

Designing homomorphic encryptions with rational functions

Gerald Gavin¹ and Sandrine Tainturier²

¹ Laboratory ERIC - University of Lyon

`gerald.gavin@univ-lyon1.fr`

² Adecco - Geneve

`sandrine-tainturier@orange.fr`

Abstract. In [Gav16], [GB19] and [GT20], new ideas to build homomorphic encryption schemes have been presented. The authors propose private-key encryption schemes whose secret key is a rational function ϕ/ϕ' . By construction, these schemes are not homomorphic. To get homomorphic properties, nonlinear homomorphic operators are derived from the secret key. In [GT20], an additive homomorphic encryption is proposed. In this paper, we adopt the same approach to build a HE based on the same private-key encryption scheme. We obtain a multivariate encryption scheme in the sense that the knowledge of the CPA attacker can be turned into an over-defined system of nonlinear equations (contrarily to LWE-based encryptions). The factoring assumption is introduced in order to make a large class of algebraic attacks (based on Gröebner bases) irrelevant. We extensively analyze the security of our scheme against algebraic attacks. In particular, we exhibit the fundamental role played by symmetry in these attacks. We also formally show that some of these attacks are exponential-time. While we did not propose a formal security proof relying on a classical cryptographic assumption, we hopefully provide convincing evidence for security.

1 Introduction

The prospect of outsourcing an increasing amount of data storage and management to cloud services raises many new privacy concerns for individuals and businesses alike. The privacy concerns can be satisfactorily addressed if users encrypt the data they send to the cloud. If the encryption scheme is homomorphic, the cloud can still perform meaningful computations on the data, even though it is encrypted.

The theoretical problem of constructing a fully homomorphic encryption scheme (HE) supporting arbitrary functions f , was only recently solved by the breakthrough work of Gentry [Gen09]. More recently, further fully homomorphic schemes were presented [SS10],[vDGHV10], [CNT12], [GHS12a],[GSW13] following Gentry's framework. The underlying tool behind all these schemes is the use of Euclidean lattices, which have previously proved powerful for devising many cryptographic primitives. A central aspect of Gentry's fully homomorphic scheme (and the subsequent schemes) is the ciphertext refreshing *Recrypt* operation. Even if many improvements have been made in one decade, this operation remains very costly [LNV11], [GHS12b], [DM15], [CGGI18]. Indeed, bootstrapped bit operations are still about one billion times slower than their plaintext equivalents (see [CGGI18]).

We adopt a recent approach developed in [Gav16], [GB19], [GT20] where the secret key is a (multivariate) rational function ϕ_S/ϕ'_S . A ciphertext is here a randomly chosen vector \mathbf{c} satisfying $\phi_S/\phi'_S(\mathbf{c}) = x$. In particular, an encryption \mathbf{c} of 0 satisfied $\phi_S(\mathbf{c}) = 0$. It follows that the expanded representations of ϕ_S should not be polynomial-size (otherwise the CPA attacker could recover it by solving a polynomial-size linear system). In order to get polynomial-time encryptions and decryptions, ϕ_S/ϕ'_S should be written in a compact form, e.g. a factored or semi-factored form. By construction, the generic cryptosystem described above is not homomorphic in the sense that the vector sum is not a homomorphic operator. To get homomorphic properties, *ad hoc* nonlinear homomorphic operators *Add* and *Mult* (sometimes denoted by \oplus or \otimes) will be derived from the secret key.

1.1 A "mathematical" approach

Craig Gentry advocates two ways to build homomorphic encryptions (HE) in his invited talk presented at Eurocrypt 2021 [Gen21]: the *cryptographic way* and the *mathematical way*. The starting point of the

cryptographic way is an established cryptographic assumption. A HE based on this assumption is then built when possible. Most of existing HE follow this way. Nevertheless, after a decade of research, no HE proposed in the literature is really efficient despite much improvement. It is maybe time to explore the second way.

In this paper, we adopt the *mathematical way*. This approach consists of exploring new cryptographic assumptions from which it is known in advance that efficient HE can be derived. The main drawback of this approach is that the HE obtained has not stood the test of time. Clearly, such approaches are relevant only if the authors succeed in convincing the community that the obtained HE may be secure. This is the main purpose of this paper.

As advocated by Gentry, the starting point of our HE is a ring homomorphism on which the decryption function will be based. For concreteness, we consider the pseudo-ring $\mathcal{R} = (\mathbf{Z}_n \times \mathbf{Z}_n^*, +, \times)$ defined as follows:

- $(a, b) + (a', b') = (ab' + a'b, bb')$
- $(a, b) \times (a', b') = (aa', bb')$

It is a pseudo-ring, and not a ring, because ' \times ' is not distributive *w.r.t.* ' $+$ '. It is important to notice that ' $+$ ' is not the vector sum. This will be fundamental in our construction. Our HE is derived from the basic pseudo-ring homomorphism $\delta : \mathcal{R} \rightarrow \mathbf{Z}_n$ defined by

$$\delta(a, b) = a/b$$

Thanks to this basic pseudo-ring homomorphism, we will develop a HE whose homomorphic operators \oplus and \otimes satisfy the two following fundamental properties:

- \oplus and \otimes are both nonlinear¹.
- (\mathcal{C}, \oplus) is not a group.

The first property is necessary to be robust against linear algebra. As noticed by Gentry, the second one is necessary to make subgroup attacks irrelevant. As the starting point of our HE is not an established cryptographic assumption, it is not inherently secure. Nevertheless, breaking our HE consists of solving a polynomial system, which was shown to be \mathcal{NP} -hard [Gar97]. This worst-case result is obviously not sufficient, but it does not close the door on the possibility of reaching security.

Why read this paper? We attempt to design a noise-free HE. As mentioned above, we do not propose a complete security proof based on an established cryptographic assumption. Nevertheless, we propose a security analysis, which may at least convince the reader that our scheme is not trivially insecure. In this sense, this paper can be seen as a nice cryptanalysis challenge. If vulnerabilities are discovered, it may be possible to reuse the underlying ideas of this scheme. Otherwise, it would close the door on such approaches, enhancing the commonly accepted idea that noise is necessary to build HE.

Assume now that our HE is secure. It would be conceptually exciting to see that noise-free HE can be designed. Moreover, our scheme deals only with elementary (high school) algebraic notions. This makes it easy to understand and implement. Nevertheless, its efficiency makes it not really practical: homomorphic operations are around hundreds of millions slower than plaintext ones. However, we are fully confident in the community's ability to provide improvements and developments.

1.2 Overview of the private-key encryption

We design a private-key encryption scheme where the secret key is a randomly chosen invertible $2\kappa - by - 2\kappa$ matrix S defined over \mathbf{Z}_n , n being a RSA modulus. Encrypting $x \in \mathbf{Z}_n$ simply consists of randomly choosing a vector \mathbf{c} satisfying

$$\frac{\langle \mathbf{s}_1, \mathbf{c} \rangle}{\langle \mathbf{s}_2, \mathbf{c} \rangle} + \dots + \frac{\langle \mathbf{s}_{2\kappa-1}, \mathbf{c} \rangle}{\langle \mathbf{s}_{2\kappa}, \mathbf{c} \rangle} = x \tag{1}$$

¹ In particular, \oplus is not the vector sum.

where \mathbf{s}_i refers to the i^{th} row of S . Equivalently, one can randomly choose $x_1, \dots, x_\kappa, r_1, \dots, r_\kappa$ in \mathbf{Z}_n s.t. $x_1 + \dots + x_\kappa = x$ and output

$$\mathbf{c} = S^{-1} \begin{pmatrix} r_1 x_1 \\ r_1 \\ \dots \\ r_\kappa x_\kappa \\ r_\kappa \end{pmatrix}$$

Throughout this paper, we will use the following convenient notation to encapsulate the internal randomness of encryptions:

$$X(\mathbf{c}) \stackrel{\text{def}}{=} (x_1, \dots, x_\kappa)$$

$$R(\mathbf{c}) \stackrel{\text{def}}{=} (r_1, \dots, r_\kappa)$$

Clearly, \mathbf{c} is an encryption of 0 if and only if

$$\phi_S(\mathbf{c}) \stackrel{\text{def}}{=} \sum_{\ell=1}^{\kappa} \langle \mathbf{s}_{2\ell-1}, \mathbf{c} \rangle \prod_{\ell' \neq \ell} \langle \mathbf{s}_{2\ell'}, \mathbf{c} \rangle = 0$$

A basic attack of this scheme consists of recovering the monomial coefficients of ϕ_S by solving a linear system. The key idea of our construction is that the expanded representation of ϕ_S is exponential-size (and thus cannot be recovered) provided

$$\kappa = \Theta(\lambda)$$

Surprisingly, a link with the famous problem **LWE** (see [Reg05]) can be established and the basic attack can be seen as an adaptation of the Arora et Ge attack [AG11a].

Nevertheless, this private-key encryption is not homomorphic. More precisely, the vector sum is not a homomorphic operator. Indeed, for any $a \in \mathbf{Z}_n^*$, $a\mathbf{c}$ and \mathbf{c} are encryptions of the same value. To get homomorphic properties, additional material derived from the secret key should be publicized.

1.3 Overview of the homomorphic operators

The (basic) operator **Add** (also denoted by \mathbf{O}_0 or \oplus in this paper) exactly follows the one considered in [GT20]. It simply exploits the following basic equality

$$\frac{a}{b} + \frac{a'}{b'} = \frac{ab' + a'b}{bb'}$$

This operator consists of evaluating 2κ quadratic variate- 2κ polynomials $p_1, \dots, p_{2\kappa}$, i.e.

$$\mathbf{c} \oplus \mathbf{c}' = (p_1(\mathbf{c}, \mathbf{c}'), \dots, p_{2\kappa}(\mathbf{c}, \mathbf{c}'))$$

A high-level description of this operator is proposed in Figure 1 (see Appendix A for a complete description of this operator in the case $\kappa = 1$. As an exercise, the reader can try to define the polynomials $p_1, \dots, p_{2\kappa}$ when S is the identity matrix).

The (basic) operator **Mult** is based on the following equality

$$\frac{a}{b} \times \frac{a'}{b'} = \frac{aa'}{bb'}$$

Nevertheless, its implementation is a little bit more complex. It cannot be achieved by applying only one quadratic operator. Indeed, it exploits the following equality

$$xx' = \sum_{i=1}^{\kappa} \sum_{j=1}^{\kappa} x_i x'_j$$

$$\mathbf{O}_0 \left(S^{-1} \begin{pmatrix} r_1 x_1 \\ r_1 \\ \dots \\ r_\kappa x_\kappa \\ r_\kappa \end{pmatrix}, S^{-1} \begin{pmatrix} r'_1 x'_1 \\ r'_1 \\ \dots \\ r'_\kappa x'_\kappa \\ r'_\kappa \end{pmatrix} \right) = S^{-1} \begin{pmatrix} r_1 r'_1 (x_1 + x'_1) \\ r_1 r'_1 \\ \dots \\ r_\kappa r'_\kappa (x_\kappa + x'_\kappa) \\ r_\kappa r'_\kappa \end{pmatrix}$$

Fig. 1. Description of the basic operator $\text{Add} = \mathbf{O}_0$. This operator is nonlinear (quadratic) and ensures that $X(\mathbf{c} \oplus \mathbf{c}') = (x_1 + x'_1, \dots, x_\kappa + x'_\kappa)$ and $R(\mathbf{c} \oplus \mathbf{c}') = (r_1 r'_1, \dots, r_\kappa r'_\kappa)$

It follows that at least κ quadratic operators are necessary to *store* all the products $x_i x'_j$ in some intermediate vectors: each operator outputs a vector storing at most κ products. In the basic implementation of Mult , this is achieved by applying exactly κ quadratic operators $\mathbf{O}_1, \dots, \mathbf{O}_\kappa$. For concreteness, $\mathbf{O}_i(\mathbf{c}, \mathbf{c}')$ outputs an encryption of

$$\pi_i = x_1 x'_i + x_2 x'_{i+1} \dots + x_\kappa x'_{i-1}$$

A high-level description of the basic operator \mathbf{O}_i is given in Figure 2. As $\pi_1 + \dots + \pi_\kappa = xx'$, it then suffices to homomorphically add these vectors, i.e.

$$\text{Mult}(\mathbf{c}, \mathbf{c}') = \mathbf{O}_1(\mathbf{c}, \mathbf{c}') \oplus \dots \oplus \mathbf{O}_\kappa(\mathbf{c}, \mathbf{c}')$$

$$\mathbf{O}_i \left(S^{-1} \begin{pmatrix} r_1 x_1 \\ r_1 \\ \dots \\ r_\kappa x_\kappa \\ r_\kappa \end{pmatrix}, S^{-1} \begin{pmatrix} r'_1 x'_1 \\ r'_1 \\ \dots \\ r'_\kappa x'_\kappa \\ r'_\kappa \end{pmatrix} \right) = S^{-1} \begin{pmatrix} r_1 r'_i x_1 x'_i \\ r_1 r'_i \\ r_2 r'_{i+1} x_2 x'_{i+1} \\ r_2 r'_{i+1} \\ \dots \\ r_\kappa r'_{i-1} x_\kappa x'_{i-1} \\ r_\kappa r'_{i-1} \end{pmatrix}$$

Fig. 2. Description of the basic operator \mathbf{O}_i . This operator is nonlinear (quadratic) and ensures that $X(\mathbf{O}_i(\mathbf{c}, \mathbf{c}')) = (x_1 x'_i, \dots, x_\kappa x'_{i-1})$ and $R(\mathbf{O}_i(\mathbf{c}, \mathbf{c}')) = (r_1 r'_i, \dots, r_\kappa r'_{i-1})$

We obtain a HE, which is unfortunately insecure. It is subject to attacks called *attacks by linearization*, i.e. attacks that do not require solving systems of nonlinear equations.

Example of attack by linearization. *The vector $\mathbf{v} = S^{-1}(0, 1, 0, 1, \dots, 0, 1)$ is the unique vector satisfying $\text{Add}(\mathbf{u}, \mathbf{v}) = \mathbf{u}$ for any valid encryption \mathbf{u} . It can be recovered by solving a size- 2κ linear system. It follows that $\mathbf{O}_1(\mathbf{c}, \mathbf{v})$ is equal to the vector $\mathbf{w} = S^{-1}(0, r_1, 0, r_2, \dots, 0, r_\kappa)$. By solving the size- 2κ linear system of equations $\text{Add}(\tilde{\mathbf{c}}, \mathbf{w}) = \mathbf{c}$, we get the vector*

$$\tilde{\mathbf{c}} = S^{-1} \begin{pmatrix} x_1 \\ 1 \\ \dots \\ x_\kappa \\ 1 \end{pmatrix}$$

satisfying $\langle \tilde{\mathbf{s}}, \tilde{\mathbf{c}} \rangle = x$ with $\tilde{\mathbf{s}} = \mathbf{s}_1 + \mathbf{s}_3 + \dots + \mathbf{s}_{2\kappa-1}$. By sampling \mathbf{c} , $\tilde{\mathbf{s}}$ can be efficiently recovered by solving a linear system of equations, breaking the security of our encryption scheme.

Other attacks by linearization are given in Appendix B. To make these attacks irrelevant, we propose to *randomize* the generation of the homomorphic operators. The underlying idea is very simple. The first step consists of slightly modifying the above operators: instead of outputting encryptions relevant under S , they

output encryptions relevant under randomly chosen matrices T_i . For instance, the operator O_1 becomes $O_1^{S \rightarrow T_1}$ satisfying

$$\mathbf{c}'' = O_1^{S \rightarrow T_1}(\mathbf{c}, \mathbf{c}') = T_1^{-1} \begin{pmatrix} r_1 r'_1 x_1 x'_1 \\ r_1 r'_1 \\ \dots \\ r_\kappa r'_\kappa x_\kappa x'_\kappa \\ r_\kappa r'_\kappa \end{pmatrix}$$

By construction, it is ensured that $\text{Decrypt}(T_1, \mathbf{c}'') = \pi_1$.

We then develop operators $\text{Rand}^{T_i \rightarrow S}$ allowing to obtain a *randomized* encryption \mathbf{c}''' valid under S satisfying $\text{Decrypt}(S, \mathbf{c}''') = \text{Decrypt}(T_i, \mathbf{c}'')$. Ideally,

$$\mathbf{c}''' = \text{Rand}^{T_i \rightarrow S}(\mathbf{c}'') = S^{-1} \begin{pmatrix} \eta_1 r''_1 (x''_1 + \nu_1) \\ \eta_1 r''_1 \\ \dots \\ \eta_\kappa r''_\kappa (x''_\kappa + \nu_\kappa) \\ \eta_\kappa r''_\kappa \end{pmatrix}$$

where ν_ℓ, η_ℓ would be ideally randomly chosen in \mathbf{Z}_n ensuring that $\nu_1 + \dots + \nu_\kappa = 0$. This would be sufficient to ensure that \mathbf{c}''' is a *fresh* encryption of $\text{Decrypt}(T_i, \mathbf{c}'')$.

Unfortunately, ν_ℓ, η_ℓ cannot be randomly chosen with deterministic operators. We propose to define them as evaluations of randomly and secretly chosen polynomials over \mathbf{c}'' . By doing this, the two *sources of randomness* $X(\mathbf{c}'')$ and $R(\mathbf{c}'')$ are mixed, i.e. $X(\mathbf{c}''')$ (resp. $R(\mathbf{c}''')$) depend on both $X(\mathbf{c}'')$ and $R(\mathbf{c}'')$ and not only on $X(\mathbf{c}'')$ (resp. $R(\mathbf{c}'')$). This is sufficient to make all the (listed) attacks by linearization irrelevant.

1.4 Organization of the paper

Section 2. We present some security results under the factoring assumption. In particular, we show that recovering evaluations of non-symmetric polynomials only knowing evaluations of symmetric ones is hard.

Section 3. Gröbner bases are currently assumed to be key tools to solve systems of nonlinear equations in finite fields. We propose a brief, high-level introduction to Gröbner bases and their applications.

Section 4. We detail the noise-free private-key encryption scheme sketched in Section 1.2. In particular, we exhibit a strong algebraic link between this encryption scheme and the famous cryptographic problem LWE [Reg05]. Moreover, we propose a new cryptographic problem, called *sum of fractions* problem (SOF), whose hardness implies the semantic security of our private-key encryption scheme.

Section 5. The private-key encryption is not homomorphic in the sense that the vector sum is not a homomorphic operator. In Section 5, we propose a basic way to build homomorphic operators from the secret key. Applying these operators consists of evaluating quadratic polynomials over encryptions. While the HE obtained is not secure, this construction captures fundamental ideas. This insecure construction is here presented mainly to accompany the reader with a pedagogical approach.

Section 6. In order to overcome the vulnerabilities of the previous HE, we propose to randomize the generation of the homomorphic operators. This is done by introducing operators Rand , as sketched in Section 1.2.

Section 7 (Security analysis). Considering operators Rand makes the attacks by linearization irrelevant. Hence, our encryption scheme should be investigated as a system of nonlinear equations. Computing Gröbner bases is a classical way to investigate such systems. Symmetry properties ensure that the internal randomness (i.e. the secret matrices and polynomials) underlying our construction cannot be recovered under the factoring assumption. This discards a large class of attacks. We then extensively study the security of our scheme against algebraic attacks, i.e. attacks based on Gröbner bases. All our experiments will be done on the computer algebra system SageMath (running on a single 64-bit Intel Core running at 3 Ghz). The obtained running times suggest that these attacks are not polynomial-time. The analysis of more formal criteria (presented in Section 3) to investigate the complexity of algebraic attacks will enhance this idea. Finally, we experimentally and theoretically show the fundamental role played by symmetry in our construction.

Appendix C. We propose a generalization of our construction by adding randomness. According to preliminary experiments (omitted in this paper), we are relatively confident in the possibility of removing the factoring assumption.

Remark 1. The source code of our HE and the sources of some attacks proposed in this paper can be found in the following archive:

<https://drive.google.com/drive/folders/1fkma-saCL05LA7eqgIn-D7OTpYXN6xCQ?usp=sharing>

1.5 Notation

We use standard Landau notations. Throughout this paper, we let λ denote the security parameter: all known attacks against the cryptographic scheme under scope should require $2^{\Omega(\lambda)}$ bit operations to mount. Let $\kappa \geq 2$ be an integer and let n be a large prime or a RSA modulus. All the computations considered in this paper will be done in \mathbf{Z}_n .

- Δ_κ refers to the set of permutations over $\{1, \dots, \kappa\}$.
- $\Sigma_\kappa = \{\sigma_1, \dots, \sigma_\kappa\} \subset \Delta_\kappa$ defined by $\sigma_i(j) = (i+j-2 \bmod \kappa)+1$, i.e. $\sigma_i(1) = i; \sigma_i(2) = i+1; \dots; \sigma_i(\kappa) = i-1$.
- ‘Choose at random $x \in X$ ’ will systematically mean that x is chosen according to uniform probability distribution over X , i.e. $x \stackrel{\$}{\leftarrow} X$.
- The uniform probability distribution over a set X will be denoted by $\mathcal{U}(X)$.
- A vector $\mathbf{v} = \begin{pmatrix} v_1 \\ \dots \\ v_t \end{pmatrix}$ can be also denoted by $\mathbf{v} = (v_1, \dots, v_t)$.
- The inner product of two vectors \mathbf{v} and \mathbf{v}' is denoted by $\langle \mathbf{v}, \mathbf{v}' \rangle$
- The set of all square t -by- t matrices over \mathbf{Z}_n is denoted by $\mathbf{Z}_n^{t \times t}$.
- Given a matrix $S \in \mathbf{Z}_n^{t \times t}$, \mathbf{L}_i^S (or simply \mathbf{L}_i) will refer to the linear function defined by

$$\mathbf{L}_i^S(\mathbf{u}) = \langle \mathbf{s}_i, \mathbf{u} \rangle$$

where \mathbf{s}_i denoted the i^{th} row of S .

Remark 2. The number $M(m, d)$ of m -variate monomials of degree d is equal to $\binom{d+m-1}{d}$. In particular, $M(2\kappa, \kappa) \approx (27/4)^\kappa$ is exponential in κ .

2 Some security results under the factoring assumption

Throughout this section, n denotes a randomly chosen RSA-modulus. Given a function $\phi : \mathbf{Z}_n^r \rightarrow \mathbf{Z}_n$, $z_\phi \stackrel{\text{def}}{=} \#\{x \in \mathbf{Z}_n^r | \phi(x) = 0\} / n^r$. Classically, a polynomial will be said *null* (or identically null) if each coefficient of its expanded representation is equal to 0.

2.1 Roots of polynomials

The following result proved in [AM09] establishes that it is difficult to output a polynomial ϕ such that z_ϕ is non-negligible. The security of RSA in the generic ring model can be quite straightforwardly derived from this result (see [AM09]).

Theorem 1. (Lemma 4 of [AM09] and Proposition 1 of [GT20]). *Assuming factoring is hard, there is no p.p.t. algorithm \mathcal{A} which inputs n and which outputs² a $\{+, \times\}$ -circuit representing a non-null polynomial $\phi \in \mathbf{Z}_n[X_1, \dots, X_r]$ such that z_ϕ is non-negligible.*

Thanks to this lemma, showing that a polynomial (built without knowing the factorization of n) is equal to 0 with non-negligible probability becomes an algebraic problem: it suffices to prove that it is identically null.

² with non-negligible probability (the coin toss being the choice of n and the internal randomness of \mathcal{A})

2.2 Symmetry

Let $\kappa \geq 2$ and $t \geq 1$ be integers. Recall that Δ_κ denotes the set of the permutations over $\{1, \dots, \kappa\}$. Throughout this section, we will consider an arbitrary subset $\Sigma \subseteq \Delta_\kappa$. Let y_1, y_2 be randomly chosen in \mathbf{Z}_n . It is well known that recovering y_1 with non-negligible probability given only $S = y_1 + y_2$ or $P = y_1 y_2$ is difficult assuming the hardness of factoring (y_1, y_2 are the roots of the polynomial $y^2 - Sy + P$). In this section, we propose to extend this. The following definition naturally extends the classical definition of symmetric polynomials.

Definition 1. Consider the tuples of indeterminate $(Y_\ell = (X_{\ell 1}, \dots, X_{\ell t}))_{\ell=1, \dots, \kappa}$. A polynomial $\phi \in \mathbf{Z}_n[Y_1, \dots, Y_\kappa]$ is Σ -symmetric if for any permutation $\sigma \in \Sigma$,

$$\phi(Y_1, \dots, Y_\kappa) = \phi(Y_{\sigma(1)}, \dots, Y_{\sigma(\kappa)})$$

Let \mathcal{P} be an arbitrary p.p.t. algorithm that inputs n and outputs m Σ -symmetric polynomials s_1, \dots, s_m and a non- Σ -symmetric polynomial π . Evaluating π only given evaluations of s_1, \dots, s_m is difficult.

Lemma 1. Let n be a randomly chosen RSA modulus and $(s_1, \dots, s_m, \pi) \leftarrow \mathcal{P}(n)$. Assuming the hardness of factoring, there is no p.p.t algorithm which outputs $\pi(y)$ given only $s_1(y), \dots, s_m(y)$ with non-negligible probability over the choice of $n, y \xleftarrow{\$} \mathbf{Z}_n^{\kappa t}$.

Proof. See [GT20].

□

3 Solving polynomial systems

Solving polynomial systems is required in numerous applications. Gröebner bases are key tools to solve such systems. A brief introduction to this algebraic notion is proposed in this section. Many details are here omitted. Nevertheless, we think that this overview is sufficient to understand, analyze and interpret our experiments.

3.1 Problem statement

Given a field k , $k[x_1, \dots, x_m]$ refers to the ring of polynomials in the m variables x_1, \dots, x_m with coefficients in k . Given a set of polynomials $F = \{f_1, \dots, f_t\}$, the smallest ideal of $k[x_1, \dots, x_m]$ containing the polynomials f_1, \dots, f_t is denoted by $\langle F \rangle$.

Definition 2. $\langle F \rangle$ denotes the polynomial ideal generated by f_1, \dots, f_t , i.e.

$$\langle F \rangle = \{g_1 f_1 + \dots + g_t f_t \mid g_1, \dots, g_t \in k[x_1, \dots, x_m]\}$$

$F = \{f_1, \dots, f_t\}$ is said to be a basis of $\langle F \rangle$.

The system of polynomial equations (or simply polynomial system) denoted by $F = 0$ is the set of simultaneous equations

$$\begin{cases} f_1(x_1, \dots, x_m) = 0 \\ \dots \\ f_t(x_1, \dots, x_m) = 0 \end{cases}$$

A solution of a polynomial system is a set of values for the x_i that make all equations true. For sake of simplicity, we here assume this system has a unique solution $x^* = (x_1^*, \dots, x_m^*)$.

The concept of ideals is intimately linked to polynomial systems. Indeed, one can easily check that

$$\forall g \in \langle F \rangle, \quad g(x^*) = 0$$

This straightforwardly implies that $F = 0$ and $G = 0$ have the same sets of solutions, provided $\langle F \rangle = \langle G \rangle$. **Gröbner bases** G (defined in the next section) are special bases, ensuring that the polynomial system $G = 0$ is easy to solve.

The ideal³ $\langle F \rangle \cap k[x_1]$ is an ideal of $k[x_1]$. It is well-known that rings of univariate polynomials are principal. It follows that $\langle F \rangle \cap k[x_1]$ (assumed to be not null) can be generated by a single polynomial g_1 , i.e.

$$\langle F \rangle \cap k[x_1] = \langle g_1 \rangle$$

It is ensured that $g_1(x_1^*) = 0$. Moreover, if k is a finite field, this equation can be efficiently solved by using Berlekamp's algorithm [BRS67]. According to the *elimination theorem* (presented in the next section), g_1 can be recovered by computing lex-Gröbner bases .

3.2 Gröbner Basis vs variable elimination

Univariate monomials are naturally ordered by their degree. This order is implicitly considered in the euclidean division of univariate polynomials. To extend the division algorithm to multivariate polynomials, the multivariate monomial set should also be totally ordered. Based on a variable ordering⁴, say $x_1 < x_2 < \dots < x_m$, several monomial orderings are classically considered:

- Lexicographic order (*lex*).
- Graded reverse lexicographic order (*grevlex*).
- Elimination monomial ordering (a mix of the two first ones)

A multivariate polynomial can be represented by a sum of monomials, called the *expanded representation*. Classically, $\text{LT}(f)$ refers to the leading monomial that is to say, the largest monomial in the expanded representation of f for the chosen monomial ordering.

Definition 3. A finite basis $G = \{g_1, \dots, g_t\}$ of a ideal $I \subset k[x_1, \dots, x_m]$ is a Gröbner basis if

$$\langle \text{LT}(I) \rangle = \langle \text{LT}(g_1), \dots, \text{LT}(g_t) \rangle$$

where $\text{LT}(I) = \{\text{LT}(f) | f \in I\}$.

In other words, a Gröbner basis $G = \{g_1, \dots, g_t\}$ of I is a special basis ensuring that the ideal generated by the leading terms $\text{LT}(g_1), \dots, \text{LT}(g_t)$ is equal to the ideal generated by the leading terms $(\text{LT}(f))_{f \in I}$.

Proposition 1. Any ideal $I \subset k[x_1, \dots, x_m]$ has a Gröbner basis.

An ideal I has even an unique *reduced* Gröbner basis. Informally speaking, it can be efficiently obtained from any Gröbner basis by removing redundancy (see [CLO91]). Hence, there is an efficient algorithm to decide whether two Gröbner bases G and G' are equivalent, i.e. $\langle G \rangle = \langle G' \rangle$.

Knowing a Gröbner basis of an ideal is very meaningful about this ideal. Many problems can be efficiently solved from this knowledge, e.g. *ideal membership*, *implicitization*, *variable elimination*, etc. In this paper, we mainly focus on variable elimination which is relevant to solve polynomial systems.

lex-Gröbner basis. Let I be an ideal of $k[x_1, \dots, x_m]$. The *lex* order is particularly relevant to eliminating variables from an ideal and thus to solving polynomial systems. More precisely, computing Gröbner bases dealing with the lexicographic order allows to recover the basis of $I \cap k[x_1]$. Indeed, assuming G is a *lex*-Gröbner basis of I , it is ensured that

$$G \cap k[x_1]$$

is a Gröbner basis of the ideal $I \cap k[x_1]$. This is a special case of the famous *Elimination theorem* (see [CLO91]).

³ It is well-known that the intersection of two ideals is also an ideal.

⁴ A variable is also a monomial.

Theorem 2. (Elimination theorem.) Let I be an ideal of $k[x_1, \dots, x_m]$ and G be a *lex*-Gröbner basis of I . It is ensured that

$$G_\ell = G \cap k[x_1, \dots, x_\ell]$$

is a Gröbner basis of the ideal $I \cap k[x_1, \dots, x_\ell]$

This theorem is fundamental in variable elimination theory. A system $F = 0$ can be seen as a set of constraints between variables. The elimination theorem says how to *eliminate* some variables from these constraints in order to obtain *all the* constraints between only a subset of variables: it suffices to compute the *lex*-Gröbner basis of $\langle F \rangle$.

3.3 Classical algorithms computing Gröbner bases

As far as we know, all algorithms for computing Gröbner bases input a finite basis F of a polynomial ideal and return a Gröbner basis G of this ideal. Moreover, the polynomials should be described by their expanded representation (sum of monomials). In particular, these algorithms are not efficient if the expanded representation of the input or output polynomials is exponential-size.

Buchberger’s algorithm [Buc06] was the first algorithm to compute Gröbner bases. Faugere et al. [Fau99] have significantly improved it with the algorithms F_4 or F_5 . These two algorithms are currently assumed to be the most efficient ones.

lex-Gröbner bases⁵ have the strongest algebraic structure making them convenient for variable elimination and thus for polynomial system solving. However, in many situations, it is a bit of overkill to compute a Gröbner basis using *lex* order. Computing *lex*-Gröbner bases can be very inefficient in practice. More efficient monomial orderings, defined especially for variable elimination, are considered in most computer algebra systems (Maple, SageMath, Mathematica, etc.).

Conversely, in most cases, *grevlex* order produces Gröbner bases with polynomials of the smallest total degree. This order is often considered to be the most efficient. It is the *default* order in most computer algebra systems. In our experiments, we will consider this order as a reference. More precisely, we will systematically compute *grevlex*-Gröbner bases. If we succeed in proving that such computations are hard, we will assume that it is also hard for the other orders.

Implementation. All our experiments will be done with SageMath. Most of the recent Gröbner basis algorithms are available on this free open-source mathematics software. In our experiments, we mainly use the SageMath functions `ideal_elimination(.)` and `groebner_basis(.)`.

- `groebner_basis(algorithm, deg_bound, ...)`
- `ideal_elimination(variables, algorithm, ...)`

3.4 Complexity

While the worst-case complexity of Gröbner basis computation is doubly exponential in the number of variables, this computation may be efficient in many applications. To prove that our scheme is secure against attacks based on such computations, we need to show that these attacks are exponential-time *w.r.t.* the security parameter λ . To achieve this, four criteria will be considered in our experiments: the three last ones dealing with the *grevlex* order.

- **Running time.** It is the most obvious criteria. The running times will be measured for several toy parameters. However, they quickly become prohibitive. In these conditions, it is difficult to measure complexity, which captures asymptotic behaviors. This is the main obstacle to this approach.
- **Size of the Gröbner basis.** If the memory size of the Gröbner basis exponentially grows with the parameters, it is ensured that its computation is also exponential-time. By carefully examining the obtained Gröbner basis, this criteria will be relevant in some cases. This criteria is surely the most significant when it can be applied.

⁵ Gröbner basis *w.r.t.* the lexicographical order (*lex*).

- Degree of semi-regularity (\mathbf{d}_{reg}). Informally speaking, \mathbf{d}_{reg} can be seen as the lowest degree at which the system starts behaving irregularly, i.e. it does not behave like a random one. The algorithms F_4 and F_5 run in

$$O\left(\binom{m + \mathbf{d}_{reg}}{m}^\omega\right)$$

arithmetic operations (with $\omega \approx 2.39$), and ω is the exponent in the complexity of efficient matrix multiplication. The function `degree_of_semi_regularity()` in SageMath efficiently computes \mathbf{d}_{reg} .

- Degree max (\mathbf{d}_{max}). It is the highest degree reached during the computation of the Gröebner basis. To measure it, we use the parameter `deg_bound` in the function `groebner_basis()`. If `deg_bound` $<$ \mathbf{d}_{max} then `groebner_basis()` does surely not return a Gröebner basis. Hence, \mathbf{d}_{max} can be defined as the smallest value of `deg_bound` such that `groebner_basis(., deg_bound)` returns a Grobner basis.

Remark 3. Only upper bounds can be derived from the criteria \mathbf{d}_{reg} and \mathbf{d}_{max} . Hence, they cannot be used to prove exponential-time complexities. Nevertheless, if they are in $O(1)$, complexity is surely polynomial-time.

Remark 4. \mathbf{d}_{reg} is efficiently measured by SageMath while \mathbf{d}_{max} requires to compute Gröebner bases.

4 A private-key encryption scheme

4.1 Definition

We design a private-key encryption scheme where the secret key K contains 2κ randomly chosen secret vectors $\mathbf{s}_1, \dots, \mathbf{s}_{2\kappa}$ belonging to $\mathbf{Z}_n^{2\kappa}$. Encrypting $x \in \mathbf{Z}_n$ simply consists of randomly choosing $\mathbf{c} \in \mathbf{Z}_n^{2\kappa}$ satisfying

$$\frac{\langle \mathbf{s}_1, \mathbf{c} \rangle}{\langle \mathbf{s}_2, \mathbf{c} \rangle} + \dots + \frac{\langle \mathbf{s}_{2\kappa-1}, \mathbf{c} \rangle}{\langle \mathbf{s}_{2\kappa}, \mathbf{c} \rangle} = x \quad (2)$$

By assuming the vectors $\mathbf{s}_1, \dots, \mathbf{s}_{2\kappa}$ are linearly independent, our scheme can be defined as follows:

Definition 4. Let λ be a security parameter. The functions *KeyGen*, *Encrypt*, *Decrypt* are defined as follows:

- *KeyGen*(λ). Let κ be a positive integer indexed by λ and let n be a randomly chosen RSA-modulus. Choose at random an invertible matrix $S \in \mathbf{Z}_n^{2\kappa \times 2\kappa}$. Output

$$K = \{S\} ; pp = \{n, \kappa\}$$

- *Encrypt*($K, pp, x \in \mathbf{Z}_n$). Choose at random r_1, \dots, r_κ in \mathbf{Z}_n^* and x_1, \dots, x_κ in \mathbf{Z}_n s.t. $x_1 + \dots + x_\kappa = x$. Output

$$\mathbf{c} = S^{-1} \begin{pmatrix} r_1 x_1 \\ r_1 \\ \dots \\ r_\kappa x_\kappa \\ r_\kappa \end{pmatrix}$$

- *Decrypt*($K, pp, \mathbf{c} \in \mathbf{Z}_n^{2\kappa}$). Output

$$x = \sum_{\ell=1}^{\kappa} \frac{\langle \mathbf{s}_{2\ell-1}, \mathbf{c} \rangle}{\langle \mathbf{s}_{2\ell}, \mathbf{c} \rangle}$$

where \mathbf{s}_i refers to the i^{th} row of S .

Throughout this section, the linear function \mathbf{L}_i^S will be simply denoted by \mathbf{L}_i , i.e.

$$\mathbf{L}_i(\mathbf{u}) = \langle \mathbf{s}_i, \mathbf{u} \rangle$$

Moreover, $pp = \{n, \kappa\}$ will be assumed to be public. The homomorphic operators, developed in the next section, will be included in pp . For sake of simplicity, this parameter will often be omitted.

Proving correctness is straightforward by using the relation $x = r_1 x_1 / r_1 + \dots + r_\kappa x_\kappa / r_\kappa$. The function `Decrypt` can be represented by the ratio of the two degree- κ polynomials $\phi_S, \phi'_S \in \mathbf{Z}_n[X_1, \dots, X_{2\kappa}]$ defined by

$$\phi_S = \sum_{\ell=1}^{\kappa} L_{2\ell-1} \prod_{\ell' \neq \ell} L_{2\ell'} ; \phi'_S = \prod_{\ell=1}^{\kappa} L_{2\ell} \quad (3)$$

with

$$\text{Decrypt}(K, \mathbf{c}) = \phi_S(\mathbf{c}) / \phi'_S(\mathbf{c})$$

At this step, our scheme is not homomorphic in the sense that the vector sum is not a homomorphic operator. Indeed, \mathbf{c} and $a \cdot \mathbf{c}$ are encryptions of the same message for any $a \in \mathbf{Z}_n^*$.

We can identify two independent sources of randomness in `Encrypt`: the choice of the *shares* x_i and the choice of the *masks* r_i . As mentioned in the introduction, we will consider the following convenient notation for capturing these two types of randomness:

$$\begin{aligned} X(\mathbf{c}) &\stackrel{\text{def}}{=} (x_1, \dots, x_\kappa) = (L_1(\mathbf{c})/L_2(\mathbf{c}), \dots, L_{2\kappa-1}(\mathbf{c})/L_{2\kappa}(\mathbf{c})) \\ R(\mathbf{c}) &\stackrel{\text{def}}{=} (r_1, \dots, r_\kappa) = (L_2(\mathbf{c}), \dots, L_{2\kappa}(\mathbf{c})) \end{aligned}$$

4.2 The factoring assumption

The factorization of n is not used in `KeyGen`. Consequently, the generation of n could be externalized⁶ (for instance, generated by an oracle). In other words, n could be a public input of `KeyGen`. It follows that all the polynomials considered in our security analysis are built without using the factorization of n . Hence, Theorem 1 and Lemma 1 can be invoked.

4.3 The basic attack

The most natural attack consists of solving a linear system in order to recover ϕ_S . Let $\mathbf{c} \leftarrow \text{Encrypt}(K, x)$ be an encryption of an arbitrary $x \in \mathbf{Z}_n$. By definition, the polynomials ϕ_S and ϕ'_S (see (3)) satisfy

$$\phi_S(\mathbf{c}) - x\phi'_S(\mathbf{c}) = 0$$

By sampling \mathbf{c} , we get a system of equations. The expanded representation of ϕ_S and ϕ'_S could be thus recovered by solving a size- $2 \times M(2\kappa, \kappa)$ linear system whose variables are their monomial coefficients. However, this attack fails provided $\kappa = \Theta(\lambda)$ because $M(2\kappa, \kappa)$ is exponential in λ in this case (see Remark 2). For instance, by choosing $\kappa = 13$, the attack consists of solving a linear system dealing with approximately 10^{10} variables: this is currently assumed to be infeasible.

4.4 Algebraic link with LWE

The Learning With Errors (LWE) Problem was introduced by Regev [Reg09]. It is a generalisation for large primes of the well-known LPN (Learning Parity with Noise) problem. Since its introduction, LWE has become a source of many innovative cryptosystems, such as homomorphic encryption. Reasons of LWE's success in cryptography include its simplicity as well as convincing theoretical arguments regarding its hardness, a reduction from (worst-case) assumed hard lattice problems to (average-case) LWE.

LWE problem. To fit to our needs, we propose a description of LWE problem that slightly differs (but equivalent) from the usual one. For integers $\kappa, n \geq 2$, a vector $\mathbf{s} \in \{1\} \times \mathbf{Z}_n^{\kappa-1}$ and a probability distribution χ on \mathbf{Z}_n , let $A_{\mathbf{s}, \chi}$ be the distribution such that $\mathbf{u} \leftarrow A_{\mathbf{s}, \chi}$ is a randomly chosen vector of \mathbf{Z}_n^κ satisfying $\langle \mathbf{s}, \mathbf{u} \rangle = e$ where $e \leftarrow \chi$ is a noise term randomly chosen.

⁶ ensuring that its factorization was forgotten just after its generation

Definition 5. For an integer $n = n(\kappa)$, a distribution ψ over $\{1\} \times \mathbf{Z}_n^{\kappa-1}$ and an error distribution $\chi = \chi(\kappa)$ over \mathbf{Z}_n , the learning with errors problem $\text{LWE}_{\kappa,n,m,\chi,\psi}$ is defined as follows: given m independent samples from $A_{\mathbf{s},\chi}$ where $\mathbf{s} \leftarrow \psi$, output \mathbf{s} with non-negligible probability.

The decision variant of the LWE problem, denoted by $\text{DLWE}_{\kappa,n,m,\chi,\psi}$ is to distinguish (with non-negligible advantage) m samples chosen according to $A_{\mathbf{s},\chi}$ from m samples chosen according to the uniform distribution over \mathbf{Z}_n^κ .

The probability distribution χ is typically taken to be the discrete Gaussian distribution on \mathbf{Z}_n with mean 0 and standard deviation σ . To simplify our purpose, we will here assume that χ is uniform over $\{-\sigma, \dots, \sigma\}$.

Arora et Ge proposed to attack LWE with a linearization approach [AG11b]. This approach considers the degree- $(2\sigma + 1)$ polynomial p defined by $p(x) = x \prod_{i=1}^{\sigma} (x - i)(x + i)$. By construction, it is ensured that $p(e) = 0$ for any $e \in \{-\sigma, \dots, \sigma\}$. Hence, for any example $\mathbf{u} \leftarrow A_{\mathbf{s},\chi}$, it is ensured that

$$\phi_{\text{LWE}}(\mathbf{u}) \stackrel{\text{def}}{=} p(\langle \mathbf{s}, \mathbf{u} \rangle) = 0$$

By definition, ϕ_{LWE} is a degree- $(2\sigma + 1)$ polynomial whose coefficients are evaluations of polynomials over \mathbf{s} . Its expanded representation size is in $\Theta(\kappa^{2\sigma})$.

By considering sufficiently many samples \mathbf{u} , the expanded representation of ϕ_{LWE} can be recovered by solving a linear system. This is mainly the attack of DLWE proposed by Arora et Ge. Hence, $\kappa^{2\sigma}$ should be exponential in the security parameter.

The analogy with our private-key encryption is now straightforward. Indeed, $\text{Encrypt}(S, 0)$ outputs a randomly chosen vector \mathbf{u} satisfying $\phi(\mathbf{u}) = 0$ with $\phi = \phi_S$. Similarly, an example $\mathbf{u} \leftarrow A_{\mathbf{s},\chi}$ of LWE is a randomly chosen vector \mathbf{u} satisfying $\phi(\mathbf{u}) = 0$ with $\phi = \phi_{\text{LWE}}$.

It follows that breaking the semantic security of our scheme or solving DLWE consist of distinguishing between $\mathcal{U}(\{\mathbf{u} \in \mathbf{Z}_n^\kappa \mid \phi(\mathbf{u}) = 0\})$ and $\mathcal{U}(\mathbf{Z}_n^\kappa)$: ϕ being a polynomial whose each monomial coefficient is a polynomial defined over a secret set K of elements belonging to \mathbf{Z}_n , $K = \{S\}$ in our scheme and $K = \{\mathbf{s}\}$ in LWE.

This simple analysis shows an unexpected algebraic link between our private-key encryption and LWE. In the next section, we propose a new cryptographic assumption whose semantic security of our scheme is a special case.

4.5 Generalization: sum of fractions problem

Throughout this section, we mainly reuse the notation of Section 4.1. We here propose a new cryptographic problem whose semantic security of our scheme is a special case. Let A_S be the uniform distribution⁷ over $\{\mathbf{u} \in \mathbf{Z}_n^{2\kappa} \mid \phi_S(\mathbf{u}) = 0\}$.

Definition 6. For an integer $n = n(\kappa)$, a distribution ψ over $\mathbf{Z}_n^{2\kappa \times 2\kappa}$, the sum of fractions problem $\text{SOF}_{\kappa,n,m,\psi}$ is defined as follows: given m independent samples from A_S where $S \leftarrow \psi$, output S with non-negligible probability.

The decision variant of the SOF problem, denoted by $\text{DSOF}_{\kappa,n,m,\psi}$ is to distinguish (with non-negligible advantage) m samples chosen according to A_S from m samples chosen according to the uniform distribution over $\mathbf{Z}_n^{2\kappa}$.

The semantic security of our private-key encryption is equivalent to $\text{DSOF}_{\kappa,n,m,\psi}$ where n is a RSA modulus and ψ is uniform over the set of invertible matrices $S \in \mathbf{Z}_n^{2\kappa \times 2\kappa}$.

As seen in the previous section, informally speaking, DSOF is algebraically equivalent to DLWE. Moreover, as our scheme is noise-free, it should be not subject to lattice-based attacks. This gives us confident in the hardness of DSOF.

In section 5.5, it will be proven that solving $\text{SOF}_{\kappa,n,m,\psi}$ is at least as hard as factoring n , for any $\kappa \geq 2$. In Section 7.4, algebraic attacks on DSOF (Attack 1 and its variants) are proposed. These attacks appear to be totally inefficient, provided $\kappa = \Theta(\lambda)$.

⁷ The vector $\mathbf{u} \leftarrow \text{Encrypt}(S, 0)$ is drawn according to A_S .

5 A basic/naive implementation of the homomorphic operators

Let $S \leftarrow \text{KeyGen}(\lambda)$. In this section, we will consider the quadratic polynomials $L_{ij} \in \mathbf{Z}_n[U_1, \dots, U_{2\kappa}, V_1, \dots, V_{2\kappa}]$ defined by $L_{ij}(\mathbf{u}, \mathbf{v}) = L_i(\mathbf{u})L_j(\mathbf{v}) = \langle \mathbf{s}_i, \mathbf{u} \rangle \langle \mathbf{s}_j, \mathbf{v} \rangle$.

In this section, we propose a natural and simple way to implement the homomorphic operators. In the following of this paper, we will consider the two encryptions \mathbf{c} and \mathbf{c}' of respectively x and x'

$$\mathbf{c} = S^{-1} \begin{pmatrix} r_1 x_1 \\ r_1 \\ \dots \\ r_\kappa x_\kappa \\ r_\kappa \end{pmatrix} ; \quad \mathbf{c}' = S^{-1} \begin{pmatrix} r'_1 x'_1 \\ r'_1 \\ \dots \\ r'_\kappa x'_\kappa \\ r'_\kappa \end{pmatrix}$$

The homomorphic operators **Add** (also denoted by \oplus) and **Mult** (also denoted by \otimes) are output by the function **OGen**. For concreteness, **OGen**(S, i) outputs a nonlinear operator \mathbf{O}_i satisfying

$$\begin{aligned} \mathbf{c} \oplus \mathbf{c}' &= \mathbf{O}_0(\mathbf{c}, \mathbf{c}') \\ \mathbf{c} \otimes \mathbf{c}' &= \mathbf{O}_1(\mathbf{c}, \mathbf{c}') \oplus \dots \oplus \mathbf{O}_\kappa(\mathbf{c}, \mathbf{c}') \end{aligned}$$

5.1 The additive operator **Add** (also denoted by \mathbf{O}_0)

The additive operator exploits the basic equality

$$\frac{a}{b} + \frac{a'}{b'} = \frac{ab' + a'b}{bb'}$$

We notice that the numerator and the denominator of this sum can be obtained by evaluating quadratic polynomials over a, a', b, b' . This is the starting point to build an additive operator **Add** satisfying (see Figure 1)

$$\begin{aligned} X(\mathbf{c} \oplus \mathbf{c}') &= (x_1 + x'_1, \dots, x_\kappa + x'_\kappa) \\ R(\mathbf{c} \oplus \mathbf{c}') &= (r_1 r'_1, \dots, r_\kappa r'_\kappa) \end{aligned}$$

Definition 7. $\mathbf{OGen}(S, 0)$ outputs the expanded representation of the quadratic polynomials $p_1, \dots, p_{2\kappa}$ defined by

$$\begin{pmatrix} p_1 \\ \dots \\ p_{2\kappa} \end{pmatrix} = S^{-1} \begin{pmatrix} L_{12} + L_{21} \\ L_{22} \\ \dots \\ L_{2\kappa-1, 2\kappa} + L_{2\kappa, 2\kappa-1} \\ L_{2\kappa, 2\kappa} \end{pmatrix}$$

and $\text{Add}(\mathbf{c}, \mathbf{c}') = (p_1(\mathbf{c}, \mathbf{c}'), \dots, p_{2\kappa}(\mathbf{c}, \mathbf{c}'))$

A toy implementation of **Add** is given in Appendix A.

Proposition 2. $\text{Add} \leftarrow \mathbf{OGen}(S, 0)$ is a valid additive homomorphic operator.

Proof. By construction, for any $\ell = 1, \dots, \kappa$

$$\begin{aligned} \langle \mathbf{s}_{2\ell-1}, (p_1(\mathbf{c}, \mathbf{c}'), \dots, p_{2\kappa}(\mathbf{c}, \mathbf{c}')) \rangle &= L_{2\ell-1, 2\ell}(\mathbf{c}, \mathbf{c}') + L_{2\ell, 2\ell-1}(\mathbf{c}, \mathbf{c}') = r_\ell x_\ell r'_\ell + r'_\ell x'_\ell r_\ell \\ &= r_\ell r'_\ell (x_\ell + x'_\ell) \end{aligned}$$

and

$$\langle \mathbf{s}_{2\ell}, (p_1(\mathbf{c}, \mathbf{c}'), \dots, p_{2\kappa}(\mathbf{c}, \mathbf{c}')) \rangle = L_{2\ell, 2\ell}(\mathbf{c}, \mathbf{c}') = r_\ell r'_\ell$$

It follows that

$$\text{Decrypt}(S, \text{Add}(\mathbf{c}, \mathbf{c}')) = \frac{r_1 r'_1 (x_1 + x'_1)}{r_1 r'_1} + \dots + \frac{r_\kappa r'_\kappa (x_\kappa + x'_\kappa)}{r_\kappa r'_\kappa} = x_1 + x'_1 + \dots + x_\kappa + x'_\kappa = x + x'$$

□

Example.

- $n = 5, \kappa = 1$
- $S = \begin{bmatrix} 3 & 1 \\ 2 & 1 \end{bmatrix}; S^{-1} = \begin{bmatrix} 1 & 4 \\ 3 & 3 \end{bmatrix}$
- $\text{Add}(\mathbf{u}, \mathbf{v}) = S^{-1} \begin{pmatrix} (u_1 + 4u_2)(3v_1 + 3v_2) + (3u_1 + 3u_2)(v_1 + 4v_2) \\ (3u_1 + 3u_2)(3v_1 + 3v_2) \end{pmatrix} = \begin{pmatrix} 3u_1v_1 + 3(u_2v_1 + u_1v_2) + u_2v_2 \\ 3u_1v_1 + (u_2v_1 + u_1v_2) + 4u_2v_2 \end{pmatrix}$
- $\mathbf{c}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}; \mathbf{c}_2 = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$
- $\text{Decrypt}(S, \mathbf{c}_1) = \frac{3 \times 1 + 1 \times 1}{2 \times 1 + 1 \times 1} = 3; \text{Decrypt}(S, \mathbf{c}_2) = \frac{3 \times 2 + 1 \times 3}{2 \times 2 + 1 \times 3} = 2$
- $\mathbf{c}_3 = \text{Add}(\mathbf{c}_1, \mathbf{c}_2) = \begin{pmatrix} p_1(\mathbf{c}_1, \mathbf{c}_2) \\ p_2(\mathbf{c}_1, \mathbf{c}_2) \end{pmatrix} = \begin{pmatrix} 3 \times 1 \times 2 + 3(1 \times 2 + 1 \times 3) + 1 \times 3 \\ 3 \times 1 \times 2 + (1 \times 2 + 1 \times 3) + 4 \times 1 \times 3 \end{pmatrix} = \begin{pmatrix} 4 \\ 3 \end{pmatrix} = \text{Add}(\mathbf{c}_2, \mathbf{c}_1)$
- $\text{Decrypt}(S, \mathbf{c}_3) = \frac{3 \times 4 + 1 \times 3}{2 \times 4 + 1 \times 3} = 0 = \text{Decrypt}(S, \mathbf{c}_1) + \text{Decrypt}(S, \mathbf{c}_2)$

5.2 The multiplicative operator Mult

The idea behind the operator \otimes exploits the equality

$$xx' = \sum_{1 \leq i, j \leq \kappa} x_i x'_j = \sum_{i=1}^{\kappa} \pi_i$$

with $\pi_i = \sum_{j=1}^{\kappa} x_j x'_{\sigma_i(j)}$ (recall that σ_i is the permutation over $\{1, \dots, \kappa\}$ defined by $\sigma_i(1) = i, \sigma_i(2) = i + 1, \dots, \sigma_i(\kappa) = i - 1$).

To build an encryption of xx' , we first build encryptions of $\pi_1, \dots, \pi_{\kappa}$. It then suffices to homomorphically add these encryptions to obtain an encryption of xx' .

Definition 8. Given $i \in \{1, \dots, \kappa\}$, $\mathcal{O}\text{Gen}(S, i)$ outputs the expanded representation of the quadratic polynomials $p_1, \dots, p_{2\kappa}$ defined by

$$\begin{pmatrix} p_1 \\ \dots \\ p_{2\kappa} \end{pmatrix} = S^{-1} \begin{pmatrix} \mathbf{L}_{1, 2\sigma_i(1)-1} \\ \mathbf{L}_{2, 2\sigma_i(1)} \\ \dots \\ \mathbf{L}_{2\kappa-1, 2\sigma_i(\kappa)-1} \\ \mathbf{L}_{2\kappa, 2\sigma_i(\kappa)} \end{pmatrix}$$

and $\mathcal{O}_i(\mathbf{c}, \mathbf{c}') = (p_1(\mathbf{c}, \mathbf{c}'), \dots, p_{2\kappa}(\mathbf{c}, \mathbf{c}'))$

As highlighted in Fig. 2, $\mathcal{O}_i(\mathbf{c}, \mathbf{c}')$ is a valid operator which outputs an encryption of π_i . More precisely,

$$\begin{aligned} X(\mathcal{O}_i(\mathbf{c}, \mathbf{c}')) &= (x_1 x'_{\sigma_i(1)}, \dots, x_{\kappa} x'_{\sigma_i(\kappa)}) \\ R(\mathcal{O}_i(\mathbf{c}, \mathbf{c}')) &= (r_1 r'_{\sigma_i(1)}, \dots, r_{\kappa} r'_{\sigma_i(\kappa)}) \end{aligned}$$

The multiplicative operator can be then defined by

$$\mathbf{c} \otimes \mathbf{c}' \stackrel{\text{def}}{=} \mathcal{O}_1(\mathbf{c}, \mathbf{c}') \oplus \dots \oplus \mathcal{O}_{\kappa}(\mathbf{c}, \mathbf{c}')$$

Proposition 3. The operator \otimes is a valid multiplicative homomorphic operator.

Proof. It suffices to show that $\mathbf{c}'' \leftarrow \mathcal{O}_i(\mathbf{c}, \mathbf{c}')$ is an encryption of $\pi_i = \sum_{j=1}^{\kappa} x_i x'_{\sigma_i(j)}$.

$$\text{Decrypt}(S, \mathbf{c}'') = \sum_{\ell=1}^{\kappa} \frac{r_{\ell} r'_{\sigma_i(\ell)} x_{\ell} x'_{\sigma_i(\ell)'}}{r_{\ell} r'_{\sigma_i(\ell)}} = x_1 x'_{\sigma_i(1)} + \dots + x_{\kappa} x'_{\sigma_i(\kappa)} = \pi_i$$

□

Case $\kappa = 2$. Given two encryptions \mathbf{c}, \mathbf{c}' of x, x' , we have

$$\mathbf{O}_1(\mathbf{c}, \mathbf{c}') = S^{-1} \begin{pmatrix} r_1 r_1' x_1 x_1' \\ r_1 r_1' \\ r_2 r_2' x_2 x_2' \\ r_2 r_2' \end{pmatrix}; \quad \mathbf{O}_2(\mathbf{c}, \mathbf{c}') = S^{-1} \begin{pmatrix} r_1 r_2' x_1 x_2' \\ r_1 r_2' \\ r_2 r_1' x_2 x_1' \\ r_2 r_1' \end{pmatrix}$$

implying that $\mathbf{c}'' = \text{Mult}(\mathbf{c}, \mathbf{c}') \stackrel{\text{def}}{=} \text{Add}(\mathbf{O}_1(\mathbf{c}, \mathbf{c}'), \mathbf{O}_2(\mathbf{c}, \mathbf{c}'))$ is a valid encryption of xx' . Indeed,

$$\mathbf{c}'' = S^{-1} \begin{pmatrix} r_1^2 r_1' r_2' (x_1 x_1' + x_1 x_2') \\ r_1^2 r_1' r_2' \\ r_2^2 r_1' r_2' (x_2 x_1' + x_2 x_2') \\ r_2^2 r_1' r_2' \end{pmatrix}$$

and $\text{Decrypt}(S, \mathbf{c}'') = x_1 x_1' + x_1 x_2' + x_2 x_1' + x_2 x_2' = (x_1' + x_2')(x_1 + x_2) = xx'$.

5.3 Efficiency

Each operator \mathbf{O}_i consists of evaluating 2κ quadratic variate- 2κ polynomials p_i leading to a running time in $O(\kappa^3)$. It follows that **Add** runs in $O(\kappa^3)$ and **Mult** runs in $O(\kappa^4)$. The running time of $\text{OGen}(S, i)$ is $O(\kappa^4)$ (2κ sums of 2κ quadratic polynomials). It follows that the generation of the homomorphic operators can be done in $O(\kappa^5)$.

5.4 Towards a public-key encryption

The classic way (see [Rot11]) to transform a private-key cryptosystem into a public-key cryptosystem consists of publicizing encryptions $\mathbf{c}_1, \dots, \mathbf{c}_m$ of known values x_1, \dots, x_m and using the homomorphic operators to encrypt x . Let **Encrypt1** denote this new encryption function. Assuming the IND-CPA security of the private-key cryptosystem, it suffices to prove that the CPA attacker cannot distinguish between $\text{Encrypt1}(pk, x)$ and $\text{Encrypt}(K, x)$.

5.5 Security analysis

In order to make the basic attack fail,

$$\kappa = \Theta(\lambda)$$

To simplify our security analysis, we propose to replace S^{-1} by $\det S \times S^{-1}$ in the definitions of the encrypting function and the homomorphic operators (see definitions 4, 7 and 8). This minor modification does not affect the correctness of our construction. Indeed, as noticed previously, the message is not modified by multiplying an encryption by a non-null scalar, here $\det S$. This is done to ensure that each *value* known by the CPA attacker can be written as a polynomial defined over the coefficients of S (instead of a rational function).

CPA attacker's knowledge. There are classically two sources of randomness *behind* the knowledge of the CPA attacker. The first source of randomness is the internal randomness of **KeyGen**, i.e. the choice of $K = \{S\}$. The second source of randomness comes from the encryption oracle. After receiving the challenge encryption $\mathbf{c}_0 \leftarrow \text{Encrypt}(K, x_0 \leftarrow \{0, 1\})$, the CPA attacker requests the encryption oracle to get encryptions $\mathbf{c}_1, \dots, \mathbf{c}_m$ of arbitrarily chosen plaintexts $x_1, \dots, x_m \in \mathbf{Z}_n$. To simplify our purpose, we will here assume that x_1, \dots, x_m are randomly chosen. The assumption could be removed in our security analysis by slightly extending Lemma 1.

Definition 9. Let $S \leftarrow \text{KeyGen}(\lambda)$, let $(x_{i1}, r_{i1}, \dots, x_{i\kappa}, r_{i\kappa})$ be the values (randomly) chosen by the encryption oracle to produce⁸ \mathbf{c}_i . For any $\ell \in \{1, \dots, \kappa\}$, the random vector θ_ℓ is defined by

$$\theta_\ell = (\mathbf{s}_{2\ell-1}, \mathbf{s}_{2\ell}, (x_{i\ell}, r_{i\ell})_{i=0, \dots, m})$$

The random vector $(\theta_1, \dots, \theta_\kappa)$ is denoted by $\boldsymbol{\theta}$.

⁸ $\mathbf{c}_i = \det S \times S^{-1}(r_{i1}x_{i1}, r_{i1}, \dots, r_{i\kappa}x_{i\kappa}, r_{i\kappa})$.

The knowledge of the CPA attacker can be represented as a vector $\alpha \in \mathbf{Z}_n^{m+O(\kappa^4)}$.

Definition 10. *The CPA attacker's knowledge $(\mathbf{c}_0, \dots, \mathbf{c}_m, x_1, \dots, x_m, \text{Add}, \text{Mult})$ can be represented by a vector α , the i^{th} component of α being the evaluation of a polynomial α_i over θ , i.e. $\alpha = (\alpha_1(\theta), \alpha_2(\theta), \dots) \stackrel{\text{def}}{=} \alpha(\theta)$.*

The polynomials α_i have intrinsic symmetry properties.

Lemma 2. *Each polynomial α_i is Σ_κ -symmetric (see Definition 1).*

Proof. See [GT20]

□

This result means that $\alpha_i(\theta_1, \dots, \theta_\kappa) = \alpha_i(\theta_{\sigma(1)}, \dots, \theta_{\sigma(\kappa)})$ for any $\sigma \in \Sigma_\kappa$. For instance, the operators $\mathbf{O}_0, \dots, \mathbf{O}_\kappa$ remain unchanged by permuting $\theta_1, \dots, \theta_\kappa$ according to σ_j , i.e. replacing $(\theta_1, \dots, \theta_\kappa)$ by $(\theta_j, \dots, \theta_\kappa, \theta_1, \dots, \theta_{j-1})$.

A fundamental result. By mixing Lemma 1 and Lemma 2, we get the following fundamental result.

Proposition 4. *Assuming the hardness of factoring, $\pi(\theta)$ cannot be evaluated, provided π is a polynomial that is not Σ_κ -symmetric. In particular, the CPA attacker cannot recover any:*

1. coefficient of S ,
2. product of strictly less than κ coefficients of S ,
3. polynomial⁹ $\mathbf{L}_{i_1} \times \dots \times \mathbf{L}_{i_t}$ provided $t < \kappa$

Proof. A direct consequence of lemmas 1 and 2.

□

The knowledge of the polynomial

$$\phi_S = \sum_{\ell=1}^{\kappa} \mathbf{L}_{2\ell-1} \prod_{\ell' \neq \ell} \mathbf{L}_{2\ell'}$$

would allow to distinguish between encryptions of 0 and encryptions of 1. Clearly, each monomial coefficient of ϕ_S is Σ_κ -symmetric (and thus could perhaps be recovered). However, the expanded representation of ϕ_S (or its multiples) is exponential-size provided $\kappa = \Theta(\lambda)$ and thus cannot be recovered.

By construction, ϕ_S (or its multiples) could nevertheless be efficiently represented with the linear functions \mathbf{L}_i (or $O(1)$ -products of these linear functions). However, these compact semi-factored representations do not deal with symmetric polynomials, and they cannot be recovered according to Proposition 4.

Vulnerabilities. Unfortunately, our scheme suffers from some vulnerabilities detailed in Appendix B. These attacks, called *attacks by linearization*, do not deal with nonlinear equations but only systems of linear equations. They exploit the fact that the function OGen is deterministic. As a countermeasure, we propose to *randomize* the generation of the homomorphic operators. This is the object of the next section.

⁹ and thus cannot be evaluated

6 Randomizing OGen()

In this section, we propose a way to randomize the function OGen() while keeping true the symmetry properties encapsulated in Lemma 1. As a consequence, this will ensure that Proposition 4 remains true. Consider an encryption \mathbf{c} of x and an invertible matrix $T \in \mathbf{Z}_n^{2\kappa \times 2\kappa}$. By construction of our private-key encryption, $\mathbf{c}' = T^{-1}S\mathbf{c}$ is also an encryption of x under the key T . In the following of this paper, $\mathbf{O}_i^{S \rightarrow T}$ will refer to the operator defined by

$$\mathbf{O}_i^{S \rightarrow T}(\mathbf{c}, \mathbf{c}') = T^{-1}S\mathbf{O}_i(\mathbf{c}, \mathbf{c}')$$

where $\mathbf{O}_i \leftarrow \text{OGen}(S, i)$ (recall that $\mathbf{O}_0 = \text{Add}$). Hence, by construction, we have

$$\text{Decrypt}(T, \mathbf{O}_i^{S \rightarrow T}(\mathbf{c}, \mathbf{c}')) = \text{Decrypt}(S, \mathbf{O}_i(\mathbf{c}, \mathbf{c}'))$$

It follows that $\mathbf{c}'' = \mathbf{O}_0^{S \rightarrow T}(\mathbf{c}, \mathbf{c}')$ is a valid encryption of $x + x'$ under T and $\mathbf{c}'' = \mathbf{O}_i^{S \rightarrow T}(\mathbf{c}, \mathbf{c}')$ is a valid encryption of π_i under the key T . In the next sections, we propose operators building *randomized* encryptions \mathbf{c}''' satisfying

$$\text{Decrypt}(S, \mathbf{c}''') = \text{Decrypt}(T, \mathbf{c}'')$$

6.1 Operator Rand

We here develop operators Rand to switch keys and to *randomize* encryptions (without modifying the plaintexts). The function RandGen() outputs such operators.

Definition 11. Let R, T be two invertible matrices of $\mathbf{Z}^{2\kappa \times 2\kappa}$. RandGen(R, T) randomly chooses variate- 2κ degree-1 polynomials $\eta_1, \dots, \eta_{2\kappa}$ and $\nu_0, \nu_1, \dots, \nu_{2\kappa}$ ensuring that $\nu_1 + \dots + \nu_{2\kappa} = 0$. It then outputs the expanded representations of the polynomials $p_1, \dots, p_{2\kappa}$ defined as follows¹⁰.

$$\begin{pmatrix} p_1 \\ \dots \\ p_{2\kappa} \end{pmatrix} = T^{-1} \begin{pmatrix} \eta_1 (\nu_0 \mathbf{L}_1^R + \nu_1 \mathbf{L}_2^R) \\ \eta_1 \nu_0 \mathbf{L}_2^R \\ \dots \\ \eta_{\kappa} (\nu_0 \mathbf{L}_{2\kappa-1}^R + \nu_{\kappa} \mathbf{L}_{2\kappa}^R) \\ \eta_{\kappa} \nu_0 \mathbf{L}_{2\kappa}^R \end{pmatrix}$$

The operator Rand \leftarrow RandGen(R, T) (sometimes denoted by Rand $^{R \rightarrow T}$) consists of evaluating the polynomials $(p_1, \dots, p_{2\kappa})$, i.e. Rand(\mathbf{c}) = $(p_1(\mathbf{c}), \dots, p_{2\kappa}(\mathbf{c}))$.

We first notice that Rand(\mathbf{c}) may be an invalid encryption, i.e. Decrypt(T , Rand(\mathbf{c})) may be not defined. Indeed, nothing ensures that $\eta_1 \nu_0 \mathbf{L}_{2\ell}^R \neq 0$. Nevertheless, this happens with overwhelming probability provided \mathbf{c} is built by applying homomorphic operators over encryptions output by the encryption oracle.

Proposition 5. Let Rand \leftarrow RandGen(R, T). Assume \mathbf{c} is built by applying the homomorphic operators on the encryptions $\mathbf{c}_1, \dots, \mathbf{c}_m$ output by the encryption oracle. With overwhelming probability, it is ensured that

$$\text{Decrypt}(R, \mathbf{c}) = \text{Decrypt}(T, \text{Rand}(\mathbf{c}))$$

Proof. By definition of Decrypt,

$$\text{Decrypt}(T, \text{Rand}(\mathbf{c})) = \sum_{\ell=1}^{\kappa} \frac{\eta_{\ell}(\mathbf{c})(\nu_0(\mathbf{c})\mathbf{L}_{2\ell-1}^R(\mathbf{c}) + \nu_{\ell}(\mathbf{c})\mathbf{L}_{2\ell}^R(\mathbf{c}))}{\eta_{\ell}(\mathbf{c})\nu_0(\mathbf{c})\mathbf{L}_{2\ell}^R(\mathbf{c})}$$

The denominators $\eta_{\ell}(\mathbf{c})\nu_0(\mathbf{c})\mathbf{L}_{2\ell}^R(\mathbf{c})$ can be expressed as a non-null polynomial defined over $\theta \stackrel{\$}{\equiv} \mathbf{Z}_n^t$. According to Theorem 1, they are equal to zero with negligible probability. Hence, with overwhelming probability,

$$\begin{aligned} \text{Decrypt}(T, \text{Rand}(\mathbf{c})) &= \sum_{\ell=1}^{\kappa} \frac{\mathbf{L}_{2\ell-1}^R(\mathbf{c})}{\mathbf{L}_{2\ell}^R(\mathbf{c})} + \frac{\nu_{\ell}(\mathbf{c})}{\nu_0(\mathbf{c})} \\ &= x_1 + \dots + x_{\kappa} + \frac{\nu_1(\mathbf{c}) + \dots + \nu_{\kappa}(\mathbf{c})}{\nu_0(\mathbf{c})} \\ &= x \end{aligned}$$

□

¹⁰ Recall that \mathbf{L}_j^R refers to the linear function defined by $\mathbf{L}_j^R(\mathbf{u}) = \langle \mathbf{r}_j, \mathbf{u} \rangle$ where \mathbf{r}_j is the j^{th} row of R .

6.2 Randomizing OGen (parameter γ)

Applying operators Rand allows to randomize encryptions. This randomization can be chained, i.e. by composing the operators $\text{Rand}^{T_1 \rightarrow T_2}$, $\text{Rand}^{T_2 \rightarrow T_3}$, \dots , $\text{Rand}^{T_\gamma \rightarrow S}$ where γ is a parameter indexed by λ . Following this idea, the function $\text{OGen}(S, i)$ can be naturally extended to $\text{OGen}(S, i, \gamma)$.

Definition 12. $\text{OGen}(S, i, \gamma)$ outputs the operators $\mathbf{O}_i^{S \rightarrow T_1}$ and $\left(\text{Rand}^{T_j \rightarrow T_{j+1}}\right)_{j=1, \dots, \gamma}$ where

- the matrices T_1, \dots, T_γ are randomly chosen and $T_{\gamma+1} = S$
- $\mathbf{O}_i^{S \rightarrow T_1} = T_1^{-1} S \mathbf{O}_i$ with $\mathbf{O}_i \leftarrow \text{OGen}(S, i)$
- $\text{Rand}^{T_j \rightarrow T_{j+1}} \leftarrow \text{RandGen}(T_j, T_{j+1})$.

The operator $\mathbf{O}_i^{[\gamma]} \leftarrow \text{OGen}(S, i, \gamma)$ is defined by

$$\mathbf{O}_i^{[\gamma]}(\mathbf{c}, \mathbf{c}') = \text{Rand}^{T_\gamma \rightarrow S} \circ \dots \circ \text{Rand}^{T_2 \rightarrow T_3} \circ \text{Rand}^{T_1 \rightarrow T_2} \circ \mathbf{O}_i^{S \rightarrow T_1}(\mathbf{c}, \mathbf{c}')$$

Thanks to Proposition 5, it is quite straightforward to show that $\mathbf{O}_i^{[\gamma]}$ is valid.

Proposition 6. Let $\mathbf{O}_i^{[\gamma]} \leftarrow \text{OGen}(S, i, \gamma)$. It is ensured that

$$\text{Decrypt}(S, \mathbf{O}_i^{[\gamma]}(\mathbf{c}, \mathbf{c}')) = \begin{cases} x + x' & \text{if } i = 0 \\ \pi_i & \text{otherwise} \end{cases}$$

Remark 5. By convention, $\mathbf{O}_i^{[0]}$ will refer to the basic operators defined in previous sections.

6.3 Efficiency

Each operator Rand runs in $O(\kappa^4)$. Thus, the running time of $\mathbf{O}_i^{S \rightarrow T_i}$ can be neglected. It follows that $\mathbf{O}_i^{[\gamma]}$ runs in

$$O(\gamma \kappa^4)$$

Hence, the additive operator runs in $O(\gamma \kappa^4)$ and the multiplicative one runs in $O(\gamma \kappa^5)$

7 Security analysis

Our homomorphic encryption scheme is parameterized by κ and γ . As discussed previously, to protect our scheme against attacks by linearization

$$\begin{aligned} \kappa &= \Theta(\lambda) \\ \gamma &> 0 \end{aligned}$$

Classically, to prove IND-CPA security (semantic security), we need to prove that the CPA attacker cannot efficiently decrypt the challenge encryption

$$\mathbf{c}_0 \leftarrow \text{Encrypt}(K, x_0 \leftarrow \{0, 1\})$$

from its knowledge.

7.1 Overview

In Section 7.2, a security result based on the factoring assumption is proposed. This is mainly an extension of Proposition 4. This result is useful to circumscribe the class of algebraic attacks, say, attacks dealing with Gröbner bases, by discarding some of them.

We then extensively analyze the security of our scheme against algebraic attacks. For concreteness, our scheme will be seen as a system of equations that can be classically solved with variable elimination techniques (classically based themselves on Gröbner bases). All our experiments will be done on **SageMath**. The running times suggest that the proposed attacks are not polynomial-time. While the results obtained are very encouraging, deeper investigations will be proposed to characterize the complexity of these attacks. As explained in Section 3, we will check that the degrees d_{max} and d_{reg} are not in $O(1)$ but linearly grow with κ, γ . Moreover, we will show that computing the Gröbner basis of some natural ideals is intrinsically exponential-time because the size of the obtained Gröbner basis is exponential-size itself. Finally, we will highlight the significant impact of symmetry on algebraic attacks.

7.2 A fundamental result (extension of Proposition 4)

We took care to preserve symmetry properties in the construction of the operators **Rand**. Thanks to this, Proposition 4 can be naturally extended. To achieve this, it suffices to include the internal randomness of **RandGen** in θ . For concreteness, θ_ℓ will include the coefficients of the polynomials ν_ℓ, η_ℓ and the rows 2ℓ and $2\ell + 1$ of each matrix involved in each operator **Rand**. Definition 10 and Lemma 2 can be straightforwardly extended by including the knowledge derived from these operators.

Lemma 3. *The CPA attacker's knowledge $(\mathbf{c}_0, \dots, \mathbf{c}_m, x_1, \dots, x_m, \mathbf{O}_0^{[\gamma]}, \dots, \mathbf{O}_\kappa^{[\gamma]})$ can be represented by a vector α , the i^{th} component of α being the evaluation of a Σ_κ -symmetric polynomial α_i over θ , i.e. $\alpha = (\alpha_1(\theta), \alpha_2(\theta), \dots) \stackrel{def}{=} \alpha(\theta)$.*

Proof. (Sketch.) It suffices to check that each operator **Rand** ^{$R \rightarrow T$} is stable by permuting $(\mathbf{r}_{2\ell-1}, \mathbf{r}_{2\ell}, \mathbf{t}_{2\ell-1}, \mathbf{t}_{2\ell}, \nu_\ell, \eta_\ell)_{\ell=1, \dots, \kappa}$ according to any $\sigma \in \Sigma_\kappa$. We conclude by invoking Lemma 2 for the remaining knowledge. □

By mixing Lemma 1 and Lemma 3, we get the following fundamental result.

Proposition 7. *Assuming the hardness of factoring, $\pi(\theta)$ cannot be evaluated, provided π is a polynomial that is not Σ_κ -symmetric. In particular, for any matrix T_j and any polynomial $\nu_{j\ell}, \eta_{j\ell}$ randomly (and secretly) chosen by **OGen**, the CPA attacker cannot recover any:*

1. coefficient of T_j ,
2. product of strictly less than κ coefficients of T_j ,
3. polynomial¹¹ $\mathbf{L}_{i_1}^{T_{j_1}} \times \dots \times \mathbf{L}_{i_t}^{T_{j_t}}$ provided $t < \kappa$,
4. polynomial $\nu_{j\ell}, \eta_{j\ell}$

Proof. A direct consequence of lemmas 1 and 3. □

Remark 6. This proposition does not ensure anything about the polynomials ν_0 . In consequence, throughout our security analysis, ν_0 will be assumed to be known, i.e.

$$\nu_0(\mathbf{u}) = u_1 + \dots + u_{2\kappa}$$

This result is fundamental. It can be used to discard some algebraic attacks¹². It excludes, for instance, algebraic attacks aiming to recover the secret matrices or the secret polynomials considered in the operators **Rand**. Indeed, thanks to symmetry, such attacks would lead to nonlinear univariate equations that cannot be solved under the factoring assumption.

¹¹ and thus cannot be evaluated

¹² which are efficient without the factoring assumption.

7.3 Representing CPA attacker's Knowledge by ideals of polynomials

Classically, solving a polynomial system $F = 0$ can be achieved by invoking algorithms dealing with the ideal $I = \langle F \rangle$ (see Section 3). In this section, a system of equations will be always *seen* as a polynomial ideal and reciprocally. The sum of two ideals $I = \langle F \rangle$ and $J = \langle G \rangle$ is defined by $I + J \stackrel{\text{def}}{=} \langle F \cup G \rangle$.

As seen in the previous section (see Lemma 3), the knowledge of the CPA attacker can be expressed as evaluations of Σ_κ -symmetric-polynomials α_i over θ . Hence, our scheme can be seen as the over-defined system of nonlinear equations

$$\begin{aligned}\alpha_1 - \alpha_1(\theta) &= 0 \\ \alpha_2 - \alpha_2(\theta) &= 0 \\ \dots &\end{aligned}$$

Moreover, the challenge encryption \mathbf{c}_0 is an encryption of an unknown value $x_0 \in \{0, 1\}$. This knowledge can be integrated into the previous system by introducing the variable x_0 and adding the two following equations:

$$\begin{aligned}x_{01} + \dots + x_{0\kappa} &= x_0 \\ x_0(x_0 - 1) &= 0\end{aligned}$$

Thanks to symmetry properties (see Lemma 3), the obtained system has at least κ solutions. Indeed, for any $\sigma \in \Sigma_\kappa$,

$$(x_0, \theta_{\sigma(1)}, \dots, \theta_{\sigma(\kappa)})$$

is a solution. In the following of this section, we will rewrite¹³ and detail these equations and we will group them in natural sub-systems/ideals.

Ideal I_0 (dealing with the challenge encryption). The CPA attacker first receives the challenge encryption \mathbf{c}_0 of $x_0 \in \{0, 1\}$. The ideal I_0 refers to the system of $\kappa + 2$ degree-2 equations satisfied by \mathbf{c}_0 and x_0 :

variables: s_{ij}, x_0, x_{0j}

$$\begin{aligned}- \langle \mathbf{s}_1, \mathbf{c}_0 \rangle - x_{01} \times \langle \mathbf{s}_2, \mathbf{c}_0 \rangle &= 0 \\ - \dots & \\ - \langle \mathbf{s}_{2\kappa-1}, \mathbf{c}_i \rangle - x_{0\kappa} \times \langle \mathbf{s}_{2\kappa}, \mathbf{c}_0 \rangle &= 0 \\ - x_{01} + \dots + x_{0\kappa} - x_0 &= 0 \\ - x_0(x_0 - 1) &= 0\end{aligned}$$

Ideal I_1 (dealing with the encryption oracle). The CPA attacker may request the encryption oracle to receive $\mathbf{c}_1, \dots, \mathbf{c}_m$ of chosen plaintexts x_1, \dots, x_m . From each encryption \mathbf{c}_i , the following system of $\kappa + 1$ degree-2 equations, denoted by $I_1(x_1, \dots, x_m)$, can be derived:

variables: s_{ij}, x_{ij}

$$\begin{aligned}- \langle \mathbf{s}_1, \mathbf{c}_i \rangle - x_{i1} \times \langle \mathbf{s}_2, \mathbf{c}_i \rangle &= 0 \\ - \dots & \\ - \langle \mathbf{s}_{2\kappa-1}, \mathbf{c}_i \rangle - x_{i\kappa} \times \langle \mathbf{s}_{2\kappa}, \mathbf{c}_i \rangle &= 0 \\ - x_{i1} + \dots + x_{i\kappa} - x_i &= 0\end{aligned}$$

¹³ To get polynomial-size equations.

Ideal I_2 (dealing with the homomorphic operators). Let \mathbf{O} be one of the operators output by OGen . For instance, consider $\mathbf{O} = (p_1, \dots, p_{2\kappa}) = \mathbf{O}_1^{S \rightarrow T}$ output by $\text{OGen}(S, 1, \gamma)$. By construction, the following 2κ polynomial equalities hold:

$$\begin{aligned} - \langle \mathbf{t}_1, (p_1, \dots, p_{2\kappa}) \rangle &= \mathbf{L}_{11}^S \\ - \dots & \\ - \langle \mathbf{t}_{2\kappa}, (p_1, \dots, p_{2\kappa}) \rangle &= \mathbf{L}_{2\kappa, 2\kappa}^S \end{aligned}$$

where \mathbf{t}_i refers to the i^{th} row of T . By equalizing each monomial coefficient, $4\kappa^2$ equations can be then derived from each equality. Hence, we get $8\kappa^3$ equations dealing with the variables s_{ij}, t_{ij} .

Example. For instance, by considering the first polynomial equality $\langle \mathbf{t}_1, (p_1, \dots, p_{2\kappa}) \rangle = \mathbf{L}_{11}^S$ and the monomial $u_1^2 v_1^2$, we obtain an equation in the form

$$a_1 t_{11} + \dots + a_{2\kappa} t_{1, 2\kappa} - s_{11}^2 = 0 \quad (4)$$

where $a_1, \dots, a_{2\kappa}$ are the monomial coefficient of $u_1^2 v_1^2$ of the public polynomials $p_1, \dots, p_{2\kappa}$. By considering the second polynomial equality, we notice that (\mathbf{t}_2, s_{21}) is also a solution of (4) i.e.

$$a_1 t_{21} + \dots + a_{2\kappa} t_{2, 2\kappa} - s_{21}^2 = 0$$

This is a direct consequence of symmetry.

This system of equations will be denoted by $I_2(\mathbf{O}_1^{S \rightarrow T=T_1})$. This can be straightforwardly extended to all the other public operators $\mathbf{O}_i^{S \rightarrow T_i}$.

The same can be done for each operator $(p_1, \dots, p_{2\kappa}) \leftarrow \text{Rand}^{R \rightarrow T}$ by considering the following 2κ polynomial equalities:

$$\begin{aligned} - \langle \mathbf{t}_{2\ell-1}, (p_1, \dots, p_{2\kappa}) \rangle &= \eta_\ell (\nu_0 \mathbf{L}_{2\ell-1}^R + \nu_\ell \mathbf{L}_{2\ell}^R) \\ - \langle \mathbf{t}_{2\ell}, (p_1, \dots, p_{2\kappa}) \rangle &= \eta_\ell \nu_0 \mathbf{L}_{2\ell}^R \end{aligned}$$

for any $\ell = 1, \dots, \kappa$. In the following of this section, I_2 will refer to the ideal coming from all the public operators output by OGen , i.e. $\mathbf{O}_i^{S \rightarrow T_i}$ and the $\gamma(\kappa+1)$ operator Rand . The variables involved in this ideal are the coefficients of all the matrices and all the polynomials η_ℓ, ν_ℓ involved in the operators output by OGen . According to remark 6, ν_0 will be assumed to be known in all our experiments, say $\nu_0(\mathbf{u}) = u_1 + \dots + u_{2\kappa}$.

7.4 Algebraic attacks

We here propose some attacks dealing with the ideals designed above. These attacks aim to solve or partially solve polynomial systems by using elimination variable techniques. Typically, these attacks aim to recover $x_0 \in \{0, 1\}$.

All these attacks were obviously implemented and evaluated. The implementation of these attacks simply consists of running SageMath's function named 'ideal_elimination(.)' on the above ideals.

It is important to note that we paid attention to memory swapping. In particular, we did not take into account running times when memory swapping was observed.

Attack 1. We here propose to attack the private-key encryption without considering the homomorphic operators. In other words, this attack deals with $I_0 + I_1$. For concreteness, it consists of decrypting the challenge encryption \mathbf{c}_0 knowing only encryptions $\mathbf{c}_1, \dots, \mathbf{c}_m$ of known plaintexts x_1, \dots, x_m : in particular without considering the homomorphic operators. This attack deals with the same system of equations as the one considered in the basic attack except that it is not *seen* as linear anymore.

- *Objective:* recovering x_0
- *Assumptions:* $s_{11} = 1$
- *Number of variables:* $4\kappa^2 + (m+1)\kappa + 1$

- *Number of equations:* $(m + 1)\kappa + 1$
- *Ideal:* $I_0 + I_1(x_1, \dots, x_m)$ where $(x_1, \dots, x_m) \xleftarrow{\$} \mathbf{Z}_n^m$
- *Eliminated variables:* all except x_0
- *Details.* The SageMath function `I.ideal_elimination([all variables except x_0])` with $I = I_0 + I_1$ outputs the ideal $\langle x_0 - 1 \rangle$ if $x_0 = 1$ and $\langle x_0 \rangle$ if $x_0 = 0$.
- *Experimental results.*

Consider now $\kappa = 2$. We obtain the following results

m	17	18	19	20	...	25
time(s)	952	378	155	2.1	...	4.8
d_{max}	6	6	6	5	...	5
d_{reg}	14	13	13	12	...	12

By noticing that $M(4, 2) = 10$, we notice a major threshold effect when

$$m < 2 \times M(2\kappa, \kappa)$$

In other words, when it is irrelevant to consider the system of equations as linear (because the number of equations is too small, i.e. strictly smaller than $2 \times M(2\kappa, \kappa)$, see Section 4.3), elimination techniques seem dramatically inefficient to recover x_0 . This would be sufficient to prove the inefficiency of such attacks by recalling that $M(2\kappa, \kappa)$ is exponential in κ . Because of prohibitive running times, this threshold effect was unfortunately not confirmed for higher values of κ , i.e. we did not obtain any result within $36h$ for $\kappa \geq 3$ except that d_{reg} grows quickly with κ , e.g. for any m , $d_{reg} \geq 42$ for $\kappa = 3$ and $d_{reg} \geq 112$ for $\kappa = 4$. In our opinion, this clearly suggests the inefficiency of these attacks.

- *Variant.* Consider only encryptions of 0, i.e. $x_1 = \dots = x_m = 0$ and test if (the polynomial) x_0 belongs to the ideal $I_0 + I_1$.
- *Analogy with LWE.* In Section 4.4, we have seen that breaking the semantic security of our private-key encryption and solving LWE are algebraically close problems. Albrecht et al. [ACF⁺12] proposed algebraic attacks of LWE (inspired by Arora and Ge's attack). Attack 1 can be seen as an adaptation of their attacks.

Attack 2. This attack aims to decrypt the challenge encryption \mathbf{c}_0 from the whole knowledge of the CPA attacker, i.e. considering the ideal $I_0 + I_1 + I_2$. The parameters κ, γ, m are thus involved in this attack.

- *Objective:* recovering x_0
- *Assumptions:* $s_{11} = 1$
- *Number of variables:* $O(\gamma\kappa^3)$
- *Number of equations:* $O(\gamma\kappa^4)$
- *Ideal:* $I_0 + I_1(x_1, \dots, x_m) + I_2$ where $(x_1, \dots, x_m) \xleftarrow{\$} \mathbf{Z}_n^m$
- *Eliminated variables:* all except x_0
- *Overview of the results.* We did not get any result within $36h$ for any $(\kappa, \gamma, m) \geq (3, 1, 0)$ (assuming the running times grow with the parameters).
- *Experimental results.* We first do not consider the encryption oracle, i.e. we consider the ideal $I_0 + I_2$. We obtain the following results ('me' means memory error and '-' means the running time is larger than $36h$).

κ	γ	0	1	2	3
2	time(s)	0.06	423	8200	179000
	d_{max}	4	10	me	me
	d_{reg}	3	9	17	26
3	time(s)	0.8	–	–	–
	d_{max}	4	–	–	–
	d_{reg}	3	13	26	39

Considering encryptions $\mathbf{c}_1, \dots, \mathbf{c}_m$ (represented by the ideal I_1) degrades performance.

- *Analysis.* The operators **Rand** were introduced as a countermeasure against attacks by linearization (see Appendix B). A major effect of these operators against algebraic attacks can also be observed. While the attack is quite instantaneous for $\gamma = 0$, it takes more than $36h$ for $\gamma \geq 1$ and $\kappa \geq 3$. In our opinion, this clearly validates the fundamental role played by the parameter γ .

As expected, d_{max} and d_{reg} grow with κ, γ . This enhances the idea that this attack is not polynomial-time. The next attack can be seen as a variant of this one. While it is experimentally more efficient than this one, evidence about its exponential-time complexity will be given.

- *Variante.* Considering only a subset of operators output by **OGen**.

Attack 3. This attack aims to decrypt the challenge encryption \mathbf{c}_0 , considering a sub-ideal $I \subset I_0 + I_1 + I_2(\text{Add}^{S \rightarrow T})$. In particular, the operators **Rand** are not considered. While this is experimentally the most efficient attack that we tested, we will provide strong evidence about its inefficiency.

- *Objective:* recovering x_0
- *Assumptions:* $s_{11} = 1$
- *Ideal:* $I \subset I_0 + I_1 + I_2(\text{Add}^{S \rightarrow T})$. More precisely, I contains the following polynomials:
 - $\langle \mathbf{s}_1, \mathbf{c}_i \rangle - x_{i1} \times \langle \mathbf{s}_2, \mathbf{c}_i \rangle$ for any $i = 0, \dots, m$
 - the polynomials coming from the two polynomials equalities

$$\begin{aligned} \langle \mathbf{t}_1, (p_1, \dots, p_{2\kappa}) \rangle &= \mathbf{L}_{12} + \mathbf{L}_{21} \\ \langle \mathbf{t}_2, (p_1, \dots, p_{2\kappa}) \rangle &= \mathbf{L}_{22} \end{aligned}$$

where $(p_1, \dots, p_{2\kappa}) = \text{Add}^{S \rightarrow T}$.

- *Intuition:* reducing the number of involved variables
- *Number of variables:* $8\kappa + m + 1$
- *Number of polynomials:* $4\kappa^2 + m + 1$
- *Eliminated variables:* all except $x_{01}, x_{11}, \dots, x_{m,1}$
- *Obtained results:* a linear combination between $x_{01}, x_{11}, \dots, x_{m,1}$
- *Remark.* The parameter γ is not involved.
- *Details.* By eliminating all the variables except $x_{01}, x_{11}, \dots, x_{m,1}$, we expect to obtain a linear equation between these variables, i.e.

$$a_0 x_{01} + a_1 x_{11} + \dots + a_m x_{m,1} = 0$$

Symmetry ensures that the variables $x_{0i}, x_{1i}, \dots, x_{m,i}$ satisfy the same equation for any $i = 1, \dots, \kappa$. It follows that this equation is also satisfied by x_0, x_1, \dots, x_m , i.e.

$$a_0 x_0 + a_1 x_1 + \dots + a_m x_m = 0$$

As x_1, \dots, x_m are known values, x_0 could be thus recovered. Clearly, as m variables should satisfy κ equations, m should be greater than κ . We experimentally observe that it works provided

$$m \geq \kappa + 2$$

- *Experimental results.* In our experiments, we choose $m = \kappa + 2$ which is experimentally optimal.

κ	2	3	4	5	6	7	...	10
time(s)	10	50	647	3500	25700	—	...	—
d_{max}	6	7	8	9	10	—	...	—
d_{reg}	5	5	5	6	6	7	...	10

- *Exponential-size of the grevlex-Gröbner basis.* We compute the (reduced) *grevlex*-Gröbner basis G for $\kappa = 2, 3, 4, 5, 6$. By carefully examining these bases, one can make the following fundamental observation:

For any $E \in \{0, \dots, \kappa + 1\}$, there is a multiple of the monomial¹⁴ $\prod_{i \in E} x_{i1}$ which is involved in at least one polynomial of G .

This straightforwardly implies that the size of G is larger than 2^κ . It is tempting to assume that these observations can be extrapolated to larger values of κ . Under this assumption, **we could conclude that computing G is exponential-time**. In our opinion, it is perhaps the most convincing evidence in favor of the inefficiency of Gröbner basis attacks.

- *The fundamental role of symmetry.* The attack considered in the section clearly seems exponential-time. We here propose to examine the crucial role played by symmetry (see Lemma 2) in these results.

To achieve it, we remove symmetry by modifying the operator **Add** as follows:

$$\text{Add}^{S \rightarrow T} = \begin{pmatrix} \mathbf{L}_{12} + \mathbf{L}_{21} \\ \mathbf{L}_{22} \\ \text{randomPoly}_1 \\ \dots \\ \text{randomPoly}_{2\kappa-2} \end{pmatrix}$$

where randomPoly_i are randomly chosen degree-2 polynomials. The implementation of the attack does not require any modification. Moreover, the polynomials and the shape of the involved polynomials are exactly the same (with exactly the same monomials). We obtain the following results:

κ	2	3	4	5	6
time(s)	0.04	0.18	0.28	0.37	0.46
d_{max}	3	3	3	4	4
d_{reg}	3	3	3	3	3

This attack clearly becomes very efficient. This clearly suggests the fundamental role played by symmetry, In particular, d_{reg} does not grow with κ anymore while symmetry experimentally ensures that $d_{reg} = \Theta(\kappa)$.

- *Variante.* Sampling $(\mathbf{u}, \mathbf{v}) \leftarrow \mathbf{Z}_n^{2\kappa} \times \mathbf{Z}_n^{2\kappa}$, computing $\mathbf{w} = \text{Add}^{S \rightarrow T}(\mathbf{u}, \mathbf{v})$ and replacing $I_2(\text{Add}^{S \rightarrow T})$ by the equations coming from the following equality

$$\frac{\langle \mathbf{s}_1, \mathbf{u} \rangle}{\langle \mathbf{s}_2, \mathbf{u} \rangle} + \frac{\langle \mathbf{s}_1, \mathbf{v} \rangle}{\langle \mathbf{s}_2, \mathbf{v} \rangle} = \frac{\langle \mathbf{t}_1, \mathbf{w} \rangle}{\langle \mathbf{t}_2, \mathbf{w} \rangle}$$

¹⁴ For instance $s_{11}x_{11}x_{31}x_{41}$

Hybrid attacks. *More guided* strategies can be imagined. Typically, one can be interested in recovering intermediate values with algebraic attacks before running efficient attacks by linearization. For instance, each monomial coefficient of ϕ_S is Σ_κ -symmetric¹⁵ and could be thus recovered (see Proposition 7). However, as mentioned above, ϕ_S is exponential-size provided $\kappa = \Theta(\lambda)$.

The CPA attacker could also be interested in recovering the naive/basic operators \mathbf{O}_i (see Section 5). Indeed, the knowledge of these operators leads to efficient attacks by linearization. The CPA attacker knows $\mathbf{O}_i^{S \rightarrow T_i}$. Hence, it could be interested in recovering $A_i = S^{-1}T_i$ by noticing that

$$\mathbf{O}_i = A_i \mathbf{O}_i^{S \rightarrow T_i}$$

To achieve this, we introduce new variables a_{ijk} representing $A_i = [a_{ijk}]$. Then, we add the quadratic polynomials coming from the matrix equality $SA_i = T_i$, i.e. for any $j, k \in \{1, \dots, 2\kappa\}$,

$$\sum_{\ell=1}^{\kappa} s_{j\ell} a_{i\ell k} - t_{ijk} = 0$$

where $S = [s_{jk}]$ and $T_i = [t_{ijk}]$, to the ideal $I_0 + I_1 + I_2$ or to sub-ideals. We did not manage to recover A_i by this way. Moreover, this attack appears inefficient in the sense that computing *grevlex*-Gröbner basis is experimentally totally inefficient.

Imagine that it were not the case, i.e. A_i can be efficiently recovered. The naive operators could thus be efficiently recovered by running this attack. It would then suffice to run Attack 2, which was shown efficient with the naive operators (dealing with the case $\gamma = 0$). This would lead to an algorithm much more efficient than the one proposed by SageMath to eliminate variables in $I_0 + I_1 + I_2$. This would contradict the implicit idea underlying our security analysis: the algorithms implemented in SageMath are the most efficient ones to compute Gröbner bases.

In [BFP12], Bettale et al. propose to combine exhaustive search (by fixing some variables) with Gröbner basis. The efficiency of this hybrid approach is related to the choice of a trade-off between the two methods. Nevertheless, the exhaustive search is here totally inefficient because n is assumed to be large.

Subgroup attacks. Let $\mathcal{C} \subset \mathbf{Z}_n^{2\kappa}$ denote the set of valid encryptions and let \oplus be the naive operator Add (dealing with the case $\gamma = 0$). It is not difficult to show that (\mathcal{C}, \oplus) is an abelian group. Moreover, the set $\mathcal{C}_0 \subset \mathcal{C}$ of encryptions of 0, i.e.

$$\mathcal{C}_0 = \{c \in \mathcal{C} \mid \text{Decrypt}(S, c) = 0\}$$

is a subgroup of \mathcal{C} . This scheme could be subject to subgroup attacks consisting of distinguishing between \mathcal{C}_0 and \mathcal{C} . In particular, there is an efficient quantum algorithm for solving Hidden Subgroup Problem (HSP).

However, thanks to the operators Rand, it does not work anymore provided $\gamma > 0$. In this case, (\mathcal{C}, \oplus) is not a group anymore: \oplus is not associative, there does not exist an identity element e and it is even not stable ($c \oplus c'$ may be an invalid encryption).

7.5 Discussion

In this section, we mainly investigate the security of our scheme against algebraic attacks. We obviously tested many variants of the above attacks. We systematically obtained results consistent with the ones presented here.

The running times obtained in our experiments suggest that these attacks are not polynomial-time. This is obviously not sufficient. To go further, we investigated this question more formally.

We exhibited the fundamental role played by symmetry. Under the factoring assumption, this property discards a large class of algebraic attacks aiming to recover non-symmetric values (see Proposition 7). Moreover, this property ensures that the systems of equations considered in our attacks have at least κ solutions. This ensures that the degree d_{\max} reached by the polynomials considered during the computation of Gröbner bases grows with κ . Attack 3 (which can be seen as a variant of Attack 2) appears to be the most efficient attack that we tested. Strong evidence in favor of its inefficiency was provided.

Claim 1. *Assuming the hardness of factoring, algebraic attacks based on Gröbner bases are not efficient provided $\kappa = \Theta(\lambda)$ and $\gamma > 0$.*

¹⁵ can be expressed as an evaluation of Σ_κ -symmetric polynomials over θ .

7.6 Efficiency

As mentioned above, the running time of **Add** is $O(\gamma\kappa^3)$ and the running time of **Mult** is $O(\gamma\kappa^4)$.

Attack 3 is the most efficient tested attack. By extrapolating our experimental results, it would suffice to choose $\kappa = 30$ and $\gamma = 1$ for a security parameter $\lambda = 100$. Under this choice, **Add** would require approximately 2.3×10^6 multiplications in \mathbf{Z}_n and **Mult** approximately 10^8 multiplications.

It should be noticed that evaluating these operators consists of evaluating expanded polynomials. It can thus be highly parallelized. Moreover, some significant improvements can be imagined, but this is not the object of this paper.

8 Future Work / Open questions

Several ways can be chosen to extend our work. The most formal one would consist of formally reducing a classical cryptographic problem to the security of our scheme. According to our security analysis, we are convinced that symmetry should play a central role in such a reduction.

Algebraic attacks based on Gröbner basis should be deeper investigated. In our opinion, it is a nice cryptanalysis challenge. More generally, new attacks should be experimented with.

The factorization of n is not used in **KeyGen**. The factoring assumption is mainly introduced to ensure that univariate nonlinear equations cannot be solved. The security of our private-key encryption (without considering the homomorphic operators) is *a priori* not related to the factoring assumption. In section 4.4, we have seen a strong link between our scheme and **LWE**. Further investigations should be done in this sense.

Nevertheless, the factoring assumption is required to make some algebraic attacks based on Gröbner bases irrelevant. For instance, let us consider the public operator $\mathbf{O}_1^{S \rightarrow T} = (p_1, \dots, p_{2\kappa})$. Without the factoring assumption, the rows \mathbf{t}_ℓ could be efficiently recovered¹⁶ by solving the polynomial system derived from the following equality

$$\langle \mathbf{t}_\ell, (p_1, \dots, p_{2\kappa}) \rangle = \eta_\ell \nu_0 \mathbf{L}_{\ell, \ell}$$

Indeed, by eliminating all the variables except $t_{\ell,1}$ (for instance), we obtain a nonlinear equation in $t_{\ell,1}$ which can be efficiently solved in finite fields.

To remove the factoring assumption, it is necessary to make such variable elimination inefficient. In our opinion, there is hope to remove this assumption by adding randomness in the generation of the homomorphic operators. We carried out preliminary experiments in this sense, leading to promising results. Roughly speaking, we add *redundancy* in encryptions in order to add randomness in homomorphic operators. This is detailed in Appendix C.

Finally, our scheme can be straightforwardly turned into a HE over real numbers. To achieve this, it suffices to choose S over the reals, e.g. $S \leftarrow [0, 1]^{2\kappa \times 2\kappa}$. Indeed, \mathbf{c} and $a \times \mathbf{c}$ are encryptions of the same value for any $a \in \mathbf{R}^*$. Moreover, \mathbf{c} and $\mathbf{c} + \varepsilon$ are encryptions of close values. Hence, the ciphertexts can be normalized after each homomorphic operation, avoiding ciphertext-size leaks information. We carried out promising experiments in this sense.

References

- [ACF⁺12] Martin Albrecht, Carlos Cid, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret. On the complexity of the Arora-Ge Algorithm against **LWE**. In *SCC 2012 – Third international conference on Symbolic Computation and Cryptography*, pages 93–99, Castro Urdiales, Spain, July 2012.
- [AG11a] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*, pages 403–415, 2011.
- [AG11b] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jiří Sgall, editors, *Automata, Languages and Programming*, pages 403–415, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

¹⁶ This is excluded under the factoring assumption according to Proposition 7.

- [AM09] Divesh Aggarwal and Ueli M. Maurer. Breaking RSA generically is equivalent to factoring. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, pages 36–53, 2009.
- [BFP12] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Solving Polynomial Systems over Finite Fields: Improved Analysis of the Hybrid Approach. In *ISSAC 2012 - 37th International Symposium on Symbolic and Algebraic Computation*, pages 67–74, Grenoble, France, July 2012. ACM.
- [BRS67] Elwyn R. Berlekamp, H. Rumsey, and G. Solomon. On the solution of algebraic equations over finite fields. *Information and Control*, 10(6):553–564, 1967.
- [Buc06] Bruno Buchberger. Bruno buchberger’s phd thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. *Journal of Symbolic Computation*, 41(3):475–511, 2006. Logic, Mathematics and Computer Science: Interactions in honor of Bruno Buchberger (60th birthday).
- [CGGI18] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: fast fully homomorphic encryption over the torus. *IACR Cryptology ePrint Archive*, 2018:421, 2018.
- [CLO91] David Cox, John Little, and Donald O’Shea. *Ideals, Varieties and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, 1991.
- [CNT12] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *EUROCRYPT*, pages 446–464, 2012.
- [DM15] Léoucas and Daniele Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. 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*, pages 617–640, 2015.
- [Fau99] Jean-Charles Faugère. A new efficient algorithm for computing groebner bases (f4). *Journal of Pure and Applied Algebra*, 139(1):61–88, 1999.
- [Gar97] Michael R Garey. Computers and intractability: A guide to the theory of np-completeness, freeman. *Fundamental*, 1997.
- [Gav16] Gérard Gavin. An efficient somewhat homomorphic encryption scheme based on factorization. Cryptology ePrint Archive, Report 2016/897, 2016. <http://eprint.iacr.org/2016/897>.
- [GB19] Gérard Gavin and Stéphane Bonneveay. Fractional lwe: a nonlinear variant of lwe. Cryptology ePrint Archive, Report 2019/2502, 2019. <http://eprint.iacr.org/2019/2502>.
- [Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.
- [Gen21] Craig Gentry. A decade (or so) of fully homomorphic encryption, 2021. Invited talk.
- [GHS12a] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In *EUROCRYPT*, pages 465–482, 2012.
- [GHS12b] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the aes circuit. In *CRYPTO*, pages 850–867, 2012.
- [GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 75–92, 2013.
- [GT20] Gérard Gavin and Sandrine Tainturier. New ideas to build noise-free homomorphic cryptosystems. In Abderrahmane Nitaaj and Amr M. Youssef, editors, *Progress in Cryptology - AFRICACRYPT 2020 - 12th International Conference on Cryptology in Africa, Cairo, Egypt, July 20-22, 2020, Proceedings*, volume 12174 of *Lecture Notes in Computer Science*, pages 423–451. Springer, 2020.
- [LNV11] Kristin Lauter, Michael Naehrig, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? *IACR Cryptology ePrint Archive*, 2011:405, 2011.
- [Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93, 2005.
- [Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. 56(6), 2009.
- [Rot11] Ron Rothblum. *Homomorphic Encryption: From Private-Key to Public-Key*, pages 219–234. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
- [SS10] Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In *ASIACRYPT*, pages 377–394, 2010.
- [vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT*, pages 24–43, 2010.

A Implementation of the basic Add in the case $\kappa = 1$

In this section, we provide an example of the implementation of the homomorphic scheme for $\kappa = 1$. Let $S = [s_{ij}] \in \mathbf{Z}_n^{2 \times 2}$ and $\Delta = s_{11}s_{22} - s_{12}s_{21}$.

$\text{Add} = (p_1, p_2) \leftarrow \text{OGen}(S, i)$ is defined by

$$\begin{aligned} \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} &= S^{-1} \begin{pmatrix} \langle \mathbf{s}_1, \mathbf{u} \rangle \langle \mathbf{s}_2, \mathbf{v} \rangle + \langle \mathbf{s}_2, \mathbf{u} \rangle \langle \mathbf{s}_1, \mathbf{v} \rangle \\ \langle \mathbf{s}_2, \mathbf{u} \rangle \langle \mathbf{s}_2, \mathbf{v} \rangle \end{pmatrix} \\ &= S^{-1} \begin{pmatrix} \langle \mathbf{s}_1, \mathbf{u} \rangle \langle \mathbf{s}_2, \mathbf{v} \rangle + \langle \mathbf{s}_2, \mathbf{u} \rangle \langle \mathbf{s}_1, \mathbf{v} \rangle \\ \langle \mathbf{s}_2, \mathbf{u} \rangle \langle \mathbf{s}_2, \mathbf{v} \rangle \end{pmatrix} \\ &= S^{-1} \begin{pmatrix} 2s_{11}s_{21}u_1v_1 + (s_{11}s_{22} + s_{21}s_{12})(u_1v_2 + u_2v_1) + 2s_{12}s_{22}u_2v_2 \\ s_{21}^2u_1v_1 + s_{21}s_{22}(u_1v_2 + u_2v_1) + s_{22}^2u_2v_2 \end{pmatrix} \\ &= \Delta^{-1} \begin{bmatrix} s_{22} & -s_{12} \\ -s_{21} & s_{11} \end{bmatrix} \begin{pmatrix} 2s_{11}s_{21}u_1v_1 + (s_{11}s_{22} + s_{21}s_{12})(u_1v_2 + u_2v_1) + 2s_{12}s_{22}u_2v_2 \\ s_{21}^2u_1v_1 + s_{21}s_{22}(u_1v_2 + u_2v_1) + s_{22}^2u_2v_2 \end{pmatrix} \end{aligned}$$

It follows that

$$\begin{aligned} \Delta \cdot p_1(\mathbf{u}, \mathbf{v}) &= (2s_{22}s_{11}s_{21} - s_{12}s_{21}^2)u_1v_1 \\ &\quad + s_{22}^2s_{11}(u_1v_2 + u_2v_1) \\ &\quad + s_{12}s_{22}^2u_2v_2 \end{aligned}$$

$$\begin{aligned} \Delta \cdot p_2(\mathbf{u}, \mathbf{v}) &= -s_{11}s_{21}^2u_1v_1 \\ &\quad - s_{21}^2s_{12}(u_1v_2 + u_2v_1) \\ &\quad + (s_{11}s_{22}^2 - 2s_{21}s_{12}s_{22})u_2v_2 \end{aligned}$$

Numerical application.

$$- n = 5, \kappa = 1$$

$$- S = \begin{bmatrix} 3 & 1 \\ 2 & 1 \end{bmatrix}; S^{-1} = \begin{bmatrix} 1 & 4 \\ 3 & 3 \end{bmatrix}$$

$$- \text{Add}(\mathbf{u}, \mathbf{v}) = S^{-1} \begin{pmatrix} (u_1 + 4u_2)(3v_1 + 3v_2) + (3u_1 + 3u_2)(v_1 + 4v_2) \\ (3u_1 + 3u_2)(3v_1 + 3v_2) \end{pmatrix} = \begin{pmatrix} 3u_1v_1 + 3(u_2v_1 + u_1v_2) + u_2v_2 \\ 3u_1v_1 + (u_2v_1 + u_1v_2) + 4u_2v_2 \end{pmatrix}$$

$$- \mathbf{c}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}; \mathbf{c}_2 = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

$$- \text{Decrypt}(S, \mathbf{c}_1) = \frac{3 \times 1 + 1 \times 1}{2 \times 1 + 1 \times 1} = 3; \text{Decrypt}(S, \mathbf{c}_2) = \frac{3 \times 2 + 1 \times 3}{2 \times 2 + 1 \times 3} = 2$$

$$- \mathbf{c}_3 = \text{Add}(\mathbf{c}_1, \mathbf{c}_2) = \begin{pmatrix} p_1(\mathbf{c}_1, \mathbf{c}_2) \\ p_2(\mathbf{c}_1, \mathbf{c}_2) \end{pmatrix} = \begin{pmatrix} 3 \times 1 \times 2 + 3(1 \times 2 + 1 \times 3) + 1 \times 3 \\ 3 \times 1 \times 2 + (1 \times 2 + 1 \times 3) + 4 \times 1 \times 3 \end{pmatrix} = \begin{pmatrix} 4 \\ 3 \end{pmatrix} = \text{Add}(\mathbf{c}_2, \mathbf{c}_1)$$

$$- \text{Decrypt}(S, \mathbf{c}_3) = \frac{3 \times 4 + 1 \times 3}{2 \times 4 + 1 \times 3} = 0 = \text{Decrypt}(S, \mathbf{c}_1) + \text{Decrypt}(S, \mathbf{c}_2)$$

B Attacks by linearization

Attack 1. By definition of our operator \otimes , we can write ($\mathbf{u} \sim \mathbf{v}$ meaning that $\exists k$ s.t. $\mathbf{u} = k\mathbf{v}$),

$$\mathbf{c} \otimes \mathbf{c} \sim S^{-1} \begin{pmatrix} r_1^\kappa x_1(x_1 + x_2 + \dots + x_\kappa) \\ r_1^\kappa \\ \dots \\ r_\kappa^\kappa x_\kappa(x_1 + x_2 + \dots + x_\kappa) \\ r_\kappa^\kappa \end{pmatrix} \sim S^{-1} \begin{pmatrix} r_1^\kappa x_1 x \\ r_1^\kappa \\ \dots \\ r_\kappa^\kappa x_\kappa x \\ r_\kappa^\kappa \end{pmatrix}$$

This is obviously a disaster in term of security. Indeed, if \mathbf{c} is an encryption of 0 then

$$\mathbf{c} \otimes \mathbf{c} \sim S^{-1} \begin{pmatrix} 0 \\ r_1^\kappa \\ \dots \\ 0 \\ r_\kappa^\kappa \end{pmatrix}$$

Let $\mathbf{c}_1, \dots, \mathbf{c}_\kappa$ be encryptions of 0. To test whether a challenge encryption \mathbf{c} is an encryption of 0, it suffices to check that $\mathbf{c} \otimes \mathbf{c}$ belongs to the vectorial space spanned by the vectors $\mathbf{c}_1 \otimes \mathbf{c}_1, \dots, \mathbf{c}_\kappa \otimes \mathbf{c}_\kappa$.

Attack 2. Let us consider the operator \mathbf{O}_1 defined in the previous section, i.e. applying \mathbf{O}_1 consists of evaluating the polynomials $p_1, \dots, p_{2\kappa}$ defined by

$$\begin{pmatrix} p_1 \\ \dots \\ p_{2\kappa} \end{pmatrix} = S^{-1} \begin{pmatrix} \mathbf{L}_{11} \\ \mathbf{L}_{22} \\ \dots \\ \mathbf{L}_{2\kappa, 2\kappa} \end{pmatrix}$$

As the vector $\mathbf{v} = S^{-1}(0, 1, 0, 1, \dots, 0, 1)$ is the unique vector satisfying $\text{Add}(\mathbf{u}, \mathbf{v}) = \mathbf{u}$ for any valid encryption \mathbf{u} , it can be recovered by solving a linear system. It follows that $\mathbf{O}_1(\mathbf{c}, \mathbf{v})$ is equal to the vector $\mathbf{w} = S^{-1}(0, r_1, 0, r_2, \dots, 0, r_\kappa)$. By solving the equation $\text{Add}(\mathbf{u}, \mathbf{w}) = \mathbf{c}$, we get the vector

$$\tilde{\mathbf{c}} = S^{-1} \begin{pmatrix} x_1 \\ 1 \\ \dots \\ x_\kappa \\ 1 \end{pmatrix}$$

satisfying $\langle \mathbf{s}_1 + \mathbf{s}_3 + \dots + \mathbf{s}_{2\kappa-1}, \tilde{\mathbf{c}} \rangle = x$. This leads to an efficient attack consisting of solving a linear system of size 2κ .

Attack 3. Let us consider the vector $\mathbf{v} = \mathbf{u}_{\kappa-1}$ built as follows:

- $\mathbf{u}_0 = \mathbf{c}$
- $\mathbf{u}_i = \mathbf{O}_{i+1}(\mathbf{u}_{i-1}, \mathbf{c})$ for any $i = 1, \dots, \kappa - 1$

By construction,

$$\mathbf{v} = S^{-1} \begin{pmatrix} r_1 \dots r_{\kappa-1} x_1 \dots x_{\kappa-1} \\ r_1 \dots r_{\kappa-1} \\ r_2 \dots r_\kappa x_2 \dots x_\kappa \\ r_2 \dots r_\kappa \\ \dots \\ r_\kappa r_1 \dots r_{\kappa-2} x_\kappa x_1 \dots x_{\kappa-2} \\ r_\kappa r_1 \dots r_{\kappa-2} \end{pmatrix}$$

It follows that

$$\begin{aligned} \varphi(\mathbf{c}, \mathbf{v}) &\stackrel{\text{def}}{=} \mathbf{L}_{14}(\mathbf{c}, \mathbf{v}) + \mathbf{L}_{36}(\mathbf{c}, \mathbf{v}) + \dots + \mathbf{L}_{2\kappa-1, 2}(\mathbf{c}, \mathbf{v}) \\ &= r_1 \dots r_\kappa (x_1 + \dots + x_\kappa) \\ &= \phi_S(\mathbf{c}) \end{aligned}$$

can be used to distinguish encryptions of 0 from random ones. Moreover, as this polynomial is quadratic, its expanded representation can be polynomially recovered by solving a linear system (by considering sufficiently many encryptions of 0).

C Removing the factoring assumption

Throughout this section, n will be a large prime. Proposition 7 excludes some algebraic attacks which become relevant without the factoring assumption (see Section 8). In order to make these attacks irrelevant, we propose to redundancy in encryptions allowing to add randomness in homomorphic operators. The idea is to exploit the following assertion:

$$\frac{a}{b} = \frac{a'}{b'} \Rightarrow \frac{a}{b} = \frac{\eta_1 a + \eta_2 a'}{\eta_1 b + \eta_2 b'} \quad (5)$$

ensuring that $\eta_1 b + \eta_2 b' \neq 0$. Concretely, an encryption $\mathbf{c} \in \mathbf{Z}_n^{2\tau\kappa}$ could be defined as follows

$$\mathbf{c} = S^{-1} \begin{pmatrix} r_{11}x_1 \\ r_{11} \\ \dots \\ r_{1\tau}x_1 \\ r_{1\tau} \\ \dots \\ r_{\kappa,1}x_\kappa \\ r_{\kappa,1} \\ \dots \\ r_{\kappa,\tau}x_\kappa \\ r_{\kappa,\tau} \end{pmatrix}$$

Equivalently, we have

$$X(\mathbf{c}) = (\underbrace{x_1, \dots, x_1}_\tau, \underbrace{x_2, \dots, x_2}_\tau, \dots, \underbrace{x_\kappa, \dots, x_\kappa}_\tau)$$

$$R(\mathbf{c}) = (r_{11}, \dots, r_{1\tau}, r_{21}, \dots, r_{2\tau}, \dots, r_{\kappa 1}, \dots, r_{\kappa\tau})$$

The homomorphic operators should be then adapted. Assertion 5 can be used to dramatically increase internal randomness of these operators. For instance, Assertion 5 ensures that

$$\frac{\eta_{11}\mathbf{L}_{11}(\mathbf{c}, \mathbf{c}') + \eta_{12}\mathbf{L}_{33}(\mathbf{c}, \mathbf{c}')}{\eta_{11}\mathbf{L}_{22}(\mathbf{c}, \mathbf{c}') + \eta_{12}\mathbf{L}_{44}(\mathbf{c}, \mathbf{c}')} = \frac{\eta_{11}r_{11}r'_{11}x_1x'_1 + \eta_{12}r_{12}r'_{12}x_1x'_1}{\eta_{11}r_{11}r'_{11} + \eta_{12}r_{12}r'_{12}} = x_1x'_1$$

By extending this idea, the operator $\mathbf{O}_1 = (p_1, \dots, p_{2\kappa})$ in the case $\kappa = \tau = 2$ (extension to the other operators for any $\kappa \geq 2, \tau \geq 2$ is straightforward) can be redefined as follows:

$$\begin{pmatrix} p_1 \\ \dots \\ p_8 \end{pmatrix} = S^{-1} \begin{pmatrix} \nu_0 (\eta_{11}\mathbf{L}_{11} + \eta_{12}\mathbf{L}_{33} + \eta_{13}\mathbf{L}_{13} + \eta_{14}\mathbf{L}_{31}) + \nu_1 (\eta_{11}\mathbf{L}_{22} + \eta_{12}\mathbf{L}_{44} + \eta_{13}\mathbf{L}_{24} + \eta_{14}\mathbf{L}_{42}) \\ \nu_0 (\eta_{11}\mathbf{L}_{22} + \eta_{12}\mathbf{L}_{44} + \eta_{13}\mathbf{L}_{24} + \eta_{14}\mathbf{L}_{42}) \\ \nu_0 (\eta_{21}\mathbf{L}_{11} + \eta_{22}\mathbf{L}_{33} + \eta_{23}\mathbf{L}_{13} + \eta_{24}\mathbf{L}_{31}) + \nu_1 (\eta_{21}\mathbf{L}_{22} + \eta_{22}\mathbf{L}_{44} + \eta_{23}\mathbf{L}_{24} + \eta_{24}\mathbf{L}_{42}) \\ \nu_0 (\eta_{21}\mathbf{L}_{22} + \eta_{22}\mathbf{L}_{44} + \eta_{23}\mathbf{L}_{24} + \eta_{24}\mathbf{L}_{42}) \\ \nu_0 (\eta_{31}\mathbf{L}_{55} + \eta_{32}\mathbf{L}_{77} + \eta_{33}\mathbf{L}_{57} + \eta_{34}\mathbf{L}_{75}) + \nu_2 (\eta_{31}\mathbf{L}_{66} + \eta_{32}\mathbf{L}_{88} + \eta_{33}\mathbf{L}_{68} + \eta_{34}\mathbf{L}_{86}) \\ \nu_0 (\eta_{31}\mathbf{L}_{66} + \eta_{32}\mathbf{L}_{88} + \eta_{33}\mathbf{L}_{68} + \eta_{34}\mathbf{L}_{86}) \\ \nu_0 (\eta_{41}\mathbf{L}_{55} + \eta_{42}\mathbf{L}_{77} + \eta_{43}\mathbf{L}_{57} + \eta_{44}\mathbf{L}_{75}) + \nu_2 (\eta_{41}\mathbf{L}_{66} + \eta_{42}\mathbf{L}_{88} + \eta_{43}\mathbf{L}_{68} + \eta_{44}\mathbf{L}_{86}) \\ \nu_0 (\eta_{41}\mathbf{L}_{66} + \eta_{42}\mathbf{L}_{88} + \eta_{43}\mathbf{L}_{68} + \eta_{44}\mathbf{L}_{86}) \end{pmatrix}$$

where $\nu_0, \nu_1, \nu_2, \eta_{ij}$ are polynomials satisfying $\nu_1 + \nu_2 = 0$. By construction, it is ensured that $\mathbf{O}_1(\mathbf{c}, \mathbf{c}')$ is equal to

$$S^{-1} \begin{pmatrix} \nu_0 x_1 x'_1 (\eta_{11} r_{11} r'_{11} + \eta_{12} r_{12} r'_{12} + \eta_{13} r_{11} r'_{12} + \eta_{14} r_{12} r'_{11}) + \nu_1 (\eta_{11} r_{11} r'_{11} + \eta_{12} r_{12} r'_{12} + \eta_{13} r_{11} r'_{12} + \eta_{14} r_{12} r'_{11}) \\ \nu_0 (\eta_{11} r_{11} r'_{11} + \eta_{12} r_{12} r'_{12} + \eta_{13} r_{11} r'_{12} + \eta_{14} r_{12} r'_{11}) \\ \nu_0 x_1 x'_1 (\eta_{21} r_{11} r'_{11} + \eta_{22} r_{12} r'_{12} + \eta_{23} r_{11} r'_{12} + \eta_{24} r_{12} r'_{11}) + \nu_1 (\eta_{21} r_{11} r'_{11} + \eta_{22} r_{12} r'_{12} + \eta_{23} r_{11} r'_{12} + \eta_{24} r_{12} r'_{11}) \\ \nu_0 (\eta_{21} r_{11} r'_{11} + \eta_{22} r_{12} r'_{12} + \eta_{23} r_{11} r'_{12} + \eta_{24} r_{12} r'_{11}) \\ \nu_0 x_2 x'_2 (\eta_{31} r_{21} r'_{21} + \eta_{32} r_{22} r'_{22} + \eta_{33} r_{21} r'_{22} + \eta_{34} r_{22} r'_{21}) + \nu_2 (\eta_{31} r_{21} r'_{21} + \eta_{32} r_{22} r'_{22} + \eta_{33} r_{21} r'_{22} + \eta_{34} r_{22} r'_{21}) \\ \nu_0 (\eta_{31} r_{21} r'_{21} + \eta_{32} r_{22} r'_{22} + \eta_{33} r_{21} r'_{22} + \eta_{34} r_{22} r'_{21}) \\ \nu_0 x_2 x'_2 (\eta_{41} r_{21} r'_{21} + \eta_{42} r_{22} r'_{22} + \eta_{43} r_{21} r'_{22} + \eta_{44} r_{22} r'_{21}) + \nu_2 (\eta_{41} r_{21} r'_{21} + \eta_{42} r_{22} r'_{22} + \eta_{43} r_{21} r'_{22} + \eta_{44} r_{22} r'_{21}) \\ \nu_0 (\eta_{41} r_{21} r'_{21} + \eta_{42} r_{22} r'_{22} + \eta_{43} r_{21} r'_{22} + \eta_{44} r_{22} r'_{21}) \end{pmatrix}$$

with $\nu_i = \nu_i(\mathbf{c}, \mathbf{c}')$ and $\eta_{ij} = \eta_{ij}(\mathbf{c}, \mathbf{c}')$. It follows that

$$\mathbf{c}'' = \nu_0 S^{-1} \begin{pmatrix} (\eta_{11} r_{11} r'_{11} + \eta_{12} r_{12} r'_{12} + \eta_{13} r_{11} r'_{12} + \eta_{14} r_{12} r'_{11})(x_1 x'_1 + \nu_1/\nu_0) \\ (\eta_{11} r_{11} r'_{11} + \eta_{12} r_{12} r'_{12} + \eta_{13} r_{11} r'_{12} + \eta_{14} r_{12} r'_{11}) \\ (\eta_{21} r_{11} r'_{11} + \eta_{22} r_{12} r'_{12} + \eta_{23} r_{11} r'_{12} + \eta_{24} r_{12} r'_{11})(x_1 x'_1 + \nu_1/\nu_0) \\ (\eta_{21} r_{11} r'_{11} + \eta_{22} r_{12} r'_{12} + \eta_{23} r_{11} r'_{12} + \eta_{24} r_{12} r'_{11}) \\ (\eta_{31} r_{21} r'_{21} + \eta_{32} r_{22} r'_{22} + \eta_{33} r_{21} r'_{22} + \eta_{34} r_{22} r'_{21})(x_2 x'_2 - \nu_1/\nu_0) \\ (\eta_{31} r_{21} r'_{21} + \eta_{32} r_{22} r'_{22} + \eta_{33} r_{21} r'_{22} + \eta_{34} r_{22} r'_{21}) \\ (\eta_{41} r_{21} r'_{21} + \eta_{42} r_{22} r'_{22} + \eta_{43} r_{21} r'_{22} + \eta_{44} r_{22} r'_{21})(x_2 x'_2 - \nu_1/\nu_0) \\ (\eta_{41} r_{21} r'_{21} + \eta_{42} r_{22} r'_{22} + \eta_{43} r_{21} r'_{22} + \eta_{44} r_{22} r'_{21}) \end{pmatrix}$$

It follows that \mathbf{O}_1 is valid, i.e. \mathbf{c}'' is a valid encryption of $\pi_1 = x_1 x'_1 + x_2 x'_2$. The running time of this operator is polynomial as long as κ, τ are polynomials and the polynomials ν_i, η_{ij} are polynomial-size. We propose to choose these polynomials in the form

$$\sum_{0 < i, j, k, \ell \leq 2\kappa; 0 \leq e, e' \leq d} a_{ijkl e, e'} U_i^e U_j^{d-e} V_k^{e'} V_\ell^{d-e'}$$

where d is a parameter indexed by λ . Such polynomials are polynomial-size as long as κ, d are polynomial-size, i.e. they have $O(\kappa^4 d^2)$ monomials. It follows that the running time of homomorphic operators is polynomial as long as κ, τ, d are polynomials in λ .

We did not identify any efficient algebraic attack, i.e. we did not obtain any relevant results (in less than $36h$) for any $(\kappa, \tau, d) \geq (2, 2, 1)$. More precisely, we have mainly experimented attacks that are irrelevant under the factoring assumption according to Proposition 7. In particular, we propose to partially recover θ :

- coefficients s_{ij} of S
- the polynomials ν_i, η_{ij} involved in the operators, in particular ν_0 .

Claim 2. *Algebraic attacks based on Gr ebner bases are not efficient provided $\kappa, \tau, d = \Theta(\lambda)$.*