Improving Additive and Multiplicative Homomorphic Encryption Schemes Based on Worst-Case Hardness Assumptions

Carlos Aguilar Melchor[1], Slim Bettaieb[1], Philippe Gaborit[1], and Javier Herranz[2]

[1] XLIM-DMI, Université de Limoges,
123, av. Albert Thomas
87060 Limoges Cedex, France
{carlos.aguilar,slim.bettaieb,philippe.gaborit}@xlim.fr
[2] Dept. Matemàtica Aplicada IV,
Universitat Politècnica de Catalunya,
C/ Jordi Girona, 1-3, 08034 Barcelona, Spain
jherranz@ma4.upc.edu

Abstract. In CRYPTO 2010, Aguilar et al. proposed a somewhat homomorphic encryption scheme, i.e. an encryption scheme allowing to compute a limited amount of sums and products over encrypted data, with a security reduction from LWE over general lattices. General lattices (as opposed to ideal lattices) do not have an inherent multiplicative structure but, using a tensorial product, Aguilar et al. managed to obtain a scheme allowing to compute products with a polylogarithmic amount of operands. In this paper we present an alternative construction allowing to compute products with polynomially-many operands while preserving the security reductions of the initial scheme. Unfortunately, despite this improvement our construction seems to be incompatible with Gentry's seminal transformation allowing to obtain fully-homomorphic encryption schemes.
Recently, Brakerski et al. used the tensorial product approach introduced by Aguilar et al. in a new alternative way which allows to radically improve the performance of the obtained scheme. Based on this approach, and using two nice optimizations, their scheme is able to evaluate products with exponentially-many operands and can be transformed into an efficient fully-homomorphic encryption scheme while being based on general lattice problems. However, even if these results outperform the construction presented here, we believe the modifications we suggest for Aguilar et al.'s schemes are of independent interest.

Keywords: homomorphic encryption, secure function evaluation, lattices.

1 Introduction

The problem of Homomorphic Encryption arises in many practical applications where privacy is required for some functionality, including cloud computing, private information retrieval, (private) search on encrypted/streaming data, etc. The input for a server, Alice, is some function f, whereas the input of a client, Bob, is some value x in the domain of f. This problem involves three steps.

1. Bob uses some randomized algorithm Enc to produce and send to Alice an *encrypted* version of x, $c_1 \leftarrow \mathsf{Enc}(pk, x)$.
2. Alice applies some evaluation algorithm Eval to her input f and the value c_1, to produce $c_2 \leftarrow \mathsf{Eval}(f, c_1)$, which is sent back to Bob.
3. Finally, Bob uses a decryption algorithm $\mathsf{Dec}(sk, c_2)$ to extract from c_2 the reply's value y.

The paradigm works correctly if $y = f(x)$ for any value of x in the domain of f, any keypair (pk, sk) and any ciphertext $c_1 \in \mathsf{Enc}(pk, x)$. The interest of a homomorphic encryption scheme is based on three criteria: the variety of functions for which correctness is ensured, the privacy obtained by Alice and Bob, and the compactness of the communications.

Considering privacy, there are two desirable properties. On one hand, if Alice does not obtain any information on x, then the protocol enjoys *privacy*. On the other hand, if Bob does not obtain any information on

f (other than what can be deduced from $f(x)$), then the protocol enjoys *function privacy*. If both conditions are respected the protocol enjoys *symmetric privacy*. Compactness is a more complex issue. If the size of Eval's output is independent of the size of the function evaluated $|f|$ we say the protocol is perfectly compact. However, for most of the schemes allowing to evaluate a wide variety of functions, the size of Eval depends on various parameters of the evaluated function. We say that such protocols are compact for a given family of functions if for these functions Eval's output is below $|f|$. If there is no function for which this is true, the scheme is said to be non-compact.

Without privacy (for Bob's input), it is trivial to build a protocol that ensures both function privacy and perfect compactness for any function f: Bob sends x to Alice, which replies with $f(x)$. It is also trivial to build a scheme that ensures perfect privacy for Bob's input if neither function privacy, nor compactness are required. Indeed, Bob can request the function f to Alice without sending any information about x, and compute directly $f(x)$. Thus we will only consider schemes that ensure privacy for Bob's input and either function privacy, or some degree of compactness (or both).

There are classic encryption schemes [24, 25, 9] with fundamental homomorphic properties (they are group homomorphisms between the plaintext and ciphertext space), which are by themselves perfectly-compact homomorphic encryption schemes for very specific families of functions such as multivariate monomials of arbitrary degree, or multivariate polynomials of degree one. The first approach allowing to evaluate any function was proposed by Yao in 1986 [30]: creating *garbled circuits* with a generic encryption scheme. This results in a homomorphic encryption scheme ensuring symmetric privacy and correctness for any function f, but which is non-compact, as Eval's output is linear in $|f|$. Obtaining schemes that provide some compactness for a larger set of functions than monomials or polynomials of degree one has been a long standing open issue. Two proposals provide generic constructions that adapt group homomorphic encryption schemes in order to deal with larger families of functions such as boolean circuits with logarithmic (in the security parameter) depth [26], or branching programs with polynomial (in the security parameter) length [16]. On the other hand, Boneh, Goh and Nissim presented a pairing based encryption scheme (not a generic construction) allowing to obtain perfectly-compact evaluations of multivariate polynomials with degree 2 [4], as long as the output size of the polynomial is logarithmic in the security parameter.

Between 2009 and 2011, lattice-based cryptography has provided a large set of interesting results on the homomorphic encryption field. Among these, there is of course the groundbreaking construction for fully-homomorphic encryption by Gentry [10]. This generic construction provides homomorphic encryption schemes that are perfectly-compact and symmetrically secure for any multivariate polynomial (hence the name, fully-homomorphic). However, the initial instantiation proposed by Gentry, as well as the first variants and instantiations of his scheme [29, 27, 11, 28, 13, 8] were based on relatively non-standard problems, among which the sparse (or low-weight) subset sum problem (SSSP). Very recently, in a work to appear in FOCS 2011 [12], Gentry and Halevi have managed to circumvent the sparse subset sum problem, which is a major step towards a completely standard proof of security. However, their construction, as all of the current fully homomorphic schemes, assumes that the underlying scheme is circularly secure. Obtaining a scheme such that this assumption can be proved is the the last open problem for a completely standard proof of security.

Building up a fully-homomorphic encryption scheme using Gentry's construction involves a step, bootstrapping, that has a very important impact on performance. Thus, obtaining schemes that allow to evaluate compactly polynomials without bootstrapping is an interesting line of research (even if this implies a bound on the polynomial degree), as we may hope to evaluate these polynomials more efficiently than using the costly fully-homomorphic schemes. In 2010, Gentry et al. proposed an alternative to [4] for the evaluation of degree 2 polynomials [14] (with arbitrarily large output) and Aguilar et al.'s proposed a generic construction [1] which allows to evaluate polynomials with a degree bounded above by a polylogarithmic function on the security parameter.

The different instantiations of Gentry's construction all provide means to evaluate polynomials of bounded degree. In particular, in [11] Gentry presents a variant of his somewhat homomorphic encryption scheme that is based on the quantum worst-case hardness of the shortest independent vector problem (SIVP) on ideal lattices over a given ring. When considering polynomials with many monomials, Gentry's somewhat homomorphic encryption scheme is able to provide a compact evaluation as long as the degree is bounded by

a polynomial on the security parameter. Aguilar et al.'s additive and d-operand multiplicative construction also provides means to do a compact evaluation of multivariate polynomials, but the bound of the degree with their construction is polylogarithmic in the security parameter (instead of polynomial as in Gentry's scheme). On the other hand, their construction's security relies on more standard security assumptions such as the quantum hardness of SIVP over integer lattices instead of ideal lattices.

In [11], working with ideal lattices restricts the hardness assumptions to this setting but, on the other hand, provides an inherent multiplicative structure that can be used when computing monomials of a given degree. This is not the case in integer lattices in which there is a need to overcome this lack of multiplicative structure. In order to do this, Aguilar et al. use a tensorial-like multiplication which results at first in an exponential ciphertext growth, and ultimately in the polylogarithmic limitation on the degree of the polynomials evaluated. Whereas it seems clear that there is a price to pay to build a multiplicative structure over an integer lattice, one may ask whether this price must be as steep as in Aguilar et al.'s construction.

In this paper, following the ideas presented in [1], we give an alternative way of chaining d encryption schemes with additively homomorphic properties. The resulting chained schemes are homomorphic encryption schemes able to evaluate degree d polynomials, with a communication cost which is polynomial in d (instead of exponential for [1]).

We also present in the Appendix (one detailed, two undetailed) examples of instantiations of the new generic construction, in which the security relies on the worst-case hardness of the Learning With Errors problem[3] (LWE) over integer lattices. Such instantiations allow to securely evaluate degree d polynomials as long as d is bounded above by a polynomial in the security parameter (instead of a polylogarithmic function for [1]). Moreover, it is possible to introduce some additional steps in the evaluation protocol that provide symmetric privacy, a feature that was not possible in [1]. The ciphertext size is logarithmic on the number of monomials of the evaluated polynomial, and, if the evaluated polynomial has degree d and the variables are binary, the ciphertext size is on $\tilde{O}(d^2)$. The protocol requires Bob to send $\tilde{O}(d^3)$ ciphertexts for the evaluation to be possible. Therefore, the total communication cost is $\tilde{O}(d^5)$. This is much better than in [1] where the cost is in $2^{O(d)}$.

Very recently, Brakerski et al. [5] (based on works of Brakerski and Vaikuntanathan [7, 6]) used the tensorial product approach introduced by Aguilar et al. in a new alternative way which allows to radically improve the performance of the obtained scheme. Based on this approach, and using two nice optimizations, their scheme is able to evaluate products with exponentially-many operands and can be transformed into an efficient fully-homomorphic encryption scheme while being based on general lattice problems. However, even if these results outperform the construction presented here, we believe the modifications we suggest for Aguilar et al.'s schemes are of independent interest.

2 Basic Idea

Notations. We denote scalars by roman letters (a, b, ...), vectors by bold letters ($\boldsymbol{a}$, $\boldsymbol{b}$, ...), and the associated coordinates with a parenthesized exponent ($\boldsymbol{b} = (b^{(1)}, \ldots, b^{(n)})$). Ciphertexts, which in this paper will always be vectors, are noted as greek bold letters ($\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, ...). To a keypair (pk, sk) of a given encryption scheme PKC we implicitly associate the instance of the scheme described by this keypair. We say that an instance (pk, sk) is an m-limited homomorphism via addition if the sum of up to m ciphertexts (possibly modulo a given integer) decrypts to the sum (possibly modulo another integer) of the corresponding plaintexts. Finally, for integer values $a < b$, we denote as $[a, b]$ the set $\{a, a+1, \ldots, b-1, b\}$.

A note on scalar operators. Let $\mathsf{PKC}_1 = (\mathsf{KeyGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ be an encryption scheme, and (pk_1, sk_1) an m-limited homomorphic instance such that the plaintext and ciphertext spaces are $\mathbb{Z}_{p_1}$ and $\mathbb{Z}_{p_2}^{n_2}$ with

[3] This problem can be reduced from the worst-case quantum hardness of SIVP over integer lattices (see [23]). Note that LWE has also a *classical* reduction from SIVP (see [21]), but this reduction requires a modulus exponential in the security parameter and does not work with our proposal *as is*. Obtaining an encryption scheme compatible with this reduction is beyond the scope of this paper.

$p_2 > p_1 m$. For any $a, b \in \mathbb{Z}_{p_1}$ such that $b < m$ and any $\boldsymbol{\alpha} \in \mathsf{Enc}_1(pk_1, a)$ we have

$$b\boldsymbol{\alpha} \bmod p_2 \text{ decrypts to } (b \bmod p_2)a \bmod p_1 = ab \bmod p_1$$

where $b\boldsymbol{\alpha} \bmod p_2$ means adding $\boldsymbol{\alpha}$ to itself b times over the integers, and replacing each coordinate with its residue modulo p_2. The assertion on the decryption is a direct consequence of the definition of an m-limited homomorphism, and the second equality comes from the fact that $p_1 m < p_2$ and thus $b < m$ implies $b < p_2$. This formulation may seem awkward, as we use b as an integer and then we apply a modular operation $\bmod p_2$ and a second modular operation $\bmod p_1$ without any relationship between p_1 and p_2. The intuition is that b is never seen as an element of $\mathbb{Z}_{p_2}$, but just used as a scalar in $\mathbb{Z}$ bounded above by p_1 (and also by p_2 as $p_1 < p_2$).

2.1 The AMGH Construction

The basic idea underlying the results that Aguilar et al. present in [1] is to apply several encryption layers to a given plaintext, in a process that they call *chaining*. If the encryption schemes used have a given set of properties, which is common in lattice based cryptography, it is possible to combine two ciphertexts into a new (much larger) ciphertext which can be seen as an encryption of the product of the associated plaintexts. It is possible to sum many of these large ciphertexts in order to evaluate a multivariate polynomial, and if there are enough monomials the evaluation will be compact.

Suppose that we have two encryption schemes such that: plaintexts are integers, ciphertexts are vectors of integers, the null vector is an encryption of zero, and it is possible to obtain instances which are m-limited homomorphisms via addition. Let $\mathsf{PKC}_1 = (\mathsf{KeyGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ and $\mathsf{PKC}_2 = (\mathsf{KeyGen}_2, \mathsf{Enc}_2, \mathsf{Dec}_2)$ be two cryptosystems with such properties. The idea is to start from encryptions $\boldsymbol{\alpha} \in \mathsf{Enc}_1(pk_1, a), \boldsymbol{\beta} \in \mathsf{Enc}_2(pk_2, b)$ of two bits $a, b \in \{0, 1\}$, and compute then a chained encryption of the product, $\boldsymbol{\gamma} \in \mathsf{Enc}((pk_1, pk_2), ab)$, being $\mathsf{Enc} = \mathsf{chain}(\mathsf{Enc}_1, \mathsf{Enc}_2)$ a new encryption function.

Suppose that ciphertexts of PKC_2 are vectors in $\mathbb{Z}^{n_2}$, so we have $\boldsymbol{\beta}^T = (\beta^{(1)}, \ldots, \beta^{(n_2)})$. If $(\alpha^{(1)}, \ldots, \alpha^{(t)})$ is the bit-representation of the ciphertext $\alpha \in \mathsf{Enc}_1(pk_1, a)$, then the idea is to compute the product between vectors $\boldsymbol{\beta} \in \mathsf{Enc}_2(b) \in \mathbb{Z}^{n_2}$ and $(\alpha^{(1)}, \ldots, \alpha^{(t)})$. The resulting ciphertext is

$$\boldsymbol{\gamma} \;=\; \begin{pmatrix} \beta^{(1)} \\ \vdots \\ \beta^{(n_2)} \end{pmatrix} \cdot (\alpha^{(1)}, \ldots, \alpha^{(t)}) \;=\; \begin{pmatrix} \uparrow & \cdots & \uparrow \\ \alpha^{(1)}\boldsymbol{\beta} & \ldots & \alpha^{(t)}\boldsymbol{\beta} \\ \downarrow & \cdots & \downarrow \end{pmatrix}$$

Note that the columns of matrix $\boldsymbol{\gamma}$ are either $\boldsymbol{\beta}$ or a column with zeros, depending on the value of each bit of $\boldsymbol{\alpha}$. Given such a ciphertext $\boldsymbol{\gamma}$, the decryption algorithm Dec_2 is applied to each column, which results in the vector of bits $(\alpha^{(1)}b, \ldots, \alpha^{(t)}b)$. Finally, an element in the ciphertext space of PKC_1 is reconstructed from this vector of bits, and algorithm Dec_1 is applied to it. If $b = 0$, then the obtained plaintext is $0 = ab$. If $b = 1$, the obtained plaintext is $a = ab$. Summing up, $\boldsymbol{\gamma}$ is decrypted, through this chained procedure, to the product ab. As the ciphertexts obtained through this product operation are sets of ciphertexts of Enc_2, which is an m-limited homomorphism, it is possible to add up many such products and the result will decrypt to the evaluation of a multivariate polynomial of degree 2 if m is large enough.

This method generalizes to compute an encryption of a product $a_1 \cdots a_d$ of d bits, by chaining d instances of such encryption schemes. However, this chaining procedure (i.e. expanding one of the initial ciphertexts in bits) necessarily leads to ciphertexts whose size is exponential with respect to d. The reason for this is that we encode $\boldsymbol{\alpha}$ very inefficiently with a binary representation $\{0, \boldsymbol{\beta}\}$. The idea of this paper is that much more efficient encodings may be used. Indeed, we can encode more than one bit at a time using a larger alphabet $\{0, \boldsymbol{\beta}, \ldots, (2^\ell - 1)\boldsymbol{\beta}\}$ if Enc_2 has the correct properties, or even encode the entire ciphertext $\boldsymbol{\alpha}$ into a single ciphertext of Enc_2 as shown in the following section.

2.2 Informal Description of the New Approach

Let n_2, n_3, p_1, p_2, p_3 be five positive integers, and (pk_1, sk_1), (pk_2, sk_2) two keypairs of two encryption schemes $\mathsf{PKC}_1 = (\mathsf{KeyGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ and $\mathsf{PKC}_2 = (\mathsf{KeyGen}_2, \mathsf{Enc}_2, \mathsf{Dec}_2)$ such that:

- the plaintext and ciphertext spaces associated to (pk_1, sk_1) are $\mathbb{Z}_{p_1}$ and $\mathbb{Z}_{p_2}^{n_2}$
- the plaintext and ciphertext spaces associated to (pk_2, sk_2) are $\mathbb{Z}_{p_2}^{n_2}$ and $\mathbb{Z}_{p_3}^{n_3}$

Moreover, suppose that for a given integer $M \geq 1$, (pk_1, sk_1) is a p_1M-limited homomorphism via addition modulo p_2 and (pk_2, sk_2) is a p_2n_2M-limited homomorphism via addition modulo p_3. As we use lattice-based encryption schemes[4] we will therefore have $p_2 > p_1 \cdot (p_1M)$ and $p_3 > p_2 \cdot (p_2n_2M)$. Thus, both of the instances described above will have an expansion factor bounded below by $2n_2$ and $2(n_3/n_2)$.

The idea is that when computing a product, we may use the plaintext space associated to (pk_2, sk_2) to encode the ciphertexts associated to (pk_1, sk_1) more efficiently than with the bit-by-bit approach proposed in the AMGH construction, as both spaces have the same structure. In order to do that, to the first multiplicand $a \in \mathbb{Z}_{p_1}$ we will associate a ciphertext using Enc_1, and to the second multiplicand $b \in \mathbb{Z}_{p_1}$ we will associate a set of ciphertexts using Enc_2, one for each vector $b\boldsymbol{e}_i$ with $(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{n_2})$ the canonical basis of $\mathbb{Z}_{p_2}^{n_2}$. More formally, let $a, b \in \mathbb{Z}_{p_1}$ and

- $\boldsymbol{\alpha} \in \mathsf{Enc}_1(pk_1, a)$ with $\boldsymbol{\alpha}^T = (\alpha^{(1)}, \ldots, \alpha^{(n_2)})$ and $\alpha^{(i)} \in \mathbb{Z}_{p_2}$;
- $(\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{n_2}) \in \mathsf{Enc}_2(pk_2, b \cdot \boldsymbol{e}_1) \times \cdots \times \mathsf{Enc}_2(pk_2, b \cdot \boldsymbol{e}_{n_2})$.

As noted in the beginning of this section, the fact that b belongs to $\mathbb{Z}_{p_1}$ and is encrypted with Enc_2 which has as plaintext space $\mathbb{Z}_{p_2}^{n_2}$ may seem strange as we generally will not have $p_1 \mid p_2$. Again, b is just a small scalar $b < p_1 < p_2$ (and thus such that $b \bmod p_2 = b$) for the basis vectors $\boldsymbol{e}_i$ which *are* the plaintexts. The idea of this construction is that the matrix-vector product

$$(\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{n_2})\ \boldsymbol{\alpha} = \begin{pmatrix} \beta_1^{(1)} & \ldots & \beta_{n_2}^{(1)} \\ \vdots & \ldots & \vdots \\ \beta_1^{(n_3)} & \ldots & \beta_{n_2}^{(n_3)} \end{pmatrix} \begin{pmatrix} \alpha^{(1)} \\ \vdots \\ \alpha^{(n_2)} \end{pmatrix} = \sum_i \alpha^{(i)} \boldsymbol{\beta}_i$$

decrypts to $ab (\bmod p_1)$. Indeed, $\sum_i \alpha^{(i)} \boldsymbol{\beta}_i \pmod{p_3}$ decrypts to $b\boldsymbol{\alpha} \pmod{p_2}$ through Dec_2 as $b\boldsymbol{\alpha} = \sum_i \alpha^{(i)} b\boldsymbol{e}_i$, and $b\boldsymbol{\alpha} \pmod{p_2}$ decrypts to $ab \pmod{p_1}$. Finally, as (pk_1, sk_1) and (pk_2, sk_2) are respectively p_1M-limited and p_2n_2M-limited homomorphisms, up to M of such encrypted products can be added up to obtain a sum of ciphertexts (modulo p_3) that decrypts to the evaluation of a degree 2 polynomial (with up to M monomials).

Of course, the product computation can be iterated to compute d operand multiplications, if we have enough instances that *chain* their respective plaintext and ciphertext spaces and are K-limited homomorphisms for K large enough.

2.3 Expansion Factor

Note that unlike in the construction proposed in [1], in this construction there is no constraint on the size of the final ciphertext, beyond the fact that the global expansion factor is the product of the expansion factors of the used schemes. In fact, this construction can be instantiated with schemes such as [2, 14] for which $(n_{i+1}/n_i) \cdot (\log p_{i+1}/\log p_i)$ is (roughly) a small constant $k > 2$.

The fact that k is bounded away from 1 is a major issue as it prevents us from having an expansion factor linear in d. Indeed, by chaining instances with shrinking expansion factors, $(2, 3/2, ..., (d-1)/(d-2), d/(d-1))$, the product of the expansion factors of d chained schemes would be d. In the next section we see how to modify this idea to solve this issue.

[4] With current lattice-based encryption schemes, ciphertexts have an underlying error which expands at each homomorphic operation, and beyond a given amount of operations the error may increase to a point of non-return, breaking decryption correctness. In order to avoid that, we choose a ciphertext space large enough to avoid errors from overflowing into data, which results in a constraint such as $p_{i+1} > p_i \cdot m$, m being a bound on the amount of homomorphic operations.

2.4 Reducing the Number of Homomorphic Operations

First, suppose that n_2/n_1 can be set asymptotically close to 1. As noted before, the issue is that in lattice based schemes, the number of homomorphic operations that can be done has an impact on ciphertext size (to ensure decryption correctness despite the growth of the underlying error). For example, in [14], with some modifications (namely, dropping the "multiplication for free" property), it is possible to have $p_2 \simeq p_1 \cdot m \cdot \mathsf{poly}(\kappa)$, κ being the dimension of the underlying lattice. However, our construction requiring the instance of PKC_1 to be at least a p_1-limited homomorphism, we have $p_2 = O(p_1^2)$ and thus an expansion factor bounded below by 2.

Thus, the idea is to reduce the number of sums done to absorb the scalars that appear in the components of ciphertexts of PKC_1 (such scalars live in $\mathbb{Z}_{p_2}$). In fact, this is pretty easy to do! Indeed, in order to absorb a coordinate $\alpha^{(i)}$ of $\boldsymbol{\alpha}$ into an encryption of $b\boldsymbol{e}_i$, we must do p_2 homomorphic operations (possibly through a double-and-add protocol). But, if we have a fresh encryption of $2^j b\boldsymbol{e}_i$ for each $j \in [0, \lceil \log p_2 \rceil - 1]$ we just need to add the encryptions corresponding to the bits of $\alpha^{(i)}$ which are non null, and therefore we just need to do $\lceil \log p_2 \rceil$ homomorphic operations!

Of course this allows us to have $p_3 = p_2 \lceil \log p_2 \rceil \cdot \mathsf{poly}(\kappa)$ and thus an (almost) linear growth of the ciphertext sizes on d as we iteratively define $p_1, p_2, p_3, \ldots$

3 Fully Chainable Schemes

3.1 Definition and Instantiations

In this subsection we define the notion of *fully chainable scheme*, which gathers the required properties to create ciphertext chains with our construction. The notion of chainable (instead of *fully*-chainable) scheme, described in [1], requires the plaintext space to be a subset of $\mathbb{Z}$ (in practice $\mathbb{Z}_p$ for some p), which is incompatible with our construction. Fully chainable schemes must be able to provide instances with plaintext space $\mathbb{Z}_p^n$ for any positive integers p, n. This could be seen as an extra constraint, possibly limiting the number of schemes fulfilling the requisites. In practice, most schemes, including the instantiations of [1] based on [14, 2] already provide such a property, and in order to fit into the definition of chainable schemes the plaintext space was artificially reduced to $\mathbb{Z}_p$ by leaving unused some of the plaintext coordinates.

Definition 1. *A scheme* $\mathsf{PKC} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is said* (f_n, f_p)*-fully-chainable for two integer functions* f_n, f_p *if:*

- *The key generation algorithm* KeyGen *takes as input a security parameter* κ *and three positive integers* m, n, p*;*
- *The integers* $n' = f_n(\kappa, m, n, p), p' = f_p(\kappa, m, n, p)$ *are defined for any set of parameters of* KeyGen*;*
- *For any keypair* $(pk, sk) \in \mathsf{KeyGen}(1^\kappa, m, n, p)$ *the following holds:*
  - *The plaintext space is* $\mathbb{Z}_p^n$*;*
  - *The ciphertext space is a subset of* $\mathbb{Z}_{p'}^{n'}$ *and* $\mathbf{0}^{n'} \in \mathsf{Enc}(pk, \mathbf{0}^n)$*;*
  - m*-limited homomorphism via addition modulo* p'*: for any* $\ell \leq m$*,* $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_\ell \in \mathbb{Z}_p^n$ *and any* $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_\ell$ *with* $\boldsymbol{\alpha}_i \in \mathsf{Enc}(pk, \boldsymbol{a}_i)$*, the vector* $\boldsymbol{\alpha} = \sum_i \boldsymbol{\alpha}_i \bmod p'$ *is decrypted via* Dec *to the integer vector* $\boldsymbol{a} = \sum_i \boldsymbol{a}_i \bmod p$*.*

The ciphertext space associated to an instance (pk, sk) of a fully-chainable scheme is a subset of $\mathbb{Z}_{p'}^{n'}$ for given p', n' but, in order to simplify, we will sometimes just say that $\mathbb{Z}_{p'}^{n'}$ is the ciphertext space. For $\boldsymbol{a} \in \mathbb{Z}_p^n$ and a strictly positive integer $\ell \in \mathbb{Z}^+$ we denote as $\mathsf{Enc}(pk, \boldsymbol{a})^{+\ell}$ the subset of $\mathbb{Z}_{p'}^{n'}$ corresponding to the sums, modulo p', of *exactly* ℓ ciphertexts whose plaintexts sum up to $\boldsymbol{a}$ modulo p. As a result of the m-limited homomorphism property, any $\boldsymbol{\alpha} \in \mathsf{Enc}(pk, \boldsymbol{a})^{+\ell}$ decrypts to $\boldsymbol{a}$ as long as $\ell \leq m$.

Theorem 1. *There is an* (f_n, f_p)*-fully-chainable scheme* $\mathsf{PKC} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *with:*

- $f_n(\kappa, m, n, p) = n + \kappa$

– $f_p(\kappa, m, n, p) \in [6\kappa^2 mp \cdot \log(\kappa mp), 12\kappa^2 mp \cdot \log(\kappa mp)]$

Suppose that parameters m, n, p are such that n is bounded above by a polynomial in κ and m, p are arbitrary functions of κ. A quantum attacker able to break the indistinguishability of this scheme is able to approximate the decision version of the shortest vector problem (GapSVP*) and the shortest independent vectors problem (*SIVP*) to within $\tilde{O}(\kappa^{5/2}mp)$ in the worst case.*

Proof. Direct consequence of Propositions 4 and 5 in Appendix A, using m' slightly above $2\kappa m \log q$ as suggested in the public key realization section (Appendix A.5).

In Appendix A, besides the intermediate results that lead to this theorem, the reader can find other alternatives and the details of how to instantiate our construction with a very simple scheme, with security reduced from LWE (and thus from quantum SIVP or quantum GapSVP).

3.2 Chaining Schemes

In this subsection we describe an algorithm that chains two encryption schemes $\mathsf{PKC}_1, \mathsf{PKC}_2$ that are respectively (f_n, f_p)-fully-chainable and (g_n, g_p)-fully-chainable, into a (h_n, h_p)-fully-chainable scheme PKC (for given h_n, h_p). The instances of this scheme are just the composition of an instance of PKC_1 and an instance of PKC_2 such that the ciphertext space of the first instance is (roughly) the plaintext space of the second.

Unlike in [1], a ciphertext of an instance $((pk_1, pk_2), (sk_1, sk_2))$ of PKC is a ciphertext of the instance (pk_2, sk_2) of PKC_2. Of course the ciphertext/plaintext expansion factor is larger for the instance of PKC as it is the product of the expansion factors of (pk_1, sk_1) and (pk_2, sk_2) which are both strictly larger than 1 (as the schemes are randomized). However, if both expansion factors are close to 1 the product will be close to 1 too, and thus the resulting expansion factor can be comparable to the ones of the instances chained.

Chaining Algorithm: $\mathsf{PKC} = \mathsf{chain}(\mathsf{PKC}_1, \mathsf{PKC}_2)$

Input:
- An (f_n, f_p)-fully-chainable scheme $\mathsf{PKC}_1 = (\mathsf{KeyGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$
- An (g_n, g_p)-fully-chainable scheme $\mathsf{PKC}_2 = (\mathsf{KeyGen}_2, \mathsf{Enc}_2, \mathsf{Dec}_2)$

Output:
- An (h_n, h_p)-fully-chainable scheme $\mathsf{PKC} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with $(h_n, h_p)(\kappa, m, n, p) = (g_n, g_p)(\kappa, m, f_n(\kappa, m, n, p), f_p(\kappa, m, n, p))$.

Return a description the encryption scheme PKC:

$\mathsf{KeyGen}(1^\kappa, m, n_1, p_1)$:
1 Set $(pk_1, sk_1) \leftarrow \mathsf{KeyGen}_1(1^\kappa, m, n_1, p_1)$
2 Define $n_2 = f_n(\kappa, m, n_1, p_1)$ and $p_2 = f_p(\kappa, m, n_1, p_1)$
3 Set $(pk_2, sk_2) \leftarrow \mathsf{KeyGen}_2(1^\kappa, m, n_2, p_2)$
4 Return $((pk_1, pk_2), (sk_1, sk_2))$

$\mathsf{Enc}((pk_1, pk_2), \boldsymbol{a} \in \mathbb{Z}_{p_1}^{n_1})$:
1 Return $\boldsymbol{\beta} \leftarrow \mathsf{Enc}_2(pk_2, \boldsymbol{\alpha})$ with $\boldsymbol{\alpha} \leftarrow \mathsf{Enc}_1(pk_1, \boldsymbol{a})$

$\mathsf{Dec}((sk_1, sk_2), \boldsymbol{\beta} \in \mathbb{Z}_{p_3}^{n_3})$ with $(n_3, p_3) = (g_n, g_p)(\kappa, m, n_2, p_2)$:
1 Return $\boldsymbol{a} \leftarrow \mathsf{Dec}_1(sk_1, \boldsymbol{\alpha})$ with $\boldsymbol{\alpha} \leftarrow \mathsf{Dec}_2(sk_2, \boldsymbol{\beta})$

Proposition 1. PKC *is (h_n, h_p)-chainable, with*

$$(h_n, h_p)(\kappa, m, n, p) = (g_n, g_p)(\kappa, m, f_n(\kappa, m, n, p), f_p(\kappa, m, n, p))$$

Proof. Trivial.

An interesting question is how the chaining procedure impacts on ciphertext size. If PKC_2 is the scheme of Theorem 1, we have:

- $g_n(\kappa, m, n, p) = n + \kappa$
- $g_p(\kappa, m, n, p) \in [6\kappa^2 mp \cdot \log(\kappa mp), 12\kappa^2 mp \cdot \log(\kappa mp)]$

and therefore, noting $(\boldsymbol{z}) = (\kappa, m, n, p)$ for clarity, it can be easily verified that in this case:

- $h_n(\boldsymbol{z}) = f_n(\boldsymbol{z}) + \kappa$
- $f_p(\boldsymbol{z}) < h_p(\boldsymbol{z}) \leq f_p(\boldsymbol{z})\log(f_p(\boldsymbol{z})) \cdot 12\kappa^2 m\log(\kappa m)$

In other words, if PKC_2 is the scheme of Theorem 1, the growth in the number of coordinates when passing from PKC_1 to PKC is independent of PKC_1, and the number of bits by coordinate added, $\log(h_p(\boldsymbol{z})) - \log(f_p(\boldsymbol{z}))$, seems to be a very slowly growing parameter (it is double-logarithmic in $f_p(\boldsymbol{z})$ and logarithmic in κ and m). We can thus hope that chaining the scheme of Theorem 1 to itself several times will result in a reasonable ciphertext growth (as it will be shown in Section 4.3).

Let us prove now that the chained scheme PKC resulting from PKC_1 and PKC_2 is IND-CPA secure if either of PKC_1, PKC_2 is IND-CPA secure (see Appendix A for a formal definition). The proof is, in fact, just a simplification of the one of Proposition 2 in [1].

Proposition 2 (IND-CPA Security). $\mathsf{PKC} = \mathsf{chain}(\mathsf{PKC}_1, \mathsf{PKC}_2)$ *is IND-CPA secure if either of* PKC_1, PKC_2 *is IND-CPA secure.*

Proof. Let us assume that there exists a CPA attacker $\mathcal{A}$ against PKC and let us prove, then, that neither of PKC_1, PKC_2 can be IND-CPA. Specifically, we can construct CPA attackers $\mathcal{A}_1, \mathcal{A}_2$ against the schemes PKC_1 and PKC_2.

For PKC_1, the attacker $\mathcal{A}_1$ is trivial as a random keypair of PKC_1 can be transformed in a random keypair of PKC by adding a random keypair of PKC_2. Moreover, the choice of the two plaintexts by $\mathcal{A}$ is maintained by $\mathcal{A}_1$, and the challenge ciphertext received from Enc_1 can be transformed into a challenge ciphertext for Enc just by encrypting it through Enc_2. Finally, as the plaintexts are the same, Attacker $\mathcal{A}_1$ will output the same guess as $\mathcal{A}$ will, and the success probability of both attackers will be exactly the same.

For PKC_2, the idea is very similar. Again, an attacker $\mathcal{A}_2$ can transform a random keypair of PKC_2 into a random keypair of PKC by adding a random keypair of PKC_1. If $\mathcal{A}$ chooses two plaintexts $\boldsymbol{a}_0, \boldsymbol{a}_1$, attacker $\mathcal{A}_2$ chooses for his attack the plaintexts $\boldsymbol{\alpha}_0 \leftarrow \mathsf{Enc}_1(pk_1, \boldsymbol{a}_0)$ and $\boldsymbol{\alpha}_1 \leftarrow \mathsf{Enc}_1(pk_1, \boldsymbol{a}_1)$. The challenge ciphertext received by $\mathcal{A}_2$ is a valid challenge ciphertext for $\mathcal{A}$. The answer (guess) of $\mathcal{A}$ is taken by $\mathcal{A}_2$ as his answer to his game. The success probability of both attackers is again the same. □

4 Computing with Chained Schemes

4.1 Products

In this section we show how to compute an encryption of the value $a_1 a_2 \bmod p_1$, under $\mathsf{PKC} = \mathsf{chain}(\mathsf{PKC}_1, \mathsf{PKC}_2)$, starting from an encryption of $a_1 \in \mathbb{Z}_{p_1}$ under PKC_1 and $a_2 \in \mathbb{Z}_{p_1}$ under PKC_2. First we show how to deal with the case $a_1, a_2 \in \{0, 1\} \subset \mathbb{Z}_{p_1}$ for any $p_1 \geq 2$, and then we describe how to modify our protocol to take into account larger plaintexts.

Computing with binary values. If $a_1, a_2 \in \{0, 1\} \subset \mathbb{Z}_{p_1}$ we can compute an encryption of the value $a_1 a_2 \bmod p_1$, under $\mathsf{PKC} = \mathsf{chain}(\mathsf{PKC}_1, \mathsf{PKC}_2)$, starting from an encryption of $a_1 \in \mathbb{Z}_{p_1}$ under PKC_1 and multiple encryptions of $a_2 \in \mathbb{Z}_{p_1}$ under PKC_2. Note that in many applications such as computationally-Private Information Retrieval [18] (cPIR) or private searching [20], a polynomial is evaluated over encrypted data on $\mathbb{Z}_{p_1}$ for large values of p_1 (in order to retrieve information with small expansion factors), but the

variables are always binary (namely 0 for the elements we are not interested in, and 1 for the elements we want to retrieve/search). Such a setting is therefore not just of theoretical interest, and this basic protocol is enough for many applications.

Without loss of generality, we assume that the plaintext space of the first encryption scheme PKC_1 in the chain is $\mathbb{Z}_{p_1}$ (i.e. $n_1 = 1$), by encoding if necessary a value $a \in \mathbb{Z}_{p_1}$ as a vector $\boldsymbol{a} = (a, 0, \ldots, 0) \in \mathbb{Z}_{p_1}^{n_1}$. The output of the product algorithm will be a sum of ciphertexts; thus, in order to be able to use the product algorithm recursively (to compute products of more than two elements), we suppose that one of the inputs of the algorithm is already a sum of (up to) ℓ_1 ciphertexts. We determine the correctness of the result (i.e. whether it decrypts to the product of the plaintexts) based on ℓ_1 and on the homomorphic capacity of the used instances.

Product Computation Algorithm: $\boldsymbol{\gamma} = \mathsf{product}(\boldsymbol{\alpha}, \boldsymbol{\beta}_{i,j})$

Setting:
- PKC_1 (f_n, f_p)-fully-chainable scheme
- PKC_2 (g_n, g_p)-fully-chainable scheme
- $\mathsf{PKC} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}) = \mathsf{chain}(\mathsf{PKC}_1, \mathsf{PKC}_2)$
- $((pk_1, pk_2), (sk_1, sk_2)) \leftarrow \mathsf{KeyGen}(1^\kappa, m, 1, p_1)$
- $\mathbb{Z}_{p_1}$, $\mathbb{Z}_{p_2}^{n_2}$ resp. plaintext and ciphertext spaces of (pk_1, sk_1)
- $\mathbb{Z}_{p_2}^{n_2}$, $\mathbb{Z}_{p_3}^{n_3}$ resp. plaintext and ciphertext spaces of (pk_2, sk_2)

Input:
- $\boldsymbol{\alpha} \in \mathsf{Enc}_1(pk_1, a_1)^{+\ell_1}$, where $a_1 \in \{0,1\} \subset \mathbb{Z}_{p_1}$
- $\boldsymbol{\beta}_{i,j} \in \mathsf{Enc}_2(pk_2, 2^j a_2 \boldsymbol{e}_i)$ for $(i,j) \in [1, n_2] \times [0, \lceil \log p_2 \rceil - 1]$, where $a_2 \in \{0,1\} \subset \mathbb{Z}_{p_1}$

Output:
- $\boldsymbol{\gamma} \in \mathsf{Enc}((pk_1, pk_2), a_1 a_2 \bmod p_1)^{+\ell_2}$ for $\ell_2 = \max\{\ell_1, n_2 \lceil \log p_2 \rceil\}$

1 For each coordinate $\alpha^{(i)} \in \mathbb{Z}_{p_2}$ of $\boldsymbol{\alpha} = (\alpha^{(1)}, \ldots, \alpha^{(n_2)})$ compute

$$\boldsymbol{\gamma}_i = \sum_{j=0}^{\lceil \log p_2 \rceil - 1} \alpha^{(i,j)} \boldsymbol{\beta}_{i,j} \bmod p_3, \text{ noting } \alpha^{(i)} = \sum_{j=0}^{\lceil \log p_2 \rceil - 1} 2^j \alpha^{(i,j)}$$

2 Return $\boldsymbol{\gamma} = \sum_{i=1}^{n_2} \boldsymbol{\gamma}_i \bmod p_3$

The proposition below guarantees that the output of the protocol decrypts to the product of plaintexts if $m \geq \ell_2$. The main idea of the proof is to show that the output, as asserted in the protocol, verifies

$$\boldsymbol{\gamma} \in \mathsf{Enc}((pk_1, pk_2), a_1 a_2 \bmod p_1)^{+\ell_2} \text{ for } \ell_2 = \max\{\ell_1, n_2 \lceil \log p_2 \rceil\}$$

and then the proposition becomes trivial. In order to obtain this proof, we must suppose that the encryption schemes satisfy the following property:

$$\text{For all integers } \ell \leq \ell' \text{ and } i \in \{1,2\},\ \mathsf{Enc}_i(pk, a)^{+\ell} \subset \mathsf{Enc}_i(pk, a)^{+\ell'} \qquad (*)$$

All the instantiations proposed in Appendix A satisfy the above-mentioned property.

Proposition 3. *If $m \geq \ell_2$, the output of the above-described protocol* $\mathsf{product}$ *decrypts through* $\mathsf{Dec}((sk_1, sk_2), \boldsymbol{\gamma})$ *to* $a_1 a_2 \bmod p_1$.

Proof. On the one hand, since $a_2 \in \{0,1\} \subset \mathbb{Z}_{p_1}$ and $\boldsymbol{\alpha} \in \mathsf{Enc}_1(pk_1, a_1)^{+\ell_1}$, and assuming that property $(*)$ is satisfied, we have

$$a_2 \boldsymbol{\alpha} \bmod p_2 \ \in\ \mathsf{Enc}_1(pk_1, a_1 a_2 \bmod p_1)^{+\ell_1} \ \subset\ \mathsf{Enc}_1(pk_1, a_1 a_2 \bmod p_1)^{+\ell_2} \qquad (1)$$

On the other hand, because of the definition of $\boldsymbol{\beta}_{i,j}$, $\alpha^{(i,j)}$, and $\alpha^{(i)}$, we have

$$\boldsymbol{\gamma} = \sum_{i=1}^{n_2} \gamma_i = \sum_{i=1}^{n_2} \sum_{j=0}^{\lceil \log p_2 \rceil - 1} \alpha^{(i,j)} \boldsymbol{\beta}_{i,j}$$

end thus we have $\gamma_i \in \mathsf{Enc}_2(pk_2, \alpha^{(i)} a_2 \boldsymbol{e}_i)^{+\lceil \log p_2 \rceil}$ and $\boldsymbol{\gamma} \in \mathsf{Enc}_2(pk_2, a_2 \boldsymbol{\alpha} \bmod p_2)^{+n_2 \lceil \log p_2 \rceil}$. Using property $(*)$, we obtain

$$\boldsymbol{\gamma} \ \in \ \mathsf{Enc}_2(pk_2, a_2 \boldsymbol{\alpha} \bmod p_2)^{+\ell_2} \tag{2}$$

Finally, the definition of $\mathsf{PKC} = \mathsf{chain}(\mathsf{PKC}_1, \mathsf{PKC}_2)$ ensures that

$$\mathsf{Enc}((pk_1, pk_2), a_1 a_2 \bmod p_1)^{+\ell_2} = \bigcup_{\tilde{\boldsymbol{\alpha}} \in \mathsf{Enc}_1(pk_1, a_1 a_2 \bmod p_1)^{+\ell_2}} \mathsf{Enc}_2(pk_2, \tilde{\boldsymbol{\alpha}})^{+\ell_2} \tag{3}$$

This is where we need the definition of $\mathsf{Enc}(pk, \boldsymbol{a})^{+\ell}$ as the sums of *exactly* ℓ ciphertexts whose plaintexts sum up to $\boldsymbol{a}$. If we had defined $\mathsf{Enc}(pk, \boldsymbol{a})^{+\ell}$ as the sums of up to ℓ ciphertexts, then Equation (3) would not hold. Joining Equations (1), (2) and (3), we get $\boldsymbol{\gamma} \in \mathsf{Enc}((pk_1, pk_2), a_1 a_2 \bmod p_1)^{+\ell_2}$, which decrypts to $a_1 a_2 \bmod p_1$ if $m \geq \ell_2$. □

Computing with arbitrary values. Note that if we have $a_1, a_2 \in \mathbb{Z}_{p_1}$ without the restriction $a_1, a_2 \in \{0, 1\}$, the product protocol works as long as we set $\ell_2 = \max\{\ell_1(p_1 - 1), n_2 \lceil \log p_2 \rceil\}$. Indeed, the only part of the proof that would change is

$$a_2 \boldsymbol{\alpha} \bmod p_2 \ \in \ \mathsf{Enc}_1(pk_1, a_1 a_2 \bmod p_1)^{+\ell_1(p_1 - 1)} \ \subset \ \mathsf{Enc}_1(pk_1, a_1 a_2 \bmod p_1)^{+\ell_2}$$

The main issue with such a protocol is that it results in $O(p_1)$ homomorphic operations. For a single product this is not a big problem, but when computing d products iteratively, as we will do in Section 4.2, we would need to do $O(p_1^d)$ homomorphic operations. It is possible to lower the number of homomorphic operations to $O(\lceil \log p_1 \rceil^d)$, which is below p_1 as long as $d \leq \log p_1 / (\log(\log p_1))$, by using a splitting technique such as the one used in Section 2.4, but the resulting protocol is complex and does not lower the amount of operations to a polynomial in d.

It is therefore possible to compute with arbitrary values, but in this case one of the contributions of our construction (providing means to have polynomial gaps) is lost. Of course with our current instantiations this is not much of a difference as they result nevertheless in gaps which are exponential in d for other reasons. However, we prefer to split the two cases apart; this allows us to highlight that, in the binary case, with a better instantiation we could obtain a scheme with polynomial gaps in d which is interesting from a security point of view.

4.2 Iterating Products, and Evaluating Polynomials

One can use a single (f_n, f_p)-fully-chainable encryption scheme PKC chained to itself iteratively to evaluate degree d polynomials. Indeed, if we set $\mathsf{PKC}_{1,1} = \mathsf{PKC}$ and $\mathsf{PKC}_{1,h} = \mathsf{chain}(\mathsf{PKC}_{1,h-1}, \mathsf{PKC})$ for $h \in [2, d]$, it is possible to compute an encryption of $a_1 \dots a_d \bmod p_1$, starting from $\boldsymbol{\gamma}_1 \in \mathsf{Enc}(pk_1, a_1)$ and $\boldsymbol{\beta}_{h,i,j} \in \mathsf{Enc}(pk_h, 2^j a_h \boldsymbol{e}_i)$, for $h \in [2, d]$, appropriate values of i, j, being $((pk_1, \dots, pk_d), (sk_1, \dots, sk_d))$ an instance of $\mathsf{PKC}_{1,d}$. Executing iteratively the $\mathsf{product}$ protocol we compute for $h \in [2, d]$,

$$\boldsymbol{\gamma}_h = \mathsf{product}(\boldsymbol{\gamma}_{h-1}, \boldsymbol{\beta}_{h,i,j}), \text{ with } \boldsymbol{\gamma}_h \in \mathsf{Enc}_{1,h}((pk_1, \dots, pk_h), a_1 \dots a_h \bmod p_1)^{\ell_h}$$

This can be done for different products, and the results, which are sums of ciphertexts of the fully-chainable scheme $\mathsf{PKC}_{1,d}$, can be added up to obtain an encrypted evaluation of a polynomial $P = \sum_{i=1}^{M} p_i X_{i_1} \dots X_{i_d} \in \mathbb{Z}_{p_1}[X_1, \dots, X_v]$ on v variables, with degree d and M monomials. We omit the details, because an almost identical (detailed) protocol can be found in [1]. Intuitively, if Alice holds this polynomial, and Bob wants

Alice to securely evaluate it on an input point $(a_1, \ldots, a_v) \in \{0,1\}^v \subset \mathbb{Z}_{p_1}^v$, Bob will encrypt the different plaintexts needed for the product protocols (i.e. the plaintexts $2^j a_h e_i$ for appropriate values of h, i, j) under the different public keys $pk_1, \ldots, pk_d$ associated to an instance of $\mathsf{PKC}_{1,d}$.

Since the encryptions that Bob sends to Alice are fresh (i.e. $\ell_1 = 1$), the reply that Alice sends back is in $\mathsf{Enc}((pk_1, \ldots, pk_d), P(a_1, \ldots, a_v) \bmod p_1)^{+Mp_1\ell_d}$, where ℓ_i is recursively defined as $\ell_1 = 1$ and $\ell_{h+1} = \max\{\ell_h, n_{h+1}\lceil \log p_{h+1} \rceil\}$. If the instance of $\mathsf{PKC}_{1,d}$ associated to $(pk_1, \ldots, pk_d)$ is an m-limited homomorphism with $m > M(p_1 - 1)\ell_d$, then Alice's reply decrypts to $P(a_1, \ldots, a_v) \bmod p_1$.

4.3 On the Size and Amount of Ciphertexts

In the protocol for the secure evaluation of a degree d polynomial, sketched in the previous section, the final ciphertext that Alice sends to Bob is a vector in the ciphertext space $\mathbb{Z}_{p_{d+1}}^{n_{d+1}}$ of $\mathsf{PKC}_{1,d}$. The size of this ciphertext is therefore $n_{d+1}\log p_{d+1}$ bits.

Note that these values n_{d+1}, p_{d+1} are determined by the full-chainability of the employed encryption schemes $\mathsf{PKC}_1, \ldots, \mathsf{PKC}_d$, in particular by the behavior of the functions f_n and f_p. If the scheme of Theorem 1 is chained to itself iteratively, all the schemes will have the same function:

- $f_n(\kappa, m, n, p) = n + \kappa$
- $f_p(\kappa, m, n, p) \in [6\kappa^2 mp \cdot \log(\kappa mp), 12\kappa^2 mp \cdot \log(\kappa mp)]$

Then we have $n_{i+1} = f_n(\kappa, m, n_i, p_i)$ and $p_{i+1} = f_p(\kappa, m, n_i, p_i)$, for $i = 1, \ldots, d$. Using the bounds on the growth of the number of coordinates and on the number of bits provided in Section 3.2 we obtain:

- $n_{i+1} = n_i + \kappa$ and $n_{i+1} > n_i$,
- $p_{i+1} \leq p_i \log(p_i) \cdot 12\kappa^2 m \log(\kappa m)$ and $p_{i+1} > p_i$.

The first relation trivially shows that $n_{d+1} = 1 + d\kappa$. Obtaining a tight bound for $\log p_{d+1}$ is a little trickier. However, it is possible to provide easily a gross approximation, using the monotony of n_i and p_i as i grows, we can say that

$$\begin{aligned} p_{d+1} &\leq p_1(\log(p_d) \cdot 12\kappa^2 m \log(\kappa m))^d \qquad \text{and thus,} \\ \log p_{d+1} &< \log p_1 + d \cdot F(d, \kappa, m) \end{aligned}$$

where $F(d, \kappa, m)$ is a function which depends just logarithmically on each of its inputs. Thus, as κ is a free parameter, we only need to show that m is bounded above by a polynomial in d to prove that $\log p_{d+1}$ is roughly linear in d. Indeed, when chaining d schemes in order to evaluate a polynomial of degree d with M monomials, as shown in the previous section, we must have

- $\ell_1 = 1$, $\ell_{h+1} = \max\{\ell_h, n_{h+1}\lceil \log p_{h+1} \rceil\}$,
- and $m > M(p_1 - 1)\ell_d$.

Based on the monotony of n_i and p_i we will have $\ell_d = n_d\lceil \log p_d \rceil$, and thus m is bounded above by a polynomial in d. As noted above this implies that F just depends logarithmically in d and thus that $\log p_{d+1}$ is in $\tilde{O}(d)$ and ciphertext size is in $\tilde{O}(d^2)$. Note that this size is relative to a ciphertext of $\mathsf{PKC}_{1,d}$ which corresponds to Alice's reply in an homomorphic encryption protocol. When using the evaluation protocol proposed in [1], in order to compute the different products, Alice needs to have many ciphertexts that will be sent by Bob. Indeed, for each variable, Bob sends a set of ciphertexts corresponding to each instance $\mathsf{PKC}_{1,i}$ in the chain, and each set contains $n_i\lceil \log p_i \rceil$ ciphertexts. In other words, Bob sends $\tilde{O}(d^3)$ ciphertexts and thus the total communication cost for the evaluation of the degree d polynomial is in $\tilde{O}(d^5)$.

5 Putting All the Pieces Together

The results of Sections 3 and 4 can be combined with the specific instantiations that we provide in Appendix A, so that we obtain the following result.

Theorem 2. *There exists a public key encryption scheme allowing to evaluate homomorphically polynomials with M monomials and degree d such that:*

- *The semantic security of the scheme relies on the quantum hardness to approximate the decision version of* GapSVP *or the quantum hardness of* SIVP *to within $\tilde{O}(\kappa^{d+5/2} \cdot M^d)$ in the worst case.*
- *The secure evaluation of a polynomial has a communication cost when the variables are binary in $\tilde{O}(d^5)$.*
- *By increasing the approximation factor by a sub-exponential function in κ it is possible to modify the scheme to provide formula privacy.*

Proof. (Sketch.) Let PKC be the fully-chainable encryption scheme from Theorem 1. We can chain d appropriate instantiations of this same scheme defining $\mathsf{PKC}_{1,1} = \mathsf{PKC}$ and $\mathsf{PKC}_{1,h} = \mathsf{chain}(\mathsf{PKC}_{1,h-1}, \mathsf{PKC})$ for $h \in [2, d]$. Each of the chained instantiations of PKC is associated to a key pair (sk_h, pk_h), a plaintext space $\mathbb{Z}_{p_h}^{n_h}$ and a ciphertext space $\mathbb{Z}_{p_{h+1}}^{n_{h+1}}$. According to Theorem 1, the semantic security of this particular h-th instantiation is based on the quantum hardness to approximate the decision version of GapSVP or the quantum hardness of SIVP to within $\tilde{O}(\kappa^{5/2} M p_h)$ in the worst case.

The public key encryption scheme whose existence we claim in this theorem is simply the cartesian product of the above d instantiations of PKC. That is, the secret key is $(sk_1, \ldots, sk_d)$ and the public key is $(pk_1, \ldots, pk_d)$. A "fresh" ciphertext for a plaintext $a \in \mathbb{Z}_{p_1}$ is the concatenation of all the ciphertexts $\mathsf{Enc}(pk_h, 2^j a \boldsymbol{e}_i)$, for $h \in [1, d]$ and $(i, j) \in [1, n_h] \times [0, \lceil \log p_2 \rceil - 1]$, where $\boldsymbol{e}_i$ is the i-th vector of the canonical basis of $\mathbb{Z}_{p_h}^{n_h}$. A formal definition of the decryption function is cumbersome as usual for secure evaluation schemes. Informally, the outcome of a computation will always be a ciphertext of one of the chained schemes $\mathsf{PKC}_{1,h}$ and we will use the associated decryption function.

Using a standard hybrid argument, the semantic security of this public key encryption scheme relies therefore on the security of the weakest scheme in the chain. As in the chained schemes security is always based in the same lattice problems but with growing gaps, the weakest will be the last one. According to the discussion provided in Section 4.3, if we want to use the scheme for the secure evaluation of a degree d polynomial, p_d may be in $\tilde{O}((\kappa \cdot M)^d)$. Combining this argument with the result stated in Theorem 1, we obtain that the semantic security of the new public key encryption scheme relies on the quantum hardness to approximate the decision version of GapSVP or the quantum hardness of SIVP to within $\tilde{O}(\kappa^{5/2} \cdot (\kappa \cdot M)^d)$ in the worst case.

The way how this public key encryption scheme can be used to securely evaluate a polynomial with up to M monomials and degree d is described in Sections 4.1 and 4.2. As we have argued in Section 4.3, in the case where the evaluation point is a binary vector, the communication cost of the evaluation protocol is in $\tilde{O}(d^5)$.

Finally, this basic protocol can be modified to enjoy formula privacy; that is, the client Bob does not obtain information about the polynomial to be evaluated by the server Alice. Such modification is now somewhat standard (see Section 7 of [10]) and works if the encryption schemes used in the chain are based on LWE (which is the case here). The idea is that adding to a given ciphertext an encryption of zero with an error super-polynomially larger than the error used in usual ciphertexts results in a ciphertext that still decrypts to the same result but statistically hides *which* ciphertext was initially given. Such a property is typically used to blind a ciphertext after a computation so that the final ciphertext only provides information about the result of the computation and not about how this result is obtained. □

Security relies on the hardness of solving two lattice problems with an approximation factor in $\tilde{O}(\kappa^{d+5/2} \cdot M^d)$. These problems are known to be hard as long as this factor is not exponential in the security parameter κ. We therefore need to suppose that M^d is bounded above by a sub-exponential function in κ to resist to these attacks. In particular, d will be bounded above by a polynomial in κ.

6 Acknowledgments

We would like to thank the anonymous reviewers who gave us many interesting recommendations in the previous versions of this paper. Given the recent results in the field some of the recommended modifications have not been done but will come in a future version of the paper.

References

1. Carlos Aguilar, Philippe Gaborit, and Javier Herranz. Additively homomorphic encryption with d-operand multiplications. In CRYPTO'10, volume 6223 of Lecture Notes in Computer Science, pages 138–154. Springer, 2010.
2. Carlos Aguilar Melchor, Guilhem Castagnos, and Philippe Gaborit. Lattice-based homomorphic encryption of vector spaces. In The 2008 IEEE International Symposium on Information Theory (ISIT'08), Toronto, Ontario, Canada, pages 1858–1862. IEEE Computer Society Press, 2008.
3. Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In CRYPTO'09, pages 595–618, Berlin, Heidelberg, 2009. Springer.
4. Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Theory of Cryptography Conference, TCC'2005, volume 3378 of Lecture Notes in Computer Science, pages 325–341. Springer, 2005.
5. Zvika Brakerski, Craig Gentry, and Shai Halevi. Fully homomorphic encryption without bootstrapping. In ITCS 2012 (to appear). Available at http://eprint.iacr.org/2011/277, 2012.
6. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In FOCS 2011 (to appear). Available at http://eprint.iacr.org/2011/344, 2011.
7. Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, volume 6841, page 501, 2011.
8. Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In Proceedings of the 31st annual conference on Advances in cryptology, CRYPTO'11, pages 487–504, Berlin, Heidelberg, 2011. Springer-Verlag.
9. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory, 31(4):469–472, 1985.
10. Craig Gentry. Fully homomorphic encryption using ideal lattices. In Proceedings of STOC'09, pages 169–178. ACM Press, 2009.
11. Craig Gentry. Toward basing fully homomorphic encryption on worst-case hardness. In CRYPTO'10, volume 6223 of Lecture Notes in Computer Science, pages 116–137. Springer, 2010.
12. Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In FOCS 2011 (to appear). Available at http://eprint.iacr.org/2011/279, 2011.
13. Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. In EUROCRYPT'2011, volume 6632 of Lecture Notes in Computer Science, pages 129–148. Springer, 2011.
14. Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. A simple BGN-type cryptosystem from LWE. In EUROCRYPT'2010, volume 6110 of Lecture Notes in Computer Science, pages 506–522. Springer, 2010.
15. Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In CRYPTO' 97, volume 1294 of Lecture Notes in Computer Science, pages 112–131. Springer, 1997.
16. Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In Theory of Cryptography Conference, TCC'2007, volume 4392 of Lecture Notes in Computer Science, pages 575–594. Springer, 2007.
17. Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Multi-bit cryptosystems based on lattice problems. In Public Key Cryptography, PKC'2007, volume 4450 of Lecture Notes in Computer Science, pages 315–329. Springer, 2007.
18. Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval (extended abstract). In FOCS: IEEE Symposium on Foundations of Computer Science (FOCS), pages 364–373, 1997.
19. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In EUROCRYPT'2010, volume 6110 of Lecture Notes in Computer Science, pages 1–23. Springer, 2010.
20. Rafail Ostrovsky and William E. Skeith III. Private searching on streaming data. J. Cryptology, 20(4):397–430, 2007.

21. Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In Proceedings of STOC'09, pages 333–342. ACM Press, 2009.
22. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In CRYPTO'08, volume 5157 of Lecture Notes in Computer Science, pages 554–571. Springer, 2008.
23. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM, 56(6):article 34, 2009.
24. R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. Communications of the ACM, 21(2):120–126, 1978.
25. S. Goldwasser and S. Micali. Probabilistic encryption. Journal of Computer and System Sciences, 28(2):270–299, 1984.
26. Tomas Sander, Adam Young, and Moti Yung. Non-interactive CryptoComputing for NC^1. In Proceedings of the 40th Symposium on Foundations of Computer Science (FOCS), pages 554–567, New York, NY, USA, October 1999. IEEE Computer Society Press.
27. Nigel Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Public Key Cryptography, PKC'2010, volume 6056 of Lecture Notes in Computer Science, pages 420–443. Springer, 2010.
28. Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In Masayuki Abe, editor, Advances in Cryptology - ASIACRYPT 2010, volume 6477 of Lecture Notes in Computer Science, pages 377–394. Springer, 2010.
29. Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In EUROCRYPT'2010, volume 6110 of Lecture Notes in Computer Science, pages 24–43. Springer, 2010.
30. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In 27th Annual Symposium on Foundations of Computer Science, pages 162–167, Toronto, Ontario, Canada, 27–29 October 1986. IEEE.

A Specific Realizations from LWE

In this appendix, after introducing the LWE problem, we describe a set of encryption schemes based on it, which can be used to instantiate our construction. The first one we introduce is a secret key encryption scheme, and then we succinctly present two public key alternatives. Even if the first encryption scheme cannot be considered fully-chainable, as it is a secret key scheme, the fact is that in the homomorphic encryption setting, and in most of the associated applications, there is no need to have a public key scheme as Bob is the only user encrypting and decrypting data. Thus it has an interest by itself, as it could be used *as is*. However, even if most of the claims and proofs in this paper can be straightforwardly adapted to the secret key setting, in order to be consistent with the protocols, we provide a modification which is a public key scheme.

A.1 Problem Definition

In [23], Oded Regev introduced a new problem, close to the learning from parity with noise problem on random linear codes, which has had a great impact on the lattice based cryptography community due to its simplicity and versatility: the Learning With Errors problem. There is a small set of possible definitions for this problem depending on whether we are interested on the decision or search version and whether we consider the average-case or the worst-case problem. The security of our encryption scheme, as usual in LWE-based schemes, is reduced from the average-case decisional version of the problem :

Definition 2 (Average-Case Decisional LWE$_{q,\chi}$). *Let $q, \kappa \in \mathbb{Z}^+$ be positive integers, and let $\chi : \mathbb{Z}_q \to \mathbb{R}^+$ be a probability distribution. Given a vector $\boldsymbol{s} \in \mathbb{Z}_q^\kappa$, let us consider the following distribution*

$$A_{\boldsymbol{s},\chi} = \{(\boldsymbol{a}, b) \mid \boldsymbol{a} \xleftarrow{\$} \mathbb{Z}_q^\kappa,\ b = \langle \boldsymbol{a}, \boldsymbol{s} \rangle + e,\ \text{with } e \xleftarrow{\chi} \mathbb{Z}_q\}$$

Given access to an oracle which samples $\mathbb{Z}_q^\kappa \times \mathbb{Z}_q$ uniformly or following $A_{\boldsymbol{s},\chi}$, decide which of the two distributions the oracle follows for a non-negligible fraction of the s.

Similarly we can define the worst-case decisional version in which for any s the attacker must distinguish both distributions with overwhelming probability, and the search versions in which the attacker must output s given samples of $A_{\boldsymbol{s},\chi}$ (for a non-negligible fraction of the s in the average case, and for any s with overwhelming probability in the worst case).

In [23], Regev reduces the average-case decisional version given above from the worst-case search version. This reduction is done in two steps. First, the average-case decisional version is reduced from the worst-case decisional version (Lemma 4.1 in [23]). Then, the worst-case decisional version is reduced from the worst-case search version (Lemma 4.2 in [23]). This part of the reduction requires q to be prime and polynomial in the security parameter (whereas the first part of the reduction has no specific requirements).

In order to connect the hardness of the worst-case search version of the problem to other standard lattice problems, Regev used a particular distribution χ: a folded and discretized Gaussian of parameter $\alpha \in \mathbb{R}^+$. Consider the following folded Gaussian

$$\forall r \in [0,1), \quad \Psi_\alpha(r) = \sum_{k=-\infty}^{+\infty} \frac{1}{\alpha} \cdot \exp\left(-\pi(\frac{r-k}{\alpha})^2\right)$$

We can now define $\bar{\Psi}_\alpha : \mathbb{Z}_q \to \mathbb{R}^+$ as the discrete probability distribution obtained by: sampling from Ψ_α, multiplying the result by q, and finally rounding to the closest integer modulo q. For $\chi = \bar{\Psi}_\alpha$, Regev proved that the worst-case hardness of search LWE, for adequate parameters, can be related to the worst-case *quantum* hardness of well established problems, as the following theorem, proved in [23], states.

Theorem 3. *Let κ, q be integers and $\alpha \in (0,1)$ be such that $\alpha q > 2\sqrt{\kappa}$. If there exists an efficient algorithm that solves worst-case search $LWE_{q,\bar{\Psi}_\alpha}$ then there exists an efficient quantum algorithm that approximates the decision version of the shortest vector problem (*GapSVP*) and the shortest independent vectors problem (*SIVP*) to within $\tilde{O}(\kappa/\alpha)$ in the worst case.*

A.2 A Secret Key Instantiation Based on LWE

We start presenting a secret key scheme which is a simple variation of the scheme proposed by Regev in [23] using different folklore improvements [17, 22] to obtain an asymptotic expansion factor of 1. This scheme illustrates how simply we can obtain a scheme based on LWE compatible with our construction, and can be easily transformed into a scheme based on other problems close to LWE, such as Ring-LWE [19].

Secret Key Scheme $\mathsf{SKC}{-}\mathsf{LWE}_{\kappa,m,n,p}$

$\mathsf{KeyGen}(\kappa, m, n, p)$:
We choose a prime number $q > \kappa mp$ and define $\chi = \bar{\Psi}_\alpha$, where $\alpha = 2/(\sqrt{\kappa}mp)$. The secret key is a set of n vectors $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n \in \mathbb{Z}_q^\kappa$ chosen uniformly at random.

$\mathsf{Enc}((\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n), \mathbf{x})$ for $\mathbf{x} = (x^{(1)}, \ldots, x^{(n)}) \in \mathbb{Z}_p^n$:
Choose a random vector $\boldsymbol{a} \in \mathbb{Z}_q^\kappa$ uniformly. Sample n times χ to obtain n values $e^{(1)}, \ldots, e^{(n)}$ and output $(\boldsymbol{a}, \boldsymbol{b})$ with $b = (b^{(1)}, \ldots, b^{(n)})$ and $b^{(i)} =< \boldsymbol{a}, \boldsymbol{s}_i > + pe^{(i)} + x^{(i)} \bmod q$.

$\mathsf{Dec}((\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n), (\boldsymbol{a}, \boldsymbol{b}))$:
Output $\mathbf{x} = (x^{(1)}, \ldots, x^{(n)})$ with $x^{(i)} = (b^{(i)} - < \boldsymbol{a}, \boldsymbol{s}_i > \bmod q) \bmod p$.

Note that the n secret keys $s_1, \cdots, \boldsymbol{s}_n$ allow to encrypt multiple elements of $\mathbb{Z}_p$ in a single ciphertext $(\boldsymbol{a}, (b^{(1)}, \ldots, b^{(n)}))$ and thus to amortize $\boldsymbol{a}$ as n grows (which is a standard technique). However, in order to reduce an attack on such ciphertexts (using an hybrid argument) from the decisional LWE setting in which we have a pair $(\boldsymbol{a}, b)$ (b being a scalar encoding a single element of $\mathbb{Z}_p$), we must take the n secret keys independently and thus the secret key grows linearly in n.

A.3 Indistinguishability

Let us prove now that this scheme is IND-CPA secure. We recall first the standard notion of indistinguishability under chosen-plaintext attacks (IND-CPA security), for a public key encryption scheme $\mathsf{PKC} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. We use the following game that an attacker $\mathcal{A}$ plays against a challenger:

$$
\begin{array}{l}
(pk, sk) \leftarrow \mathsf{KeyGen}(1^\kappa) \\
(St, a_0, a_1) \leftarrow \mathcal{A}(\text{find}, pk) \\
b \leftarrow \{0,1\} \text{ at random} \\
c^* \leftarrow \mathsf{Enc}(pk, a_b) \\
b' \leftarrow \mathcal{A}(\text{guess}, c^*, St).
\end{array}
$$

The advantage of such an adversary $\mathcal{A}$ is defined as

$$\mathrm{Adv}(\mathcal{A}) = \left| \Pr[b' = b] - \frac{1}{2} \right|.$$

A public key encryption scheme enjoys IND-CPA security if $\mathrm{Adv}(\mathcal{A})$ is a negligible function of the security parameter κ, for any attacker $\mathcal{A}$ running in polynomial time (in κ). For a secret key encryption scheme the game is trivially translated with a major exception. In the last phase of the game, the attacker must have access to an encryption oracle for the secret key involved in the game.

In order to reduce the indistinguishability of $\mathsf{SKC}-\mathsf{LWE}$ from standard problems we will first reduce it from average-case decisional LWE in Lemma 1. We will then reduce this problem from worst-case search LWE in Lemma 2 for parameters for which the reduction given by Regev in [23] is not valid, and finally we will note that our parameters allow to use Theorem 3 to reduce this problem from GapSVP and SIVP. In the following lemmas, κ will be the security parameter, n a parameter of the encryption scheme polynomial in κ, and m, p two other parameters of the scheme which are (possibly exponential) functions of κ.

Lemma 1. *$\mathsf{SKC}-\mathsf{LWE}_{\kappa,m,n,p}$ enjoys indistinguishability against chosen-plaintexts attacks if the average-case decisional $LWE_{q,\bar{\Psi}_\alpha}$ problem is hard for $q \in [\kappa mp, 2\kappa mp)$ prime and $\alpha = 2/(\sqrt{\kappa}mp)$.*

Proof (Sketch.). Suppose that an attacker given access to an encryption oracle, can distinguish with a non-negligible advantage the encryptions of two plaintexts $\mathbf{x_1}, \mathbf{x_2} \in \mathbb{Z}_p^n$. Using a standard hybrid argument, we can say that the attacker can distinguish with a non-negligible advantage between encryptions of $\mathbf{x}_i$ and elements uniformly chosen in $\mathbb{Z}_q^\kappa \times \mathbb{Z}_q^n$ for $i = 1$ or $i = 2$. Suppose, w.l.o.g., that we have $i = 1$. Again, using a standard hybrid argument and the fact that n is polynomial in κ, we can say that there is a coordinate $j \in [1, n]$ such that the attacker can distinguish with a non-negligible advantage between pairs $(\boldsymbol{a}, b^{(j)})$, $(\boldsymbol{a}, \boldsymbol{b})$ being a random encryption of $\mathbf{x}_1$, and uniform elements in $\mathbb{Z}_q^\kappa \times \mathbb{Z}_q$. Such a distinguisher can be used for decisional $\mathrm{LWE}_{q,\chi}$. Indeed, consider the transformation that takes an element from $Z_q^\kappa \times \mathbb{Z}_q$, multiplies it by p (which is coprime with q) and adds to the second coordinate $\mathbf{x}_1^{(j)}$. This transformation takes the uniform distribution to itself on the one hand and $A_{s,\chi}$ to the pairs $(\boldsymbol{a}, b^{(j)})$ for $(\boldsymbol{a}, \boldsymbol{b})$ a random encryption of $\mathbf{x}_1$ on the other hand.

The original distinguisher needs access to an encryption oracle, but in decisional $\mathrm{LWE}_{q,\chi}$ the available oracle might just provide uniform elements of $\mathbb{Z}_q^\kappa \times \mathbb{Z}_q$. However, testing the performance of the distinguisher with trial cases, as done by Regev in [23] (Lemma 5.4), we can ensure that this reduces the advantage of the final distinguisher by at most a constant factor. □

The construction provided in this paper requires instances of $\mathsf{SKC}-\mathsf{LWE}$ with q sub-exponential in κ. This implies that we cannot use the Lemma 4.2 given by Regev in [23] to reduce worst-case decisional $\mathrm{LWE}_{q,\chi}$ to worst-case search $\mathrm{LWE}_{q,\chi}$. The next step in our reduction process therefore uses the first part of the reduction given by Regev (Lemma 4.1 in [23]) and then a Lemma by Applebaum et al. given in [3].

Lemma 2. *Assume that there exists a polynomial-time algorithm W solving the average-case decisional version of $LWE_{q,\bar{\Psi}_\alpha}$, $q \in [\kappa mp, 2\kappa mp)$ being prime and $\alpha = 2/(\sqrt{\kappa}mp)$. Then, there is a polynomial-time algorithm solving the worst-case search version of $LWE_{q,\bar{\Psi}_\alpha}$.*

Proof. Suppose that we have access to the algorithm W. Using the transformation provided in [23] (Lemma 4.1), we can use W to solve decisional $\mathrm{LWE}_{q,\chi}$ in the worst-case as the reduction given works for any q and χ. We note the algorithm obtained W'. Suppose that we have a challenge for the worst-case search version of $\mathrm{LWE}_{q,\chi}$. We can sample $A_{s,\chi}$ for a given s and we must find s with overwhelming probability.

As noted before we cannot use the approach of Regev which consists on trying to guess one by one the coordinates of s and using the distinguisher W' to know if we have done the right choice. Indeed, there may be a sub-exponential number of possibilities (because q may be sub-exponential in κ) for each coordinate and we can only test a negligible fraction of them in polynomial time.

We therefore use the deterministic polynomial-time transormation proposed in [3] (Lemma 2), which maps $A_{s,\chi}$ to $A_{\bar{x},\chi}$ where $\bar{x} \leftarrow \chi^n$ and leaves the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ unchanged. This transformation also provides a matrix and a vector allowing to compute s from $\bar{x}$ and thus to solve the search challenge we only need to get $\bar{x}$.

As $\chi = \bar{\Psi}_\alpha$ the coordinates of $\bar{x}$ will be bounded by $q\alpha \times \sqrt{\kappa} < 2\kappa$ with overwhelming probability. We can therefore use the approach given in [23] (Lemma 4.2) to test each coordinate at a time using W' to obtain $\bar{x}$ and thus s. □

The choice of parameters provided on the key generation algorithm ensures that $q\alpha > 2\sqrt{\kappa}$ and thus that Theorem 3 applies to our setting. Using this fact together with Lemmas 1 and 2 we obtain the following proposition.

Proposition 4 (Indistinguishability). $\mathsf{SKC{-}LWE}_{\kappa,m,n,p}$ *enjoys indistinguishability against chosen-plaintexts attacks, assuming the quantum hardness to approximate the decision version of* GapSVP *or the quantum hardness of* SIVP *to within* $\tilde{O}(\kappa\sqrt{\kappa}mp)$ *in the worst case.*

There is no polynomial-time algorithm known to solve these problems even for sub-exponential approximation factors. As long as this remains true, our scheme is secure if n is polynomial and m, p are at most sub-exponential in κ.

A.4 Correctness

We show now that decryption is correct with overwhelming probability even if m ciphertexts are added up. This is dealt with in the following proposition.

Proposition 5 (m-limited homomorphism). *Let* $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n) \leftarrow \mathsf{KeyGen}(\kappa, m, n, p)$ *be an instance of* $\mathsf{SKC{-}LWE}$*, and* $\mathbb{Z}_{p'}^{n'}$ *the closure of its ciphertext space. For any* $\ell \leq m$*,* $\mathbf{x}_1, \ldots, \mathbf{x}_\ell \in \mathbb{Z}_p^n$ *and any* $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_\ell$ *with* $\boldsymbol{\beta}_i \in \mathsf{Enc}((\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n), \mathbf{x}_i)$*, the vector* $\boldsymbol{\beta} = \sum_i \boldsymbol{\beta}_i \bmod p'$ *is decrypted via* Dec *to the integer vector* $\mathbf{x} = \sum_i \mathbf{x}_i \bmod p$*.*

Proof (Sketch.). We have $\boldsymbol{\beta} = (\boldsymbol{a}, \boldsymbol{b})$ with $\boldsymbol{a} = \sum_i \boldsymbol{a}_i$ and $b^{(j)} =< \boldsymbol{a}, \boldsymbol{s}_j > + p\sum_i e_i^{(j)} + \sum_i x_i^{(j)} \bmod q$, using the notations proposed in the algorithms of $\mathsf{SKC{-}LWE}$. The decryption function returns

$$\mathbf{x}' \text{ with } x'^{(j)} = (b^{(j)} - < \boldsymbol{a}, \boldsymbol{s}_j > \bmod q) \bmod p = (p \sum_i e_i^{(j)} + \sum_i x_i^{(j)} \bmod q) \bmod p$$

Thus, in order to ensure correctness it is enough to have $p\sum_i e_i^{(j)} + \sum_i x_i^{(j)} < q$ for each $j \in [1, n]$ as in this case we have $p\sum_i e_i^{(j)} + \sum_i x_i^{(j)} \bmod q = p\sum_i e_i^{(j)} + \sum_i x_i^{(j)}$ (over the integers) and thus

$$x'^{(j)} = (p \sum_i e_i^{(j)} + \sum_i x_i^{(j)}) \bmod p = \sum_i x_i^{(j)} = x^{(j)}$$

We therefore just need to prove that with overwhelming probability we have $p\sum_i e_i^{(j)} + \sum_i x_i^{(j)} < q$, for all $j \in [1, n]$.

A first trivial bound is $\sum_i x_i^{(j)} < \ell p \leq mp$. In order to bound the other term, it is possible to use the same argumentation as the one Regev uses in [23] (Claim 5.2), which put short says that $p\sum_i e_i^{(j)}$ is p times a sum of up to m roundings and thus the difference between this term and multiplying by q a sampling from $\Psi_{p\sqrt{m}\alpha}$ is at most pm. The expected value of such a multiplied sample is $p\sqrt{m}q\alpha \simeq 2\sqrt{\kappa m}p < q/(1/2\sqrt{\kappa})$. Thus the probability that $p\sum_i e_i^{(j)} + \sum_i x_i^{(j)} > q$ is exponentially small in κ and thus negligible in the security parameter. As n is polynomial in κ this remains true even when we consider all the coordinates. □

We can therefore conclude that, apart from not being a public key encryption scheme, the properties of a fully-chainable scheme are verified by $\mathsf{SKC}-\mathsf{LWE}$. Indeed, we can choose the parameters m, n, p, and the ciphertext space is included in $Z_{p'}^{n'}$ with $n' = f_n(\kappa, m, n, p) = \kappa + n$ and $p' = f_p(\kappa, m, n, p) = \kappa mnp + \varepsilon$ where $\varepsilon < \kappa mnp$ by the Bertrand-Chebychev theorem on prime numbers. Moreover, the null ciphertext is clearly a particular encryption of the null plaintext. Fortunately, it is pretty trivial to transform it into a public key scheme.

A.5 Public-Key Realization

Two alternatives are possible in order to obtain a public key encryption scheme able to instantiate efficiently our construction. Both will be presented here just succinctly because of their simplicity.

The first one is to transform the symmetric scheme presented in the previous subsection into a public scheme using the same technique than Regev in [23]. That is, publish a large enough set of random encryptions of the null vector as the public key, and keep the symmetric key as the secret key. In order to encrypt, it is enough to do a random subset sum of the public key ciphertexts and add the message to the result. The decryption algorithm remains the same than for the symmetric scheme. Because of the homomorphic properties of the scheme, if we set the homomorphic parameter to $m = Cm'$, C being the number of ciphertexts in the public key, we obtain an instance of a public key encryption scheme which is an m'-limited homomorphism. The value proposed in [23] is $C = (1+\varepsilon)(\kappa+1)\log q$ for $\varepsilon > 0$. In order to simplify the formulas in our work we set $C = 2\kappa \log q$. The security proof, transforming an IND-CPA distinguisher into a decision LWE distinguisher being the same as in [23] we don't present here the details.

The second alternative is to modify the parameters of the encryption scheme proposed by Gentry et al. in [14]. Indeed, their proposal satisfies all the properties needed for a fully-chainable encryption scheme (if we transform $\ell \times \ell$ matrices into vectors of ℓ^2 coordinates), but results in an expansion factor which is bounded below by 2. Thus, even if it can be used with our construction, the expansion factor after chaining d schemes is bounded below by 2^d. This bound is a consequence of an extra functionality of their system which allows to do a twisted multiplication. This property results in a protocol for the evaluation of degree 2 polynomials over encrypted data, as in [4]. If we drop this functionality it is possible to change their parameters in order to have an expansion factor arbitrarily close to 1 and therefore, when chaining d schemes, reach expansion factors which are just polynomial in d.

B Toy Example

Let PKC_1, PKC_2 be chainable schemes such that $\mathsf{Enc}_1 : \mathbb{Z}_5 \to \mathbb{Z}_{79}^2$ and $\mathsf{Enc}_2 : \mathbb{Z}_{79}^2 \to \mathbb{Z}_{p_3}^{n_3}$, for some integer values p_3, n_3. That is, we have $p_1 = 5$, $p_2 = 79$ and $n_2 = 2$. A protocol for the secure evaluation of the product $a_1 a_2$, being $a_1 = 3$ and $a_2 = 4$, would work as follows.

Bob computes an encryption $\boldsymbol{\alpha} = (6, 38) \in \mathsf{Enc}_1(3)$. To encrypt $a_2 = 4$, Bob computes encryptions $\boldsymbol{\beta}_{i,j} \in \mathsf{Enc}_2((4 \cdot 2^j)\boldsymbol{e}_i \bmod 79)$, for $i = 1, 2$ and $j \in \{0, 1, \ldots, 6\}$. In other words:

$$\begin{array}{l}
\boldsymbol{\beta}_{1,0} \in \mathsf{Enc}_2((4,0)),\ \ \boldsymbol{\beta}_{1,1} \in \mathsf{Enc}_2((8,0)),\ \boldsymbol{\beta}_{1,2} \in \mathsf{Enc}_2((16,0)), \boldsymbol{\beta}_{1,3} \in \mathsf{Enc}_2((32,0)),\\
\boldsymbol{\beta}_{1,4} \in \mathsf{Enc}_2((64,0)), \boldsymbol{\beta}_{1,5} \in \mathsf{Enc}_2((49,0)), \boldsymbol{\beta}_{1,6} \in \mathsf{Enc}_2((19,0)),\\
\boldsymbol{\beta}_{2,0} \in \mathsf{Enc}_2((0,4)),\ \ \boldsymbol{\beta}_{2,1} \in \mathsf{Enc}_2((0,8)),\ \boldsymbol{\beta}_{2,2} \in \mathsf{Enc}_2((0,16)), \boldsymbol{\beta}_{2,3} \in \mathsf{Enc}_2((0,32)),\\
\boldsymbol{\beta}_{2,4} \in \mathsf{Enc}_2((0,64)), \boldsymbol{\beta}_{2,5} \in \mathsf{Enc}_2((0,49)), \boldsymbol{\beta}_{2,6} \in \mathsf{Enc}_2((0,19)).
\end{array}$$

The product protocol takes the first component $\alpha^{(1)} = 6$ of $\boldsymbol{\alpha}$, considers its bit representation 0110000 and computes

$$\boldsymbol{\gamma}_1 = \boldsymbol{\beta}_{1,1} + \boldsymbol{\beta}_{1,2} \bmod p_3 \in \mathsf{Enc}_2((24, 0)).$$

Due to the homomorphic properties of PKC_2, we indeed have $\boldsymbol{\gamma}_1 \in \mathsf{Enc}_2((24, 0))$. Analogously, since the second component $\alpha^{(2)} = 38$ of $\boldsymbol{\alpha}$ has bit representation 0110010, we would have

$$\boldsymbol{\gamma}_2 = \boldsymbol{\beta}_{2,1} + \boldsymbol{\beta}_{2,2} + \boldsymbol{\beta}_{2,5} \bmod p_3 \in \mathsf{Enc}_2((0, 73)).$$

The final ciphertext output by the product protocol is $\boldsymbol{\gamma} = \boldsymbol{\gamma}_1 + \boldsymbol{\gamma}_2$, which satisfies $\boldsymbol{\gamma} \in \mathsf{Enc}_2((24, 73))$, due again to the homomorphic properties of PKC_2.

If now we apply the decryption procedure Dec of $\mathsf{PKC} = \mathsf{chain}(\mathsf{PKC}_1, \mathsf{PKC}_2)$ to the ciphertext $\boldsymbol{\gamma}$, we first run Dec_2 on $\boldsymbol{\gamma}$, which results in $(24, 73)$, and then we run $\mathsf{Dec}_1((24, 73))$. Note that, due to the homomorphic properties of PKC_1, since we had $(6, 38) \in \mathsf{Enc}_1(3)$, we know that

$$(6, 38) + (6, 38) + (6, 38) + (6, 38) \bmod 79 \in \mathsf{Enc}_1(3 + 3 + 3 + 3 \bmod 5).$$

Summing up, we have $(24, 73) \in \mathsf{Enc}_1(2)$. Therefore, the final output of the decryption procedure would be $\mathsf{Dec}_1((24, 73)) = 2 = 3 \cdot 4 \bmod 5 = a_1 a_2 \bmod p_1$, as desired.