributed among the encryptions of a given plaintext. We then show that this plaintext is $\boldsymbol{m}$ by evaluating the correctness.

For the correctness we show that $\boldsymbol{ct}_a$ is in $\mathsf{Encrypt}_{\mathsf{SHE}}(\boldsymbol{sk}_2, \boldsymbol{d}^T \mathbf{Mat}(\boldsymbol{a}))$, then that $\boldsymbol{ct}_b$ is in $\mathsf{Encrypt}_{\mathsf{SHE}}(\boldsymbol{sk}_2, \boldsymbol{d}^T \mathbf{Mat}(\boldsymbol{s}_1 \cdot \boldsymbol{b}))$, and finally that $\boldsymbol{ct}_c$ is in $\mathsf{Encrypt}_{\mathsf{SHE}}(\boldsymbol{sk}_2, \boldsymbol{d}^T \mathbf{Mat}(\boldsymbol{s}_1 \cdot \boldsymbol{s}_1 \cdot \boldsymbol{c}))$. As a consequence we obtain that $\boldsymbol{ct}$ is in $\mathsf{Encrypt}_{\mathsf{SHE}}(\boldsymbol{sk}_2, \mathsf{DecryptMul}_{\mathsf{SHE}}(\boldsymbol{sk}_1, (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}))$ and thus in $\mathsf{Encrypt}_{\mathsf{SHE}}(\boldsymbol{sk}_2, \boldsymbol{m})$.

□

```
– $\mathsf{GenRelinKey}_{\mathsf{SHE}}(\boldsymbol{sk}_1 = (\boldsymbol{f}_1, \boldsymbol{g}_1, \boldsymbol{D}_1, \boldsymbol{s}_1, P))$:
    • generates a new basis $\boldsymbol{b}_2$ of $\mathbb{F}_{q^m}$, and $\boldsymbol{sk}_2 = (\boldsymbol{f}_2, \boldsymbol{g}_2, \boldsymbol{D}_2, \boldsymbol{s}_2, P)$ as in $\mathsf{KeyGen}_{\mathsf{SHE}}$
    • for $1 \leq i \leq m$ defines $\boldsymbol{s}_{1,i} = \boldsymbol{d}^T \mathbf{Mat}(\gamma_i \star \boldsymbol{s}_1)$ with $\boldsymbol{d} = \boldsymbol{D}_1^{(2)}$
    • for $1 \leq i \leq m$ defines $\boldsymbol{s}_{1sq,i} = \boldsymbol{d}^T \mathbf{Mat}(\gamma_i \star \boldsymbol{s}_1 \cdot \boldsymbol{s}_1)$
    • defines $\mathsf{ksk} = (\mathsf{ksk}_1, \ldots, \mathsf{ksk}_m)$, with $\mathsf{ksk_i} \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{SHE}}(\boldsymbol{sk}_2, \boldsymbol{s}_{1,i})$
    • defines $\mathsf{ksksq} = (\mathsf{ksksq}_1, \ldots, \mathsf{ksksq}_m)$, with $\mathsf{ksksq_i} \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{SHE}}(\boldsymbol{sk}_2, \boldsymbol{s}_{1sq,i})$
    • for $1 \leq i \leq m$ defines $\boldsymbol{p}_i = \boldsymbol{d}^{(i)} \star (1, 0, \ldots, 0) \in \mathbb{F}_{q^m}^n$
    • defines $\mathsf{projk} = (\mathsf{projk}_1, \ldots, \mathsf{projk}_m)$ with $\mathsf{projk}_i \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{SHE}}(\boldsymbol{sk}_2, \boldsymbol{p}_i)$
    • returns $(\boldsymbol{sk}_2, \mathsf{ksk}, \mathsf{ksksq}, \mathsf{projk})$
– $\mathsf{Relinearize}_{\mathsf{SHE}}(\mathsf{ksk}, \mathsf{ksksq}, \mathsf{projk}, (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \in (\mathbb{F}_{q^m}^n)^* \times (\mathbb{F}_{q^m}^n)^* \times (\mathbb{F}_{q^m}^n)^*))$:
    • defines $\boldsymbol{a}_i \in \mathbb{F}_q^n$ such that $\boldsymbol{a} = \sum_{1 \leq i \leq m} \gamma_i \star \boldsymbol{a}_i$
    • defines $\boldsymbol{b}_i \in \mathbb{F}_q^n$ such that $\boldsymbol{b} = \sum_{1 \leq i \leq m} \gamma_i \star \boldsymbol{b}_i$
    • defines $\boldsymbol{c}_i \in \mathbb{F}_q^n$ such that $\boldsymbol{c} = \sum_{1 \leq i \leq m} \gamma_i \star \boldsymbol{c}_i$
    • computes $\boldsymbol{ct}_a = \sum_{1 \leq i \leq m} \boldsymbol{a}_i \cdot \mathsf{projk}_i$
    • computes $\boldsymbol{ct}_b = \sum_{1 \leq i \leq m} \boldsymbol{b}_i \cdot \mathsf{ksk}_i$
    • computes $\boldsymbol{ct}_c = \sum_{1 \leq i \leq m} \boldsymbol{c}_i \cdot \mathsf{ksksq}_i$
    • returns $\boldsymbol{ct}_a + \boldsymbol{ct}_b + \boldsymbol{ct}_c$
```

Fig. 6. Relinearization Algorithms.

Note that the relinearization bootstraps the ciphertext. Unlike in previous FHE schemes, the resulting ciphertext is fresh. It has, exactly the same amount of noise as the fresh ciphertexts used for the relinearization. Note that these are direct results of Corollary 3 (doing linear combinations of ciphertexts with coefficients in $\mathbb{F}_q$ does not change the distribution). This seems to be a very powerful property.

Another very interesting property that makes these algorithms possible is the structure of the coefficients of the polynomials forming the ciphertexts. As these coefficients have structure it is much easier to project and reconstruct inside a ciphertext than it would be with bits inside an integer (as in lattice cryptosystems). This structure can also lead to many interesting applications such as directly encoding structured plaintexts (*e.g.* AES states if we take $\mathbb{F}_q = \mathbb{F}_{2^{8 \times 4 \times 4}}$).

6 Problem: Limitation on the number of independent ciphertexts

Even though it is beautiful, this FHE is not secure because, as we will demonstrate in this section, the number of ciphertexts that allows an attacker to retrieve the secret key in polynomial time ($2w$) is lower than the number of ciphertexts of the bootstrapping material ($2m$).

Definition 11 (Rank-SHE Ciphertext Learning Problem (RCL)). *Let* $\boldsymbol{sk} = (\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{D}, \boldsymbol{s}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathsf{SHE}}()$*. Let* $\mathcal{O}$ *be an oracle which samples randomly independent encryptions of* $\mathbf{0}$ *under secret key* $\boldsymbol{sk}$*. The problem* $\mathsf{RCL}_{n,\ell,w}$ *is to recover* $F = \mathsf{supp}(\boldsymbol{f})$ *given* ℓ *accesses to the oracle.*

We will prove below the following result which upper bounds to $2w$ the number of independent ciphertexts that can be crafted with the same key. For the sake of the security reduction, we will now assume that w is below the rank Gilbert-Varshamov bound for parameters $(q, \ell n, n, m)$ for every $\ell > 1$, in order to guarantee the unicity of a solution to the considered problems.

Proposition 8. *$\mathsf{RCL}_{n,\ell,w}$ can be solved in polynomial time when $\ell \geq 2w$.*

The proof requires the following lemma that connects RCL and IRSD.

Lemma 1. *$\mathsf{IRSD}_{n,\ell+1,w}$ is polynomially equivalent to $\mathsf{RCL}_{n,\ell,w}$.*

Proof. Suppose we have a solver for $\mathsf{RCL}_{n,\ell,w}$. Let $(\boldsymbol{H}, \boldsymbol{y} = \boldsymbol{H}\boldsymbol{e}^\top) \in \mathbb{F}_{q^m}^{\ell n \times (\ell+1)n} \times \mathbb{F}_{q^m}^{n\ell}$ be an instance of $\mathsf{IRSD}_{n,\ell+1,w}$.

By applying Gaussian elimination on the ideal blocks of $\boldsymbol{H}$, we can reduce $\boldsymbol{H}$ to its systematic form. Namely there exists an invertible matrix $\boldsymbol{M} \in \mathcal{M}_{\ell n}(\mathbb{F}_{q^m})$ such that:

$$\boldsymbol{H} = \boldsymbol{M} \begin{pmatrix} \boldsymbol{I}_n & & & & \mathcal{IM}_Q(\boldsymbol{u}_1) \\ & \boldsymbol{I}_n & & & \mathcal{IM}_Q(\boldsymbol{u}_2) \\ & & \ddots & & \vdots \\ & & & \boldsymbol{I}_n & \mathcal{IM}_Q(\boldsymbol{u}_{\ell-1}) \\ & & & & \boldsymbol{I}_n \ \mathcal{IM}_Q(\boldsymbol{u}_\ell) \end{pmatrix} \tag{1}$$

By rewriting $\boldsymbol{M}^{-1}\boldsymbol{y}$ as $(\boldsymbol{v}_1, \cdots, \boldsymbol{v}_\ell)$ where the $\boldsymbol{v}_i$ are in $\mathbb{F}_{q^m}^n$, and $\boldsymbol{e}$ as $(\boldsymbol{e}_1, \cdots, \boldsymbol{e}_\ell, \boldsymbol{s})$ we get the following equalities :

$$\boldsymbol{v}_1 = \boldsymbol{u}_1 \cdot \boldsymbol{s} + \boldsymbol{e}_1$$
$$\vdots$$
$$\boldsymbol{v}_\ell = \boldsymbol{u}_\ell \cdot \boldsymbol{s} + \boldsymbol{e}_\ell$$

Therefore, as the distributions of $\boldsymbol{e}$ in IRSD on one hand, and $(\boldsymbol{e}_1, \cdots, \boldsymbol{e}_\ell, \boldsymbol{s})$ in RCL on the other hand, are the same, $(\boldsymbol{u}_i, \boldsymbol{v}_i)_{i \leq \ell}$ is a (random) instance of $\mathsf{RCL}_{n,\ell,w}$ (for a secret key $\boldsymbol{sk} = (\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{D}, \boldsymbol{s})$ such that $\mathsf{supp}(\boldsymbol{f}) = \mathsf{supp}(\boldsymbol{e})$) so a solver can recover the support of $\boldsymbol{e}$. It is only a matter of linear algebra to compute the exact coordinates of $\boldsymbol{e}$ and thus to solve the instance of $\mathsf{IRSD}_{n,\ell+1,w}$.

Conversely, suppose we have a solver for $\mathsf{IRSD}_{n,\ell+1,w}$. Let $\boldsymbol{sk} = (\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{D}, \boldsymbol{s})$ and let $(\boldsymbol{u}_i, \boldsymbol{v}_i = \boldsymbol{u}_i \cdot \boldsymbol{s} + \boldsymbol{e}_i)_{i \leq \ell}$ be an instance of $\mathsf{RCL}_{n,\ell,w}$. We then draw a random invertible matrix $\boldsymbol{M}$ and apply the same transformation as before: define a matrix $\boldsymbol{H}$ as in Equation 1 and $\boldsymbol{y} = (\boldsymbol{M}\boldsymbol{v}_1, \cdots, \boldsymbol{M}\boldsymbol{v}_\ell)$ .We obtain a random instance $(\boldsymbol{H}, \boldsymbol{y})$ of $\mathsf{IRSD}_{n,\ell+1,w}$,we can use our solver, and thanks to the unicity of a solution, the support of the recovered error will precisely be the secret space F.

All transformations in this proof are obviously polynomial so we get that $\mathsf{IRSD}_{n,\ell+1,w}$ and $\mathsf{RCL}_{n,\ell,w}$ are polynomially equivalent. □

Proof (of Proposition 8). We use the linearization attack against RSD presented in [21, Proposition 2] that is effective against small rate codes. This attacks solves the decoding problem with weight w in an $[n, k]_{q^m}$ code under the following condition:

$$n \geq (k + 1)(w + 1) - 1.$$

As seen in the above lemma, the ideal code constructed from the $\mathsf{RCL}_{n,\ell,w}$ instance is an $[n(\ell+1), n]_{q^m}$ code. In that setting, the linearization attack works when

$$n(\ell+1) \geq (n+1)(w+1) - 1,$$

i.e.

$$\ell + 1 \geq w + \frac{w}{n}.$$

Because $n \geq 1$, this clearly shows that $\ell \geq 2w$ is a sufficient condition to break RCL in polynomial time. □

Remark 4. The RCL problem is defined for encryptions of $\mathbf{0}$, so it is not obvious whether the polynomial attack would work for encryptions of random messages $\boldsymbol{m}$. However, we find this attack sufficiently dangerous to claim that $2w$ is the maximal number of ciphertexts that can be safely published, even for non-zero messages.

7 Reducing the number of bootstrapping ciphertexts

In order to reduce the number of ciphertexts, we pack the plaintext into several components which are linked publicly so that the server can select which component of the packing they want. It allows to reduce the number of bootstrapping ciphertexts from $2m$ to $2(w+1)$, which is a major improvement but is unfortunately still unsecure.

7.1 Packing plaintexts

In order to simplify and since our bootstrapping procedure does not need any multiplication, we only present packing for the additive homomorphic scheme. It could be easily extended to the somewhat homomorphic scheme.

In this section we present a variant of AHE which packs as many plaintexts in $\mathbb{F}_q^n$ as possible in a single ciphertext. We call this variant PAHE (for Packed Additively Homomorphic Encryption).

The fundamental algorithms constituting our scheme PAHE, and the additively homomorphic algorithms directly inherited from AHE are depicted in Fig. 7. The scheme has now a public key $\boldsymbol{pk}$ that consists of a field element $\rho \in \mathbb{F}_{q^m}$ that will be used for manipulating packed ciphertexts.

The scheme is parametrized by an additional parameter t that accounts for the size of the packing:

- q, the base field cardinality;
- m, the dimension of the field extension;
- n, the length of the vectors;
- w, the rank weight of the error;
- t, the maximal number of plaintexts that can be packed in a single ciphertext; it must be that $t(w+1) \leq m$ and that m is divisible by t.

- $\mathsf{KeyGen}_{\mathsf{PAHE}}()$:
    - samples $\rho \xleftarrow{\$} \mathbb{F}_{q^m}$ such that $\rho^t = 1$.
    - samples $g_0 \xleftarrow{\$} \mathbb{F}_{q^m}$, $\boldsymbol{f} = (f_1, \ldots, f_w) \xleftarrow{\$} \mathcal{S}_w^w(\mathbb{F}_{q^m})$
    - defines $\tilde{F} = Vect_{\mathbb{F}_q}((\rho^j f_i)_{1 \le i \le w, 0 \le j \le t-1})$
    - computes $\tilde{\boldsymbol{f}} = (f_1, \ldots, f_{d_{\tilde{F}}}) \in \mathcal{S}_{d_{\tilde{F}}}^{d_{\tilde{F}}}(\mathbb{F}_{q^m})$ a basis of $\tilde{F}$ with $d_{\tilde{F}} = \dim(\tilde{F})$
    - defines $g_i = \rho^{-i} g_0$
    - checks that $\mathsf{rw}(f_1, \ldots, f_{d_{\tilde{F}}}, g_0, ..., g_{t-1}) = d_{\tilde{F}} + t$ (if not it restarts)
    - extends this vector into a basis $\boldsymbol{b} = (f_1, \ldots, f_{d_{\tilde{F}}}, g_0, \ldots, g_{m-d_{\tilde{F}}-1}) \in \mathcal{S}_m^m(\mathbb{F}_{q^m})$
    - defines $\boldsymbol{g} = (g_0, \ldots, g_{m-d_{\tilde{F}}-1})$
    - computes the matrix $\boldsymbol{B} = \mathbf{Mat}(\boldsymbol{b})$ and its transposed inverse $(\boldsymbol{B}^{-1})^T$
    - defines $\boldsymbol{D}$ as the last $m - d_{\tilde{F}}$ columns of $(\boldsymbol{B}^{-1})^T$
    - samples $\boldsymbol{s} \xleftarrow{\$} F^n$ with $F = \mathsf{supp}(\boldsymbol{f})$
    - returns $(\boldsymbol{sk} = (\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{D}, \boldsymbol{s}), \boldsymbol{pk} = \rho)$.
- $\mathsf{Encrypt}_{\mathsf{PAHE}}(\boldsymbol{sk}, (\boldsymbol{m}_0, \ldots, \boldsymbol{m}_{t-1}), \boldsymbol{r} \in \mathbb{F}_{q^m}^n \times \mathcal{M}_{w,n}(\mathbb{F}_q))$ with $\boldsymbol{m}_i \in \mathbb{F}_q^n$): notes $\boldsymbol{r} = (\boldsymbol{r}_1, \boldsymbol{R}_2)$, defines $\boldsymbol{u} = \boldsymbol{r}_1$ and $\boldsymbol{e} = \boldsymbol{f}\boldsymbol{R}_2$ and sets $\boldsymbol{v} = \boldsymbol{s} \cdot \boldsymbol{u} + \boldsymbol{e} + \hat{\boldsymbol{m}}$ with $\hat{\boldsymbol{m}} = \sum_{0 \le i < t} g_i \star \boldsymbol{m}_i \in \mathbb{F}_{q^m}^n$. Returns $\boldsymbol{ct} = (\boldsymbol{u}, \boldsymbol{v})$.
- $\mathsf{Decrypt}_{\mathsf{PAHE}}(\boldsymbol{sk}, \boldsymbol{ct} = (\boldsymbol{u}, \boldsymbol{v}))$: returns $(\boldsymbol{m}_0, \ldots, \boldsymbol{m}_{t-1})$ with $\boldsymbol{m}_i = (\boldsymbol{D}^{(i+1)})^T \mathbf{Mat}(\boldsymbol{v} - \boldsymbol{s} \cdot \boldsymbol{u})$.
- $\mathsf{Add}_{\mathsf{PAHE}} = \mathsf{Add}_{\mathsf{AHE}}$.
- $\mathsf{PtxtMul}_{\mathsf{PAHE}} = \mathsf{PtxtMul}_{\mathsf{AHE}}$.

Fig. 7. Description of the packed additive homomorphic scheme.

Actually, in the following we will consider that the equality is met for the above condition, i.e. $t(w+1) = m$, because it is the optimal setup to have the least number of bootstrapping ciphertexts.

Proposition 9 (Extension of AHE properties to PAHE). *Proposition 1, and Corollaries 1, 2 and 3 remain true when replacing* AHE *with* PAHE*. Proposition 2 remains true except for the definition of the cosets which are now* $\{\mathsf{Encrypt}_{\mathsf{PAHE}}(\boldsymbol{sk}, \mathbf{0})\} + \sum_{0 \le i < t} g_i \star \boldsymbol{m}_i$.

Proof. For proposition 1 we can follow the same proof noting that

$$\boldsymbol{v} - \boldsymbol{s} \cdot \boldsymbol{u} = \sum_{1 \le i \le w} f_i \star \boldsymbol{e}_i + \sum_{0 \le i < t} g_i \star \boldsymbol{m}_i.$$

We can thus write,

$$\begin{aligned}(\boldsymbol{D}^{(j+1)})^T \mathbf{Mat}(\boldsymbol{v} - \boldsymbol{s} \cdot \boldsymbol{u}) &= \sum_{1 \le i \le w} (\boldsymbol{D}^{(j+1)})^T \mathbf{vec}(f_i) \star \boldsymbol{e}_i + \sum_{0 \le i < t} (\boldsymbol{D}^{(j+1)})^T \mathbf{vec}(g_i) \star \boldsymbol{m}_i \\ &= \boldsymbol{m}_j\end{aligned}$$

using the same arguments as in the proof of proposition 1.

The proofs of proposition 2, and corollaries 1, 2 and 3 do not depend on the specific encoding we have on PAHE but only on how the noise vectors are chosen and used. As this is unchanged from AHE, the proofs are immediately valid for PAHE.

□

7.2 Plaintext rotation

In this section we define a rotation operation that is publicly computable and rotates plaintexts inside a packed ciphertext. It is described in Figure 8.

– $\mathit{Rotate}_{\mathsf{PAHE}}(\boldsymbol{ct} = (\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{pk} = \rho, j \in \mathbb{N})$ returns $(\rho^j \boldsymbol{u}, \rho^j \boldsymbol{v})$.

Fig. 8. Description of plaintext rotation.

Proposition 10 (Rotation correctness). *For any* $(\boldsymbol{m}_0, ..., \boldsymbol{m}_{t-1}) \in \mathbb{F}_q^{n\times t}$, *and a properly generated key* $(\boldsymbol{sk}, \boldsymbol{pk})$, *let* $\boldsymbol{ct} = (\boldsymbol{u}, \boldsymbol{v})$ *be obtained by* $\boldsymbol{ct} \overset{\$}{\leftarrow} \mathsf{Encrypt}_{\mathsf{PAHE}}(\boldsymbol{sk}, \boldsymbol{m}_0, ..., \boldsymbol{m}_{t+1})$, $\boldsymbol{ct}' \overset{\$}{\leftarrow} \mathit{Rotate}_{\mathsf{PAHE}}(\boldsymbol{ct}, \boldsymbol{pk}, j)$. *We have* $\mathsf{Decrypt}_{\mathsf{PAHE}}(\boldsymbol{sk}, \boldsymbol{ct}') = (\boldsymbol{m}_j, ..., \boldsymbol{m}_{j+t-1})$[2].

Proof. Like in the proof of Proposition 9, we have

$$\boldsymbol{v} - \boldsymbol{s}\cdot\boldsymbol{u} = \sum_{1\le i\le w} f_i \star \boldsymbol{e}_i + \sum_{0\le i<t} g_i \star \boldsymbol{m}_i.$$

By noting $\boldsymbol{ct}' = (\boldsymbol{u}', \boldsymbol{v}')$ and $\boldsymbol{pk} = \rho$ we get

$$\boldsymbol{v}' - \boldsymbol{s}\cdot\boldsymbol{u}' = \sum_{1\le i\le w} (\rho^j f_i) \star \boldsymbol{e}_i + \sum_{0\le i<t} (\rho^j g_i) \star \boldsymbol{m}_i.$$

Changing variables $i' = i - j$ in the second sum gives

$$\begin{aligned}\boldsymbol{v}' - \boldsymbol{s}\cdot\boldsymbol{u}' &= \sum_{1\le i\le w} (\rho^j f_i) \star \boldsymbol{e}_i + \sum_{0\le i'<t} (\rho^j g_{i'+j}) \star \boldsymbol{m}_{i+j} \\ &= \sum_{1\le i\le w} (\rho^j f_i) \star \boldsymbol{e}_i + \sum_{0\le i'<t} g_{i'} \star \boldsymbol{m}_{i+j}\end{aligned}$$

Now just like in the proof of Proposition 9, we can write for any $0 \le k < t$,

$$\begin{aligned}(\boldsymbol{D}^{(k+1)})^T\mathbf{Mat}(\boldsymbol{v}' - \boldsymbol{s}\cdot\boldsymbol{u}') &= \sum_{1\le i\le w} (\boldsymbol{D}^{(k+1)})^T\mathbf{vec}(\rho^j f_i) \star \boldsymbol{e}_i + \sum_{0\le i'<t} (\boldsymbol{D}^{(k+1)})^T\mathbf{vec}(g_{i'}) \star \boldsymbol{m}_{i+j} \\ &= \boldsymbol{m}_{k+j}\end{aligned}$$

because thanks to the definition of $\boldsymbol{D} = (\boldsymbol{B}^{-1})^T$, for any $1 \le i \le w$, $(\boldsymbol{D}^{(k+1)})^T\mathbf{vec}(\rho^j f_i) = 0$, $(\boldsymbol{D}^{(k+1)})^T\mathbf{vec}(g_k) = 1$ and for any $0 \le i' < t, i' \neq k$, $(\boldsymbol{D}^{(k+1)})^T\mathbf{vec}(g_{i'}) = 0$. □

Remark 5. In particular, $\mathit{Rotate}_{\mathsf{PAHE}}(\cdot, \boldsymbol{pk}, t) = \mathit{id}$.

[2] The indexes are taken modulo t

7.3 Homomorphic decrytion with packing

The new relinearization with packing is presented in Figure 9. The key switching material (ksk and projk) is now composed of $2(w+1)$ packed ciphertexts (instead of $2m$ simple ciphertexts in Figure 5).

- $\mathsf{GenKeySwitch}_{\mathsf{PAHE}}(\boldsymbol{sk}_1 = (\boldsymbol{f}_1, \boldsymbol{g}_1, \boldsymbol{D}_1, \boldsymbol{s}_1))$:
  - generates a new basis $\boldsymbol{b}_2$ of $\mathbb{F}_{q^m}$, $\boldsymbol{sk}_2 = (\boldsymbol{f}_2, \boldsymbol{g}_2, \boldsymbol{D}_2, \boldsymbol{s}_2)$ and $\boldsymbol{pk}_2$ as in $\mathsf{KeyGen}_{\mathsf{PAHE}}$
  - for $1 \le i \le m$ defines $\boldsymbol{s}_{1,i} = \boldsymbol{d}^T\mathbf{Mat}(\gamma_i \star \boldsymbol{s}_1)$ with $\boldsymbol{d} = \boldsymbol{D}_1^{(1)}$
  - defines $\mathsf{ksk} = (\mathsf{ksk}_0, \ldots, \mathsf{ksk}_w)$, with $\mathsf{ksk_i} \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{PAHE}}(\boldsymbol{sk}_2, (\boldsymbol{s}_{1,it+j})_{1 \le j \le t})$
  - for $1 \le i \le m$ defines $\boldsymbol{p}_i = \boldsymbol{d}^{(i)} \star (1, 0, \ldots, 0) \in \mathbb{F}_{q^m}^n$
  - defines $\mathsf{projk} = (\mathsf{projk}_0, \ldots, \mathsf{projk}_w)$ with $\mathsf{projk}_i \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{PAHE}}(\boldsymbol{sk}_2, (\boldsymbol{p}_{it+j})_{1 \le j \le t}))$
  - returns $(\boldsymbol{sk}_2, \boldsymbol{pk}_2, \mathsf{ksk}, \mathsf{projk})$.

- $\mathsf{HomDecrypt}_{\mathsf{PAHE}}(\mathsf{ksk}, \mathsf{projk}, \boldsymbol{ct} = (\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{pk}_2)$:
  - defines $\boldsymbol{u}_i \in \mathbb{F}_q^n$ such that $\boldsymbol{u} = \sum_{1 \le i \le m} \gamma_i \star \boldsymbol{u}_i$
  - defines $\boldsymbol{v}_i \in \mathbb{F}_q^n$ such that $\boldsymbol{v} = \sum_{1 \le i \le m} \gamma_i \star \boldsymbol{v}_i$
  - computes $\boldsymbol{ct}_1 = \sum_{1 \le i \le m} \boldsymbol{v}_i \cdot \mathit{Rotate}_{\mathsf{PAHE}}(\mathsf{projk}_{\lfloor (i-1)/t \rfloor}, \boldsymbol{pk}_2, (i-1))$
  - computes $\boldsymbol{ct}_2 = \sum_{1 \le i \le m} \boldsymbol{u}_i \cdot \mathit{Rotate}_{\mathsf{PAHE}}(\mathsf{ksk}_{\lfloor (i-1)/t \rfloor}, \boldsymbol{pk}_2, (i-1))$
  - returns $\boldsymbol{ct}_1 - \boldsymbol{ct}_2$

Fig. 9. Homomorphic Decryption Packing Algorithms.

The following proposition establishes that a ciphertext $\boldsymbol{ct}$ encrypting a **single** plaintext $\boldsymbol{m}$ (without packing), after homomorphic decryption with a PAHE key switching material, decrypts correctly.

Proposition 11 (Homomorphic Decryption with packing Correctness). *For any properly generated key* $\boldsymbol{sk}_1$ *and* $\boldsymbol{m} \in \mathbb{F}_q^n$, $\boldsymbol{ct} \xleftarrow{\$} \mathsf{Encrypt}_{\mathsf{AHE}}(\boldsymbol{sk}_1, \boldsymbol{m})$, $(\boldsymbol{sk}_2, \boldsymbol{pk}_2, \mathsf{ksk}, \mathsf{projk}) \xleftarrow{\$} \mathsf{GenKeySwitch}_{\mathsf{PAHE}}(\boldsymbol{sk}_1)$, $\mathsf{Decrypt}_{\mathsf{PAHE}}(\boldsymbol{sk}_2, \mathsf{HomDecrypt}_{\mathsf{PAHE}}(\mathsf{ksk}, \mathsf{projk}, \boldsymbol{ct}, \boldsymbol{pk}_2)) = \boldsymbol{m}$.

Proof. Let $1 \le i \le m$. By Proposition 10,

$$\mathit{Rotate}_{\mathsf{PAHE}}(\mathsf{projk}_{\lfloor (i-1)/t \rfloor}, \boldsymbol{pk}_2, (i-1)) \in \{\mathsf{Encrypt}_{\mathsf{PAHE}}(\boldsymbol{sk}_2, (\boldsymbol{p}_{\alpha(i,j)})_{1 \le j \le t})\}.$$

where

$$\begin{aligned} \alpha : [1, m] \times [1, t] &\longrightarrow [1, m] \\ (i, j) &\longmapsto \lfloor \frac{i-1}{t} \rfloor t + 1 + ((i + j - 2) \mod t). \end{aligned}$$

Similarly,

$$\mathit{Rotate}_{\mathsf{PAHE}}(\mathsf{ksk}_{\lfloor (i-1)/t \rfloor}, \boldsymbol{pk}_2, (i-1)) \in \{\mathsf{Encrypt}_{\mathsf{PAHE}}(\boldsymbol{sk}_2, (\boldsymbol{s}_{1,\alpha(i,j)})_{1 \le j \le t})\}.$$

Using Proposition 9

$$(\boldsymbol{ct}_1 - \boldsymbol{ct}_2) \in \{\mathsf{Encrypt}_{\mathsf{PAHE}}(\boldsymbol{sk}_2, \sum_{1 \leq i \leq m} (\boldsymbol{v}_i \cdot \boldsymbol{p}_{\alpha(i,j)} - \boldsymbol{u}_i \cdot \boldsymbol{s}_{1,\alpha(i,j)})_{1 \leq j \leq t})\}.$$

Using $\mathsf{Decrypt}_{\mathsf{AHE}}$ on ciphertext $(\boldsymbol{ct}_1 - \boldsymbol{ct}_2)$ corresponds to retrieving the first component of the packed plaintext hence

$$\mathsf{Decrypt}_{\mathsf{AHE}}(\boldsymbol{sk}_2, \mathsf{HomDecrypt}_{\mathsf{PAHE}}(\mathsf{ksk}, \mathsf{projk}, \boldsymbol{ct}, \boldsymbol{pk}_2)) = \sum_{1 \leq i \leq m} \boldsymbol{v}_i \cdot \boldsymbol{p}_{\alpha(i,1)} - \boldsymbol{u}_i \cdot \boldsymbol{s}_{1,\alpha(i,1)}.$$

Noting that for all $1 \leq i \leq m$, we have the identity $\alpha(i,1) = i$,

$$\begin{aligned}\mathsf{Decrypt}_{\mathsf{AHE}}(\boldsymbol{sk}_2, \mathsf{HomDecrypt}_{\mathsf{PAHE}}(\mathsf{ksk}, \mathsf{projk}, \boldsymbol{ct}, \boldsymbol{pk}_2)) &= \sum_{1 \leq i \leq m} \boldsymbol{v}_i \cdot \boldsymbol{p}_i - \boldsymbol{u}_i \cdot \boldsymbol{s}_{1,i} \\ &= \boldsymbol{m}\end{aligned}$$

using the same argument as in the proof of Proposition 6. □

Remark 6. After homomorphic decryption, the resulting ciphertext $\boldsymbol{ct}' = \mathsf{HomDecrypt}_{\mathsf{PAHE}}(\mathsf{ksk}, \mathsf{projk}, \boldsymbol{ct}, \boldsymbol{pk}_2)$ **does not** belong to the distribution $\mathsf{Encrypt}_{\mathsf{AHE}}(\boldsymbol{sk}_2, \boldsymbol{m})$ nor $\mathsf{Encrypt}_{\mathsf{PAHE}}(\boldsymbol{sk}_2, (\boldsymbol{m}, \cdot, \dots))$. Indeed, the noise component of $\boldsymbol{ct}'$ lives in the bigger space $\tilde{F}$ (as defined in $\mathsf{KeyGen}_{\mathsf{PAHE}}$), whereas the noise component of the freshly encrypted ciphertexts belong to F. It does not impact security since $\boldsymbol{ct}'$ results from public operations only with fresh ciphertexts $\boldsymbol{ct}$, ksk and projk.

Remark 7. Because the $\mathsf{HomDecrypt}_{\mathsf{PAHE}}$ operation consists essentially in a homomorphic evaluation of the orthogonal projection of $\boldsymbol{v} - \boldsymbol{s} \cdot \boldsymbol{u}$ on $\langle g_0 \rangle_{\mathbb{F}_q}$, note that the initial ciphertext $\boldsymbol{ct}$ does not need to be fresh (i.e. with its noise in F). In particular, Proposition 11 is still valid when $\boldsymbol{ct}$ results from a previous homomorphic decryption with another set of keys.

Remark 8. If the initial ciphertext $\boldsymbol{ct}$ had been taken with packed plaintexts (i.e. drawn from $\mathsf{Encrypt}_{\mathsf{PAHE}}$ instead of $\mathsf{Encrypt}_{\mathsf{AHE}}$), the homomorphic decryption would be correct only for the first component of the packed plaintext. Other components would be lost.

8 Parameters

In this section we give example parameters for our scheme. Several sets are proposed for different values of d, the number of possible multiplications in the SHE. The parameter selection ran through the following steps. For given m and n, the weight w is set to the minimum between the half-rate rank Gilbert-Varshamov bound d_{rGV} and w^{d-1} (to prevent an overflow on the noise after d multiplications). We search for the lowest n such that a sufficient number of ciphertexts $\ell = 3w/4$ can be published, i.e. the best attacks against IRSD in an s-ideal $[sn, n]_{q^m}$ random code for every $2 \leq s \leq \ell$ are above the security level. Two attacks against IRSD were taken into account:

- the combinatorial attack from [8] whose complexity is given by $(sn - n)^{\omega} m^{\omega} q^{w\lceil \frac{m(n+1)}{sn} \rceil - m}$ for ω the linear algebra exponent;
- the algebraic attack from [10] whose complexity is given by $q^{aw} m \binom{sn-n-1}{w} \binom{sn-a}{w}^{\omega - 1}$ where a is defined as the smallest integer such that the condition $m\binom{sn-n-1}{w} \geq \binom{sn-a}{w} - 1$ is fulfilled.

If no such n can be found, the process restarts with an increased m. The SageMath script of our parameter selection is available at:

https://www.github.com/victordyseryn/rank-fhe-parameter-selection

The key and ciphertext sizes in bits are given by the following formulas:

$$
\begin{aligned}
|\boldsymbol{sk}| &= \log_2(q)(m^2 + nw) \\
|\boldsymbol{ct}| &= 2\log_2(q)mn
\end{aligned}
$$

The approximate timings for addition, multiplication and bootstrapping operations are estimated as follows:

- The addition consists in the sum of two vectors in $\mathbb{F}_{q^m}^n$, the number of bit operations is then
$$\mathcal{T}_{\text{Add}} = 2mn;$$
- The multplication consists in three multiplications of vectors in $\mathbb{F}_{q^m}^n$. The use of the Karatsuba algorithm gives a number of bits operations of
$$\mathcal{T}_{\text{Mul}} = 3(mn)^{1.6};$$
- The bootstrapping requires $2m$ plaintext absorptions, i.e. a multiplication of a vector in $\mathbb{F}_q^n$ times a vector in $\mathbb{F}_{q^m}^n$. With Karatsuba algorithm, the number of bit operations of a plaintext absorption is $mn^{1.6}$, hence the total cost of bootstrapping is
$$\mathcal{T}_{\text{Bootstrap}} = 2m^2n^{1.6}.$$
The bootstrapping time is informative only as it is unsecure as proven in Section 6.

The timing in milliseconds are then computed by diving the number of bit operations by 3 millions, accounting for a processor running at 3 GHz. Note that this is an extremely conservative estimation, as many modern processors run several bit operations in one clock cycle.

Our parameters are presented in the following table:

d	q	m	n	w	ℓ	Security	Key size	$\boldsymbol{ct}$ size	Add	Mul	Bootstrap
1	2	172	20	13	9	128	3.7 kB	0.9 kB	0.002 ms	0.5 ms	2 ms
2	2	367	183	7	5	128	17.0 kB	16.8 kB	0.04 ms	52 ms	374 ms
3	2	1296	314	6	4	128	210 kB	102 kB	0.3 ms	944 ms	11 s
4	2	3125	713	5	3	128	1.22 MB	557 kB	1 ms	14.3 s	239 s

Table 1. Example of paramaters for our SHE scheme, with associated sizes and execution timings. d is the number of possible multiplications. q, m and n are parameters of the rank linear code and w is the rank weight of the error. ℓ is the number of independant ciphertexts that can be published.

The sizes and expected performance of our somewhat homomorphic encryption scheme are very positive, and could already be used for practical applications with a small number of multiplications.

These numbers additionally show a strong potential regarding bootstrapping, should it be repaired. With bootstrapping enabled, the only parameters to consider would be those for which $d = 1$, and in such a context bootstrapping would be very efficient with an unoptimized running time of only 2 milliseconds. For example, it is more than 6 times more efficient than the bootstrap in TFHE [15], one of the most popular and widely used lattice-based FHE framework. Our scheme shows that error correcting codes can lead to very competitive FHE constructions.

References

[1] Aguilar Melchor, C., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Zémor, G., Couvreur, A., Hauteville, A.: RQC. Technical report, National Institute of Standards and Technology (2019) available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions. 2, 3, 8

[2] Aguilar-Melchor, C., Aragon, N., Dyseryn, V., Gaborit, P., Zémor, G.: LRPC codes with multiple syndromes: near ideal-size KEMs without ideals. In: International Conference on Post-Quantum Cryptography, Springer (2022) 45–68 2

[3] Alekhnovich, M.: More on average case vs approximation complexity. In: 44th FOCS, IEEE Computer Society Press (October 2003) 298–307 3

[4] Aragon, N., Blazy, O., Deneuville, J.C., Gaborit, P., Hauteville, A., Ruatta, O., Tillich, J.P., Zémor, G., Aguilar Melchor, C., Bettaieb, S., Bidoux, L., Bardet, M., Otmani, A.: ROLLO. Technical report, National Institute of Standards and Technology (2019) available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions. 2, 8

[5] Aragon, N., Blazy, O., Gaborit, P., Hauteville, A., Zémor, G.: Durandal: A rank metric based signature scheme. In Ishai, Y., Rijmen, V., eds.: EUROCRYPT 2019, Part III. Volume 11478 of LNCS., Springer, Heidelberg (May 2019) 728–758 2

[6] Aragon, N., Dyseryn, V., Gaborit, P., Loidreau, P., Renner, J., Wachter-Zeh, A.: LowMS: a new rank metric code-based KEM without ideal structure. Cryptology ePrint Archive, Report 2022/1596 (2022) https://eprint.iacr.org/2022/1596. 2

[7] Aragon, N., Gaborit, P., Hauteville, A., Ruatta, O., Zémor, G.: Low rank parity check codes: New decoding algorithms and applications to cryptography. IEEE Transactions on Information Theory **65**(12) (2019) 7697–7717 *2, 7*

[8] Aragon, N., Gaborit, P., Hauteville, A., Tillich, J.P.: A new algorithm for solving the rank syndrome decoding problem. In: 2018 IEEE International Symposium on Information Theory, ISIT 2018, Vail, CO, USA, June 17-22, 2018, IEEE (2018) 2421–2425 *24*

[9] Armknecht, F., Augot, D., Perret, L., Sadeghi, A.R.: On constructing homomorphic encryption schemes from coding theory. In: IMA International Conference on Cryptography and Coding, Springer (2011) 23–40 *2*

[10] Bardet, M., Bros, M., Cabarcas, D., Gaborit, P., Perlner, R.A., Smith-Tone, D., Tillich, J.P., Verbel, J.A.: Improvements of algebraic attacks for solving the rank decoding and MinRank problems. In: ASIACRYPT 2020, Part I. LNCS, Springer, Heidelberg (December 2020) 507–536 *24*

[11] Bidoux, L., Chi-Domínguez, J.J., Feneuil, T., Gaborit, P., Joux, A., Rivain, M., Vinçotte, A.: RYDE: A Digital Signature Scheme based on Rank-Syndrome-Decoding Problem with MPCitH Paradigm. arXiv preprint arXiv:2307.08726 (2023) *2*

[12] Bogdanov, A., Lee, C.H.: Homomorphic encryption from codes. Cryptology ePrint Archive, Report 2011/622 (2011) https://eprint.iacr.org/2011/622. *2*

[13] Boneh, D., Goh, E.J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Theory of Cryptography: Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005. Proceedings 2, Springer (2005) 325–341 *1*

[14] Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. SIAM Journal on computing **43**(2) (2014) 831–871 *1*

[15] Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: Fast fully homomorphic encryption over the torus. Journal of Cryptology **33**(1) (January 2020) 34–91 *1, 25*

[16] Cho, J., Kim, Y.S., No, J.S.: Homomorphic computation in reed-muller codes. Cryptology ePrint Archive, Report 2020/565 (2020) https://eprint.iacr.org/2020/565. *2*

[17] Debris-Alazard, T., Tillich, J.P.: Two attacks on rank metric code-based schemes: RankSign and an IBE scheme. In Peyrin, T., Galbraith, S., eds.: ASIACRYPT 2018, Part I. Volume 11272 of LNCS., Springer, Heidelberg (December 2018) 62–92 *2*

[18] Ducas, L., Micciancio, D.: FHEW: Bootstrapping homomorphic encryption in less than a second. In Oswald, E., Fischlin, M., eds.: EUROCRYPT 2015, Part I. Volume 9056 of LNCS., Springer, Heidelberg (April 2015) 617–640 *1*

[19] ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory **31**(4) (1985) 469–472 *1*

[20] Gaborit, P., Hauteville, A., Phan, D.H., Tillich, J.P.: Identity-based encryption from codes with rank metric. In Katz, J., Shacham, H., eds.: CRYPTO 2017, Part III. Volume 10403 of LNCS., Springer, Heidelberg (August 2017) 194–224 *2*

[21] Gaborit, P., Ruatta, O., Schrek, J.: On the Complexity of the Rank Syndrome Decoding Problem. IEEE Trans. Inf. Theory **62**(2) (2016) 1006–1019 *18*

[22] Gauthier, V., Otmani, A., Tillich, J.P.: A distinguisher-based attack of a homomorphic encryption scheme relying on reed-solomon codes. Cryptology ePrint Archive, Report 2012/168 (2012) https://eprint.iacr.org/2012/168. *2*

[23] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the forty-first annual ACM symposium on Theory of computing. (2009) 169–178 *1, 2*

[24] Goldwasser, S., Micali, S.: Probabilistic encryption and how to play mental poker keeping secret all partial information. In: 14th ACM STOC, ACM Press (May 1982) 365–377 *1*

[25] Hauteville, A., Tillich, J.P.: New algorithms for decoding in the rank metric and an attack on the LRPC cryptosystem (2015) abs/1504.05431. *8*

[26] Overbeck, R.: Structural attacks for public key cryptosystems based on Gabidulin codes. Journal of Cryptology **21**(2) (April 2008) 280–301 *2*

[27] Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In Stern, J., ed.: EUROCRYPT'99. Volume 1592 of LNCS., Springer, Heidelberg (May 1999) 223–238 *1*

[28] Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. Foundations of secure computation **4**(11) (1978) 169–180 *1*

[29] Sendrier, N.: Decoding one out of many. In: Post-Quantum Cryptography 2011. Volume 7071 of LNCS. (2011) 51–67 *8*

[30] Sidelnikov, V.M., Shestakov, S.O.: On insecurity of cryptosystems based on generalized Reed-Solomon codes. (1992) *2*