Boosting Linearly-Homomorphic Encryption to Evaluate Degree-2 Functions on Encrypted Data

Dario Catalano[1] and Dario Fiore[2]

[1] Dipartimento di Matematica e Informatica, Università di Catania, Italy.
`catalano@dmi.unict.it`
[2] IMDEA Software Institute, Madrid, Spain.
`dario.fiore@imdea.org`

Abstract. We show a technique to transform a linearly-homomorphic encryption into a homomorphic encryption scheme capable of evaluating degree-2 computations on ciphertexts. Our transformation is surprisingly simple and requires only one very mild property on the underlying linearly-homomorphic scheme: the message space must be a public ring in which it is possible to sample elements uniformly at random. This essentially allows us to instantiate our transformation with virtually all existing number-theoretic linearly-homomorphic schemes, such as Goldwasser-Micali, Paillier, or ElGamal. Our resulting schemes achieve circuit privacy and are compact when considering a subclass of degree-2 polynomials in which the number of additions of degree-2 terms is bounded by a constant.
As an additional contribution we extend our technique to build a protocol for outsourcing computation on encrypted data using two (non-communicating) servers. Somewhat interestingly, in this case we can boost a linearly-homomorphic scheme to support the evaluation of any degree-2 polynomial while achieving full compactness.

1 Introduction

The problem of enabling computation over encrypted data is one of the most intriguing questions in cryptography and has been attracting significant attention lately. In a nutshell, this problem involves two parties, a client holding an input x and a server holding some function f. The goal of the client is to learn $f(x)$ without disclosing unnecessary information about x (aka *semantic security*). The goal of the server is to perform the computation without revealing to the client any information (beyond $f(x)$) about f (aka *circuit privacy*). Moreover, to minimize the communication between client and server as well as the client's work, it would be desirable that the server's response be shorter than the size of f (aka *compactness*). Computing on encrypted data is a problem that arises in a variety of settings, including, for instance, secure cloud computing, encrypted database search, and many more.

A natural way to solve the problem is to rely on so-called *homomorphic encryption (HE) schemes*. Informally, these are encryption mechanisms that allow one to perform computations on the encrypted plaintexts by performing similar operations on the ciphertexts. In other words, homomorphic encryption allows to perform meaningful operations on plaintexts (e.g., additions and multiplications) without, at any stage, needing to decrypt the corresponding ciphertexts.

The idea of homomorphic encryption was first suggested in 1978 by Rivest, Adleman and Dertouzous [30], though its first fully-fledged realization was proposed only in 2009 in a breakthrough result by Gentry [17]. Earlier than that, many other authors suggested encryption schemes that, albeit not supporting arbitrary functionalities, still allow for meaningful operations. This is the case, for instance, of the Goldwasser-Micali cryptosystem [20], Paillier's cryptosystem [29] and many other schemes [9,27,28,12,6,22]. All these schemes are *linearly-homomorphic* (i.e., they support linear functions only), and they can be seen as based on the same blueprint. Namely, they are all (probabilistic) public-key encryption schemes based on a discrete log trapdoor modulo a large integer which is hard to factor.[3] In such schemes, the message space is a ring $\mathcal{M}$ of modular residues and ciphertexts are in the group $\mathbb{G}$ (denoted multiplicatively) of invertible elements of some particular ring of integers modulo a number hard to factor. The encryption of a message m is a group element of the form $\mathsf{Enc}(m; r) = g^m r^e \in \mathbb{G}$, where e is some public integer, g a fixed public element, and r is chosen at random in some particular (multiplicative) subgroup R of $\mathbb{G}$. Since R is a subgroup, such schemes have an additive homomorphic property: an encryption of $m_1 + m_2$ can be obtained from any encryption of m_1 and m_2, as $E(m_1; r_1) \cdot E(m_2; r_2) \equiv E(m_1 + m_2; r_1 r_2)$. In other words, (homomorphic) additions of plaintexts are obtained by multiplying the corresponding ciphertexts.

Generalizing these schemes to support more complex functionalities – say, multiplications – seems like a lost cause at first as, being the ciphertext space only a group, in general there might be no way to operate on two ciphertexts in order to obtain a multiplication of the corresponding plaintexts. A nice exception to this barrier was suggested by Boneh, Goh and Nissim [3] who revisited the above blueprint in the context of composite-order bilinear groups, and in this setting show how to use the bilinear map to gain one single multiplication on encrypted plaintexts. However, this construction is very specific to bilinear groups, and it remains an intriguing open problem whether it is possible to extend *any* linearly-homomorphic scheme (e.g., Paillier or Goldwasser-Micali) in a natural way in order to support multiplications.[4] Beyond its theoretical interest, answering this question in the positive, might allow to build homomorphic cryptosystems that could adopt (*directly* and for *free*!) many of the satellite protocols and tools (e.g., ZK-PoK, threshold variants and so on) developed for the underlying linear schemes over the last thirty+ years.

1.1 Our Contribution

Homomorphic Encryption for Quadratic Functions. Our main result is a way to generalize the blueprint described above[5] in order to gain the possibility of performing one multiplication on encrypted plaintexts. Slightly more in detail, we show a simple transformation which takes a linearly-homomorphic encryption scheme and uses it to build an HE scheme which supports arithmetic computations of degree 2 on ciphertexts.[6]

Our transformation is quite generic and requires only one very mild property from the underlying linearly-homomorphic scheme: the message space must be a public ring in which it is possible to sample elements uniformly at random. We call HE schemes satisfying this property *public-space* and we show that virtually all existing schemes are so (or can be easily modified in order to become so). This means that we can instantiate our transformation with a variety of existing schemes (e.g., [19,9,27,29,12,6,22]) thus obtaining several HE schemes capable of evaluating one multiplication and whose security relies on a variety of assumptions, such as quadratic/composite residuosity, DDH, or decision linear, to name a few. Furthermore, when applied to the BGN encryption scheme [3], our solution yields an HE scheme that supports computations of degree up to 4.

Our technique is surprisingly simple, and at an intuitive level it works as follows. Starting from a linearly homomorphic encryption scheme $\mathcal{HE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ based on the blueprint described

[3] An exception is the scheme by Bresson *et al.* [6] in which the ciphertext is composed by two group elements, as well as schemes such as ElGamal "in the exponent" [10] where the modulus N is allowed to be prime.

[4] At this point, it is worth noting that Ishai and Paskin [21] builds homomorphic encryption from a linearly-homomorphic scheme, albeit in a "less natural way". We discuss this work in more detail in Section 1.2.

[5] Actually, we do not need to assume the group structure – we do it here only for ease of exposition.

[6] Precisely, our solution achieves compactness for a subclass of degree-2 polynomials that we specify slightly below.

above, we encrypt a message m as $C = (m - b, \mathsf{Enc}(b))$ where b is random in $\mathcal{M}$ and "$-$" denotes subtraction (i.e., addition of the additive inverse) in the ring $\mathcal{M}$. With such ciphertexts, the scheme is clearly linearly-homomorphic. To perform the multiplication one proceeds as follows. Given $C_1 = (m_1 - b_1, \mathsf{Enc}(b_1))$ and $C_2 = (m_2 - b_2, \mathsf{Enc}(b_2))$ one first computes the product $(m_1 - b_1)(m_2 - b_2)$, moves it up to the ciphertext space (i.e., encrypts it), and then (homomorphically) removes the terms $m_1 b_2$ and $m_2 b_1$ from the product by exploiting the linearly-homomorphic properties of $\mathcal{HE}$. Slightly more in detail, one computes $C = (\alpha, \beta_1, \beta_2)$ where:

$$\begin{aligned} \alpha &= \mathsf{Enc}(m_1 m_2 - b_1 b_2) = \mathsf{Enc}\left((m_1 - b_1) \cdot (m_2 - b_2)\right) \cdot [\mathsf{Enc}(b_1)^{(m_2 - b_2)}] \cdot [\mathsf{Enc}(b_2)^{(m_1 - b_1)}] \\ \beta_1 &= \mathsf{Enc}(b_1), \qquad\qquad \beta_2 = \mathsf{Enc}(b_2) \end{aligned}$$

Decryption of any ciphertext $(\alpha, \beta_1, \beta_2)$ works by first retrieving b_1 and b_2 (from β_1 and β_2) and then by subtracting $(b_1 b_2)$ from the decryption of α. At this point, once obtained level-2 ciphertexts via the above manipulations, these can be kept additively-homomorphic. This however comes at the cost of loosing compactness. In fact, addition at level 2 consists into homomorphically adding the α components of the ciphertexts and concatenating the β components. So the ciphertext will start growing (linearly) with additions after performing the multiplication. Importantly, however, we show a technique to re-randomize ciphertexts so as to achieve circuit privacy. To be precise, our scheme compactly supports all degree-2 multivariate polynomials in which the number of additions of degree-2 terms is bounded by a constant L, i.e., polynomials of the form $P(\boldsymbol{x}) + \sum_{i=1}^{L} Q_i(\boldsymbol{x}) \cdot R_i(\boldsymbol{x})$ where P, Q_i, R_i are linear. Despite limited, this class of computations is still meaningful in some contexts. For instance, we observe that the celebrated SPDZ protocol [13] requires a somewhat homomorphic encryption scheme capable of evaluating exactly this class of computations (for $L = 1$). Finally, as we illustrate in a couple of paragraphs below, we show how to completely remove this compactness issue in a novel protocol for outsourcing computation on encrypted data using two servers. In this case we can boost linearly-homomorphic encryption to evaluate *any* degree-2 multivariate polynomial on encrypted data, with compact ciphertexts.

On the importance of being simple and black box. Beyond its simplicity, a very attractive feature of our transformation is that it applies in a black box way to essentially all known linearly-homomorphic encryption schemes. This allows us to directly "inherit" virtually all the protocols and tools previously developed for the underlying schemes.[7] For instance, if one starts from a cryptosystem $\mathcal{HE}$ admitting a threshold variant $\mathcal{HE}_T$, by applying our trasformation to $\mathcal{HE}_T$ one immediately gets a threshold homomorphic cryptosystem supporting one multiplication. Similarly, if the underlying scheme admits a zero knowledge proof of plaintext knowledge protocol, the *same* protocol (see Section 4) can be used to prove the *same* statement for the scheme resulting when applying our transformation.

Providing alternatives to existing schemes. Our construction provides the first realizations of practical homomorphic encryption schemes (beyond linear) in groups like $\mathbb{Z}_N^*$. Compared to the recent lattice-based homomorphic encryption schemes, ours are clearly much less expressive. Yet we believe that our results remain relevant for at least two reasons.

First, they provide alternatives to lattice-based cryptography. Given our current understanding of complexity theory it is possible to imagine a world where the lattice problems used to build homomorphic encryption are easy to solve while (some) number theoretic problems remain hard. Notice that we are not saying that this scenario is plausible or even likely, just that it is *possible*.

[7] An exception to this rule is the bit security analysis of [8] that does not survive our transformation.

Second, our schemes can immediately take advantage of the thirty+ years efforts done on number-theoretic (linearly) homomorphic encryption schemes. Indeed, adopting our schemes induces only a very small overhead (i.e., one ring element more in the ciphertext, which is one bit in the case of Goldwasser-Micali), and this holds also from an implementation perspective (no need to develop complex new libraries or to deal with new delicate parameters choices).

Overcoming the Compactness Issue: Two-Server Delegation of Computation on Encrypted Data. As a second contribution, we show how to extend our techniques to build a protocol for outsourcing computation on encrypted data by using two servers. In brief, two-server delegation of computation enables a client to outsource the computation of a function f on some input m by using two distinct servers in such a way that: it is possible to recover the result $f(m)$ from the servers' response, and the two servers do *not* need to interact. For security, the protocol preserves the confidentiality of the client's inputs as long as the two servers do not collude. Somewhat interestingly, for this protocol we show a construction which completely removes the compactness issue arising in our previous transformation. Namely, we use a linearly-homomorphic scheme in order to outsource the evaluation of *any* multivariate polynomial of degree 2 while keeping the ciphertexts (i.e., the servers' responses) of constant-size. This for instance offers a solution for outsourcing the computation on encrypted data of various statistical functions (e.g., mean, variance, covariance) or distance measures (e.g., euclidean distance) over vectors of integers. And interestingly, all this can be performed rather efficiently by using cryptosystems such as Paillier's [29] or Joye-Libert [22].

Actually, our solution offers two more interesting properties that we call *unbalanced efficiency* and *unbalanced security*. The former says that one of the two servers can perform the computation essentially "for free", i.e., as if it computes over the original inputs in $\mathcal{M}$. This is possible because only one of the servers need to do public-key operations. Unbalanced security instead guarantees that w.r.t. such efficient server the semantic security of the protocol holds information-theoretically.

Finally, we show (cf. Section 5.4) that two-server delegation of computation can be used to build protocols for *server-aided secure function evaluation* [23]. By using our construction, we obtain an SFE semi-honest protocol that uses two servers for outsourcing the computation of degree-2 polynomials (or degree-3 by using BGN), and these servers do *not* need to communicate at all. We stress that such absence of communication further justifies the plausibility of assuming absence of collusions between the servers.

Our construction of two-server delegation of computation on encrypted data builds on the same idea illustrated before, with the difference that now a ciphertext consists of two components: one for the first server and one for the second server. More in detail, the first server receives $C^{(1)} = (m-b, \mathsf{Enc}(b))$ – i.e., a ciphertext of our HE scheme – while the second server receives just $C^{(2)} = b$. As one can notice, as long as the two servers do not collude, the confidentiality of the message m is preserved. In order to perform computations on these ciphertexts, the second server simply operates over the b_i's, i.e., it computes $b = f(b_1, \dots, b_t)$, while the first server works as in our HE scheme with the key difference that, after performing a multiplication, it can throw away the β components of the ciphertexts and keep doing additions on the α components *without any ciphertext growth*. This way, the first server is eventually able to compute $\alpha = \mathsf{Enc}(f(m_1, \dots, m_t) - f(b_1, \dots, b_t))$. So, when the client receives such values α and $b = f(b_1, \dots, b_t)$ from the first and the second server respectively, then it can recover the computation's result as $\mathsf{Dec}(\alpha) + b$. It is interesting to note that the two servers do not need to interact during the computation, and actually they do not even need to know about their mutual existence!

As an extension, we further generalize our technique to outsource the computation of all degree-3 polynomials by using a level-2 homomorphic encryption. Such transformation yields for instance a protocol for degree-3 polynomials based on the BGN cryptosystem.

1.2 Other Related Work

As already mentioned above, fully homomorphic encryption (FHE) allows to perform arbitrary computation over encrypted data (e.g., [17,32,33,5,4]). A key feature of FHE is that ciphertexts are both compact and guarantee circuit privacy. Beyond FHE-based solutions, many other works considered the problem of computing over encrypted data. Cachin *et al.* [7] observed that (any) two-message protocol for secure function evaluation (SFE) can be used to perform computation over encrypted data with circuit privacy. In particular, this idea can be extended to construct an homomorphic public key cryptosystem from any two-message SFE. This construction uses Yao's garbled circuits [34] as underlying building block, achieves semantic security and circuit privacy but unfortunately is not compact (roughly, because the ciphertext includes a garbled circuit). Gentry *et al.* [18] generalized this construction to support computations on previously evaluated ciphertexts via bootstrapping (they called *i-hop* an homomorphic cryptosystem allowing up to i such computations). Other works proposed HE schemes whose complexity of decryption (and ciphertext size) depends in various ways on the evaluated function f. Here we discuss some of them.

Sander *et al.* [31] proposed a solution to evaluate *constant* fan-in (boolean) circuits in NC^1. More precisely, their scheme allows to evaluate circuits (in NC^1) composed of OR and NOT gates. Unfortunately however, it also requires communication complexity exponential in the depth of the circuit, as ciphertexts grow exponentially with the number of OR gates.

Building on earlier work of Kushilevitz and Ostrovsky [24], Ishai and Paskin [21] proposed a scheme to evaluate branching programs on encrypted data. Their protocol uses *strong* oblivious transfer, a notion that can be implemented using any linearly homomorphic encryption. This makes this work somewhat related to ours. In comparison, their scheme clearly supports a wider class of functionalities. On the other hand, if we consider the question of building upon a linearly-homomorphic encryption to obtain more expressive functionalities, their construction is less direct than ours: for instance, they have to change the computation model to branching programs, and it is unclear whether tools originally designed for the underlying HE are "recyclable" in the transformed scheme. Moreover, while when focusing on all degree-2 polynomials we achieve the same level of compactness, instead for the specific subset of polynomials considered in this paper[8] the scheme in [21] induces much larger ciphertexts (quadratic in the number of computation's inputs).

Another work related to ours is the one of Aguilar Melchor *et al.* [26] who proposed a construction of homomorphic encryption (called *chained*) supporting up to d multiplications, for some constant d. In this scheme the ciphertext grows exponentially with each multiplication (but is not affected by additions). The basic idea is somewhat similar to ours. There they show how to achieve chained encryption out of a linearly-homomorphic scheme with certain properties. These properties, however, are more stringent than those required in this paper. In particular, none of the currently available number-theoretic cryptosystems is known to meet such requirements.

[8] We stress that, although our solution is not fully compact when considering all degree-2 polynomials, it achieves compactness for the specific subset of degree-2 polynomials discussed above.

2 Preliminaries

In this section, we review the notation and some basic definitions that we will use in our work.

We will denote with $\lambda \in \mathbb{N}$ the security parameter, and by $\mathsf{poly}(\lambda)$ any function which is bounded by a polynomial in λ. Informally, we say that a function $\epsilon(\lambda)$ is *negligible* if it vanishes faster than the inverse of any polynomial in λ, and we compactly denote it as $\epsilon(\lambda) = \mathsf{negl}(\lambda)$. An algorithm $\mathcal{A}$ is said to be PPT if it is modeled as a probabilistic Turing machine that runs in time polynomial in λ. If S is a set, $x \xleftarrow{\$} S$ denotes the process of selecting x uniformly at random in S (which in particular assumes that S can be sampled efficiently. If $\mathcal{A}$ is a probabilistic algorithm, $x \xleftarrow{\$} \mathcal{A}(\cdot)$ denotes the process of running $\mathcal{A}$ on some appropriate input and assigning its output to x. For a positive integer n, we denote by $[n]$ the set of integers $\{1, \ldots, n\}$.

Definition 1 (Statistical Distance). *Let X, Y be two random variables over a finite set $\mathcal{U}$. The statistical distance between X and Y is defined as*

$$\mathsf{SD}[X, Y] = \frac{1}{2} \sum_{u \in U} \big| \Pr[X = u] - \Pr[Y = u] \big|$$

2.1 Homomorphic Encryption

Here we recall the definition of homomorphic encryption. In this work we make the (somewhat canonical) assumption that the messages live in some ring $(\mathcal{M}, +, \cdot)$ while the computations are expressed as arithmetic circuits (i.e., additions, multiplications and multiplications by constants) over such ring. A homomorphic encryption scheme $\mathcal{HE}$ consists of a tuple of four PPT algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ working as follows:

$\mathsf{KeyGen}(1^\lambda)$**:** the key generation algorithm takes as input the security parameter λ and produces a secret key sk and a public key pk. The public key pk implicitly defines a message space $\mathcal{M}$.

$\mathsf{Enc}(\mathsf{pk}, m)$**:** the encryption algorithm takes as input pk and a message $m \in \mathcal{M}$, and it outputs a ciphertext C.

$\mathsf{Eval}(\mathsf{pk}, f, C_1, \ldots, C_t)$**:** the evaluation algorithm takes as input pk, an arithmetic circuit $f : \mathcal{M}^t \to \mathcal{M}$ in a class $\mathcal{F}$ of "permitted" circuits, and t ciphertexts $C_1, \ldots, C_t$. It returns a ciphertext C.

$\mathsf{Dec}(\mathsf{sk}, C)$**:** given sk and a ciphertext C, the decryption algorithm outputs a message m.

A homomorphic encryption scheme should satisfy four main properties: *correctness*, *compactness*, *security* and *circuit privacy*. The first two properties regard the functionality of the scheme while the remaining two properties model security.

The basic requirement is correctness:

Definition 2 (Correctness). *A homomorphic encryption scheme $\mathcal{HE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ is said to* correctly evaluate *a family of circuits $\mathcal{F}$ if for all honestly generated keys $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$, for all $f \in \mathcal{F}$ and for all messages $m_1, \ldots, m_t \in \mathcal{M}$ we have that if $C_i \leftarrow \mathsf{Enc}(\mathsf{pk}, m_i)$ $\forall i \in [t]$, then*

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(\mathsf{pk}, f, (C_1, \ldots, C_t))) = f(m_1, \ldots, m_t)] = 1 - \mathsf{negl}(\lambda)$$

where the probability is taken over all the algorithms' random choices.

In addition to correctness, any "interesting" homomorphic encryption scheme must be *compact.* Roughly speaking, this means that the ciphertexts output by Eval have some fixed size, which does not depend on the size of the evaluated circuit. This is formally defined as follows:

Definition 3 (Compactness). *A homomorphic encryption scheme* $\mathcal{HE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *is said to* compactly evaluate *a family of circuits* $\mathcal{F}$ *if the running time of the decryption algorithm* Dec *is bounded by a fixed polynomial in* λ*.*

The security of a homomorphic encryption scheme is defined using the notion of semantic security put forward by Goldwasser and Micali [19].

Definition 4 (Semantic Security). *Let* $\mathcal{HE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *be a (homomorphic) encryption scheme, and* $\mathcal{A}$ *be a PPT adversary. Consider the following experiment:*

```
Experiment $\mathbf{Exp}^{\mathsf{SS}}_{\mathcal{HE},\mathcal{A}}(\lambda)$
    $b \xleftarrow{\$} \{0,1\}$; $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$
    $(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk})$
    $c \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, m_b)$
    $b' \leftarrow \mathcal{A}(c)$
    If $b' = b$ return 1. Else return 0.
```

and define $\mathcal{A}$*'s advantage as* $\mathbf{Adv}^{\mathsf{SS}}_{\mathcal{HE},\mathcal{A}}(\lambda) = \Pr[\mathbf{Exp}^{\mathsf{SS}}_{\mathcal{HE},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}$*. Then we say that* $\mathcal{HE}$ *is* semantically-secure *if for any PPT algorithm* $\mathcal{A}$ *it holds* $\mathbf{Adv}^{\mathsf{SS}}_{\mathcal{HE},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$*.*

When considering homomorphic encryption, the notion of semantic security may not be sufficient to guarantee the confidentiality of the encrypted messages, and in particular of the computation's inputs. Roughly speaking, in a homomorphic encryption scheme we would like that the ciphertexts output by Eval do not reveal any information about the messages encrypted in the input ciphertexts. This property is formalized via the following notion of circuit privacy:

Definition 5 (Circuit Privacy). *We say that a homomorphic encryption scheme* $\mathcal{HE}$ *is* circuit private *for a family of circuits* $\mathcal{F}$ *if there exists a PPT simulator* Sim *and a negligible function* $\epsilon(\lambda)$ *such that the following holds. For any* $\lambda \in \mathbb{N}$*, any pair of keys* $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$*, any circuit* $f \in \mathcal{F}$*, any tuple of messages* $m_1, \ldots, m_t \in \mathcal{M}$ *and ciphertexts* $C_1, \ldots, C_t$ *such that* $\forall i \in [t]$*:* $C_i \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, m_i)$*, then it holds*

$$\mathsf{SD}[\mathsf{Eval}(\mathsf{pk}, f, C_1, \ldots, C_t),\ \mathsf{Sim}(1^\lambda, \mathsf{pk}, f(m_1, \ldots, m_t))] \leq \epsilon(\lambda)$$

Leveled Homomorphic Encryption. In this work we consider the notion of *leveled homomorphic encryption* in which the parameters of the scheme depend on the depth of the circuits that the scheme can evaluate. In particular, in our work we refer to the level as the degree of the arithmetic circuits. So, for example, a linearly-homomorphic encryption is a level-1 HE.

Definition 6 (Level-d Homomorphic Encryption). *For a positive integer* $d \in \mathbb{Z}^+$*,* $\mathcal{HE}^{(d)}$ *is a level-d homomorphic encryption scheme if* $\mathcal{HE}^{(d)}$ *compactly evaluates circuits of degree at most d and the running time of* $\mathcal{HE}^{(d)}$*'s algorithms is polynomial in the security parameter* λ*, the degree d, and (only in the case of* $\mathsf{Eval}^{(\mathsf{d})}$*) the circuit size.*

Now, for leveled homomorphic encryption schemes it is possible to consider a weaker version of circuit privacy which, roughly speaking, says that ciphertexts "at the same level" look indistinguishable. In other words, this means that circuit privacy holds with respect to a different distribution for each level. More formally:

Definition 7 (Leveled Circuit Privacy). *We say that a leveled homomorphic encryption $\mathcal{HE}^{(d)}$ satisfies* leveled circuit privacy *for a family of circuits $\mathcal{F}$ if there exists a PPT simulator* Sim *and a negligible function $\epsilon(\lambda)$ such that the following holds. For any $\lambda \in \mathbb{N}$, any pair of keys* $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$*, any circuit $f \in \mathcal{F}$ of degree $d' \leq d$, any tuple of messages $m_1, \ldots, m_t \in \mathcal{M}$ and ciphertexts $C_1, \ldots, C_t$ such that $\forall i \in [t]$: $C_i \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, m_i)$, then it holds*

$$\mathsf{SD}[\mathsf{Eval}(\mathsf{pk}, f, C_1, \ldots, C_t),\ \mathsf{Sim}(1^\lambda, \mathsf{pk}, d', f(m_1, \ldots, m_t))] \leq \epsilon(\lambda)$$

3 Public-Space Homomorphic Encryption

In this section we define the property that a homomorphic encryption scheme needs to satisfy in order to be used in our transformation presented in Section 4. We call such a scheme a *public-space homomorphic encryption*, and we formalize this notion below.

Definition 8 (public-space homomorphic encryption). *A homomorphic encryption scheme $\hat{\mathcal{HE}} = (\hat{\mathsf{KeyGen}}, \hat{\mathsf{Enc}}, \hat{\mathsf{Eval}}, \hat{\mathsf{Dec}})$ with message space $\mathcal{M}$ is said to be* public-space *if: (1) $\mathcal{M}$ is a (publicly known) finite and commutative ring with a unity, and (2) it is possible to efficiently sample uniformly distributed elements $m \in \mathcal{M}$.*

We stress that the above is a very mild requirement, and we point out that *virtually all* known number-theoretic homomorphic encryption schemes (e.g., [19,9,27,29,12,6,22]) are public-space, or can be easily adapted to be so (as we show in Appendix A for the case of [10,28,2,3]). In essence, given the current state of the art we have public-space linearly-homomorphic encryption schemes based on a variety of number-theoretic assumptions, such as p-subgroup [28], Quadratic Residuosity and its extensions [19,27,22], Composite Residuosity [29,12], DDH [10], DLin [2,16], subgroup decision [3]. Finally, we note that also the more recent lattice-based homomorphic encryption schemes (e.g., [5,4]) satisfy our notion of public-space.

4 Our Transformation

In this section we present our main construction, that is a methodology to convert a public-space linearly-homomorphic encryption scheme into a homomorphic encryption scheme supporting one multiplication. Precisely, the resulting scheme can compactly evaluate arithmetic circuits in which the number of additions of degree-2 terms is bounded by some constant (yet the number of additions of degree 1 is unbounded). At the same time, the scheme satisfies leveled circuit-privacy.

Let $\hat{\mathcal{HE}} = (\hat{\mathsf{KeyGen}}, \hat{\mathsf{Enc}}, \hat{\mathsf{Eval}}, \hat{\mathsf{Dec}})$ be a public-space linearly-homomorphic encryption scheme as per Definition 3. To ease the presentation, in our description we denote by $\hat{\mathcal{C}}$ the ciphertext space of $\hat{\mathcal{HE}}$, we use greek letters to denote elements of $\hat{\mathcal{C}}$ and roman letters for elements of $\mathcal{M}$. Without loss of generality we assume $\hat{\mathsf{Eval}}$ consists of two subroutines: one to perform (homomorphic) addition and one for performing (homomorphic) multiplication by known constants. We compactly denote these operations with $\boxplus$ and $\cdot$, respectively.[9] Namely, given two cihertexts $\beta_1, \beta_2 \in \hat{\mathcal{C}}$, $\beta = \beta_1 \boxplus \beta_2$

[9] Here we slightly abuse notation as the symbol $\cdot$ is also used to denote multiplication in the ring $\mathcal{M}$.

denotes their homomorphic addition, and $\beta = c \cdot \beta_1$ denotes a multiplication by a constant $c \in \mathcal{M}$. Addition and multiplication over $\mathcal{M}$ are denoted by $+$ and $\cdot$, respectively.

In what follows, we propose a homomorphic encryption scheme $\mathcal{HE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ capable of evaluating arithmetic circuits of degree 2 over $\mathcal{M}$, in which the number of additions of degree-2 terms is bounded by some constant L. Precisely, let $\mathcal{F}_d$ be the class of (multi-variate) polynomials of total degree d over the ring $\mathcal{M}$. Then our scheme $\mathcal{HE}$ can compactly evaluate polynomials in the class $\mathcal{F}_2^\star = \{f(\boldsymbol{m})\} \subset \mathcal{F}_2$, where $f(\boldsymbol{m})$ is a polynomial of form $P(\boldsymbol{m}) + \sum_{i=1}^{L} Q_i(\boldsymbol{m}) \cdot R_i(\boldsymbol{m})$ where $P(\boldsymbol{m}), \{Q_i(\boldsymbol{m}), R_i(\boldsymbol{m})\}$ are all polynomials in $\mathcal{F}_1$.

$\mathsf{KeyGen}(1^\lambda)$: On input 1^λ, where $\lambda \in \mathbb{N}$ is a security parameter, it runs $\widehat{\mathsf{KeyGen}}(1^\lambda)$ to get $(\mathsf{pk}, \mathsf{sk})$, and outputs $(\mathsf{pk}, \mathsf{sk})$. We assume that pk implicitly contains a description of $\hat{\mathcal{HE}}$'s message space $\mathcal{M}$ and ciphertext space $\hat{\mathcal{C}}$. The message space of the scheme $\mathcal{HE}$ will be the same $\mathcal{M}$, while the ciphertext space is discussed below.

$\mathsf{Enc}(\mathsf{pk}, m)$: Given a message $m \in \mathcal{M}$, the randomized encryption algorithm chooses a random value $b \xleftarrow{\$} \mathcal{M}$, and it sets $a \leftarrow (m - b) \in \mathcal{M}$ and $\beta \leftarrow \widehat{\mathsf{Enc}}(\mathsf{pk}, b)$. The output is $C = (a, \beta) \in \mathcal{M} \times \hat{\mathcal{C}}$.

$\mathsf{Eval}(\mathsf{pk}, f, C_1, \ldots, C_t)$: We describe this algorithm in terms of five different procedures: $(\mathsf{Add}_1, \mathsf{Mult}, \mathsf{Add}_2, \mathsf{cMult}_1, \mathsf{cMult}_2)$ that implement the basic homomorphic operations: additions, multiplications and multiplications by known constants. Informally, Add_1 and Mult operate over pairs of level-1 ciphertexts (i.e., ciphertexts that encode either "fresh" messages or messages obtained as linear combinations of fresh messages) whereas Add_2 operates over pairs of level-2 ciphertexts (i.e., ones containing "multiplied" messages). cMult_1 and cMult_2 instead operate over a single ciphertext of level 1 and 2 respectively. Therefore, homomorphically evaluating a circuit f consists of evaluating the corresponding homomorphic operations. Furthermore, in order to achieve circuit privacy, the ciphertext output by Eval must be re-randomized by using one of the procedures described later.

Add_1 : On input two level-1 ciphertexts $C_1, C_2 \in \mathcal{M} \times \hat{\mathcal{C}}$ where, for $i = 1, 2$, $C_i = (a_i, \beta_i)$, this algorithm produces a (level-1) ciphertext $C = (a, \beta) \in \mathcal{M} \times \hat{\mathcal{C}}$ computed as follows:

$$a = a_1 + a_2, \quad \beta = \beta_1 \boxplus \beta_2$$

For correctness, it is easy to see that if $a_i = (m_i - b_i)$ and $\beta_i \in \widehat{\mathsf{Enc}}(\mathsf{pk}, b_i)$ for some $b_i \in \mathcal{M}$, then $a = (m_1 + m_2) - (b_1 + b_2)$ and $\beta \in \widehat{\mathsf{Enc}}(\mathsf{pk}, b_1 + b_2)$.

Mult : On input two level-1 ciphertexts $C_1, C_2 \in \mathcal{M} \times \hat{\mathcal{C}}$ where, for $i = 1, 2$, $C_i = (a_i, \beta_i)$, this algorithm computes a level-2 ciphertext $C = (\alpha, \beta) \in \hat{\mathcal{C}} \times \hat{\mathcal{C}}^2$ as follows:

$$\alpha = \widehat{\mathsf{Enc}}(\mathsf{pk}, a_1 \cdot a_2) \boxplus a_1 \cdot \beta_2 \boxplus a_2 \cdot \beta_1$$
$$\beta = (\beta_1, \beta_2)^\top$$

For correctness, one can see that if $a_i = (m_i - b_i)$ and $\beta_i \in \widehat{\mathsf{Enc}}(\mathsf{pk}, b_i)$ for some $b_i \in \mathcal{M}$, then $\alpha \in \widehat{\mathsf{Enc}}\left(\mathsf{pk}, (m_1 m_2 - b_1 m_2 - b_2 m_1 + b_1 b_2) + (b_2 m_1 - b_1 b_2) + (b_1 m_2 - b_1 b_2)\right) = \widehat{\mathsf{Enc}}(\mathsf{pk}, m_1 m_2 - b_1 b_2)$ and $\beta \in (\widehat{\mathsf{Enc}}(\mathsf{pk}, b_1), \widehat{\mathsf{Enc}}(\mathsf{pk}, b_2))^\top$.

Add_2 : On input two level-2 ciphertexts C_1, C_2, where $\forall i = 1, 2$, $C_i = (a_i, \beta_i) \in \hat{\mathcal{C}} \times \hat{\mathcal{C}}^{2 \times \ell_i}$ such that $\beta_i = [(\beta_{1,1}^{(i)}, \beta_{2,1}^{(i)})^\top, \ldots, (\beta_{1,\ell_i}^{(i)}, \beta_{2,\ell_i}^{(i)})^\top]$, this algorithm returns a level-2 ciphertext $C = (\alpha, \beta) \in \hat{\mathcal{C}} \times \hat{\mathcal{C}}^{2 \times (\ell_1 + \ell_2)}$ computed as follows:

$$\alpha = \alpha_1 \boxplus \alpha_2, \quad \beta = [\beta_1, \beta_2]$$

For correctness, if $\alpha_i \in \hat{\mathsf{Enc}}(\mathsf{pk}, m_i - b_i)$ and β_i is a matrix of $(\beta^{(i)}_{j,k} \in \hat{\mathsf{Enc}}(\mathsf{pk}, b^{(i)}_{j,k}))_{j,k}$ such that $\sum_{k=1}^{\ell_i} b^{(i)}_{1,k} \cdot b^{(i)}_{2,k} = b_i$, then it is not hard to see that $\alpha \in \hat{\mathsf{Enc}}(\mathsf{pk}, (m_1 + m_2) - (b_1 + b_2))$ and β is a matrix of $\beta_{j,k} \in \hat{\mathsf{Enc}}(\mathsf{pk}, b_{j,k})$ such that $\sum_{k=1}^{\ell_1+\ell_2} b_{1,k} \cdot b_{2,k} = b_1 + b_2$.

cMult_1 : On input a constant $c \in \mathcal{M}$ and a level-1 ciphertext $C = (a, \beta) \in \mathcal{M} \times \hat{\mathcal{C}}$, this algorithm returns a level-1 ciphertext $C' = (c \cdot a, c \cdot \beta) \in \mathcal{M} \times \hat{\mathcal{C}}$. The correctness of this operation is straightforward.

cMult_2 : On input a constant $c \in \mathcal{M}$ and a level-2 ciphertext $C = (\alpha, \beta) \in \hat{\mathcal{C}} \times \hat{\mathcal{C}}^{2\times\ell}$ such that $\beta = [(\beta_{1,1}, \beta_{2,1})^\top, \ldots, (\beta_{1,\ell_i}, \beta_{2,\ell})^\top]$, this algorithm returns a level-2 ciphertext $C' = (\alpha', \beta') \in \hat{\mathcal{C}} \times \hat{\mathcal{C}}^{2\times\ell}$ computed as follows:

$$\begin{aligned} \alpha' &= c \cdot \alpha, \\ \beta' &= [(c \cdot \beta_{1,1}, \beta_{2,1})^\top, \ldots, (c \cdot \beta_{1,\ell_i}, \beta_{2,\ell})^\top] \end{aligned}$$

To see the correctness of ciphertexts obtained through cMult_2, if $\alpha \in \hat{\mathsf{Enc}}(\mathsf{pk}, m - b)$ and β is a matrix of $(\beta_{j,k} \in \hat{\mathsf{Enc}}(\mathsf{pk}, b_{j,k}))_{j,k}$ such that $\sum_{k=1}^{\ell} b_{1,k} \cdot b_{2,k} = b$, then it is not hard to see that $\alpha' \in \hat{\mathsf{Enc}}(\mathsf{pk}, cm - cb)$ and β' is a matrix of $\beta'_{j,k}$ such that $\beta'_{1,k} \in \hat{\mathsf{Enc}}(\mathsf{pk}, cb_{1,k})$ and $\beta'_{2,k} = \beta_{2,k} \in \hat{\mathsf{Enc}}(\mathsf{pk}, b_{2,k})$. Hence, $\sum_{k=1}^{\ell} c \cdot b_{1,k} \cdot b_{2,k} = c \cdot b$.

$\mathsf{Dec}(\mathsf{sk}, C)$**:** We distinguish two different decryption procedures according to whether the ciphertext C is of level-1 or level-2.

Level-1 Decryption: On input a level-1 ciphertext $C = (a, \beta)$ and the secret key sk, the algorithm outputs $m \leftarrow a + \hat{\mathsf{Dec}}(\mathsf{sk}, \beta)$.

Level-2 Decryption On input a level-2 ciphertext $C = (\alpha, \beta) \times \hat{\mathcal{C}} \times \hat{\mathcal{C}}^{2\times\ell}$ and the secret key sk, the algorithm outputs

$$m \leftarrow \hat{\mathsf{Dec}}(\alpha) + \left(\sum_{i=1}^{\ell} \hat{\mathsf{Dec}}(\mathsf{sk}, \beta_{1,i}) \cdot \hat{\mathsf{Dec}}(\mathsf{sk}, \beta_{2,i}) \right)$$

Before concluding the description of the scheme, we describe how to perform ciphertext re-randomization. Namely, we give a procedure ReRand that takes as input a (level-1 or level-2) ciphertext encrypting some message m, using random pad b, and outputs a new encryption of m padded with a *fresh* random b'. We stress that such re-randomization is crucial to achieve circuit privacy which would not be possible by using only the re-randomization of the underlying linearly-homomorphic scheme. As above, we describe this procedure in terms of two different subroutines ReRand_1 and ReRand_2 which operate over level-1 and level-2 ciphertexts respectively.

$\mathsf{ReRand}_1(\mathsf{pk}, C)$**:** On input a level-1 ciphertext $C = (a, \beta)$, it chooses a random $b' \xleftarrow{\$} \mathcal{M}$ and outputs $C' = (a', \beta')$ computed as follows

$$a' = a - b', \quad \beta' = \hat{\mathsf{Enc}}(\mathsf{pk}, b') \boxplus \beta$$

$\mathsf{ReRand}_2(\mathsf{pk}, C)$**.** On input a level-2 ciphertext $C = (\alpha, \beta) \in \hat{\mathcal{C}} \times \hat{\mathcal{C}}^{2\times\ell}$ the algorithm chooses random $\tilde{b} \xleftarrow{\$} \mathcal{M}^{2\times\ell}$ and outputs $C' = (\alpha', \beta')$ computed as follows

1. For $i = 1$ to ℓ, compute

$$\gamma_i = \hat{\mathsf{Enc}}\left(\mathsf{pk}, -(\tilde{b}_{1,i} \cdot \tilde{b}_{2,i})\right) \boxplus (-\tilde{b}_{2,i}) \cdot \beta_{1,i} \boxplus (-\tilde{b}_{1,i}) \cdot \beta_{2,i}$$

2. Set $\alpha' \leftarrow \alpha + \sum_{i=1}^{\ell} \gamma_i \in \hat{\mathcal{C}}$
3. For $i = 1$ to ℓ and $j = 1, 2$, set $\beta'_{j,i} \leftarrow \beta_{j,i} \boxplus \hat{\mathsf{Enc}}(\mathsf{pk}, \tilde{b}_{j,i})$.

To see correctness, first assume that $\alpha \in \hat{\mathsf{Enc}}(\mathsf{pk}, m - b)$ with $b = \sum_{i=1}^{\ell} b_{1,i} \cdot b_{2,i}$ and $\beta_{j,i} \in \hat{\mathsf{Enc}}(\mathsf{pk}, b_{j,i})$. Then we have $\gamma_i \in \hat{\mathsf{Enc}}\left(\mathsf{pk}, -(\tilde{b}_{1,i} \cdot \tilde{b}_{2,i} + \tilde{b}_{1,i} \cdot b_{2,i} + b_{1,i} \cdot \tilde{b}_{2,i})\right)$, $\beta'_{j,i} \in \hat{\mathsf{Enc}}(\mathsf{pk}, b'_{j,i})$ with $b'_{j,i} = b_{j,i} + \tilde{b}_{j,i}$. Thus we can write $\alpha' \in \hat{\mathsf{Enc}}(\mathsf{pk}, m - b')$ where

$$\begin{aligned} b' &= b + \sum_{i=1}^{\ell} \tilde{b}_{1,i} \cdot \tilde{b}_{2,i} + \tilde{b}_{1,i} \cdot b_{2,i} + b_{1,i} \cdot \tilde{b}_{2,i} \\ &= \sum_{i=1}^{\ell} b_{1,i} \cdot b_{2,i} + \tilde{b}_{1,i} \cdot \tilde{b}_{2,i} + \tilde{b}_{1,i} \cdot b_{2,i} + b_{1,i} \cdot \tilde{b}_{2,i} \\ &= \sum_{i=1}^{\ell} (b_{1,i} + \tilde{b}_{1,i}) \cdot (b_{2,i} + \tilde{b}_{2,i}) = \sum_{i=1}^{\ell} b'_{1,i} \cdot b'_{2,i} \end{aligned}$$

Therefore, α is a valid encryption of m under a completely fresh pad $(b'_{j,i}) \in \mathcal{M}^{2\times\ell}$.

The correctness of the scheme $\mathcal{HE}$ follows from the correctness of $\hat{\mathcal{HE}}$, the fact that its message space $\mathcal{M}$ is a finite ring, and by following the observations made along our description. Second, it is easy to see that the only "extra" property we require to the underlying $\hat{\mathcal{HE}}$ is the ability to sample random elements in $\mathcal{M}$.

Proof of Security. First, we prove that the scheme $\mathcal{HE}$ is semantically-secure.

Theorem 1. *If $\hat{\mathcal{HE}}$ is semantically-secure, then $\mathcal{HE}$ is semantically secure.*

The proof is straightforward:

$$(m_0 - b, \hat{\mathsf{Enc}}(\mathsf{pk}, b)) \approx (m_0 - b, \hat{\mathsf{Enc}}(\mathsf{pk}, 0)) \equiv (m_1 - b, \hat{\mathsf{Enc}}(\mathsf{pk}, 0)) \approx (m_1 - b, \hat{\mathsf{Enc}}(\mathsf{pk}, b))$$

where $\approx$ denotes computational indistinguishability by the semantic security of $\hat{\mathcal{HE}}$ and $\equiv$ means that the distributions are identical.

Second, we show that $\mathcal{HE}$ satisfies the notion of leveled circuit-privacy given in Definition 7.

Theorem 2. *If $\hat{\mathcal{HE}}$ is circuit private, then $\mathcal{HE}$ is a leveled circuit-private homomorphic encryption.*

Proof. The simulator $\mathsf{Sim}(1^\lambda, \mathsf{pk}, d, f(m_1, \ldots, m_t))$ is very simple: intuitively, it needs to create a fresh encryption of the message $m = f(m_1, \ldots, m_t)$ at level $d = 1$ or 2. More precisely, let $\hat{\mathsf{Sim}}$ be the simulator for the circuit privacy of $\hat{\mathcal{HE}}$. Then, if $d = 1$, Sim picks a random $b \xleftarrow{\$} \mathcal{M}$ and outputs $C = (m - b, \hat{\mathsf{Sim}}(1^\lambda, \mathsf{pk}, b))$. If $d = 2$, Sim picks random $B \xleftarrow{\$} \mathcal{M}^{2\times\ell}$, set $b = \sum_{i=1}^{\ell} B_{1,i} \cdot B_{2,i}$ and outputs $C = (\alpha, \beta)$ where $\alpha = \hat{\mathsf{Sim}}(1^\lambda, \mathsf{pk}, m - b)$ and $\beta = [\beta_{j,i} = \hat{\mathsf{Sim}}(1^\lambda, \mathsf{pk}, B_{j,i})]_{j,i}$ for $j = 1, 2$ and $i \in [\ell]$. Now, if we compare the distribution of the ciphertexts created by Sim and the one of those generated by Eval (where re-randomization is run at the end), then one can immediately see that these distributions will be indistinguishable under the assumption that $\hat{\mathcal{HE}}$ is circuit private.

We remark that in the above proof we assumed that Sim gets to know the number ℓ of additions of degree-2 terms that are performed by the circuit f. While this information is not considered as a Sim's input in Definition 7, one can either think of ℓ as some trivial information that one does not want to hide or, otherwise, it is possible to easily change our scheme to avoid this issue. Namely, assume that L is the constant which bounds such number of additions, then one can "pad" any level-2 ciphertext with more encryptions in the β component up to having L pairs of ciphertexts. Namely, one first adds to β the columns $[(\beta_{1,\ell+1}, \beta_{2,\ell+1})^\top, \ldots, (\beta_{1,L}, \beta_{2,L})^\top]$ where, for $i = \ell + 1, \ldots, L$ and $j = 1, 2$, each $\beta_{j,i} = \hat{\mathsf{Enc}}(\mathsf{pk}, b_{j,i})$ for random $b_{j,i} \xleftarrow{\$} \mathcal{M}$. Second, one adds the new random pad to α homomorphically, i.e., computes $\gamma = \sum_{i=\ell+1}^{L} b_{1,i} \cdot b_{2,i}$ and $\alpha' \leftarrow \alpha \boxplus \hat{\mathsf{Enc}}(\mathsf{pk}, -\gamma)$.

On the ciphertext growth. As we mentioned earlier, our scheme cannot compactly evaluate all degree-2 polynomials. Yet we achieve compactness as long as the polynomials are in $\mathcal{F}_2^\star$, i.e., are of the form $P(\boldsymbol{m}) + \sum_{i=1}^{L} Q_i(\boldsymbol{m}) \cdot R_i(\boldsymbol{m})$, with $P(\boldsymbol{m}), \{Q_i(\boldsymbol{m}), R_i(\boldsymbol{m})\}$ all of degree 1. Such a class of polynomials is still meaningful. For example, the SPDZ protocol [13], in its offline phase, requires a somewhat homomorphic encryption capable of evaluating polynomials exactly of the form $P(\boldsymbol{m}) + Q(\boldsymbol{m}) \cdot R(\boldsymbol{m})$, which is a subset (for $L = 1$) of our supported class $\mathcal{F}_2^\star$.

Furthermore, as a second contribution of this work, we show in Section 5 how to completely remove this issue in a specific application (thus achieving fully compact ciphertexts while supporting *all* degree-2 polynomials).

Inheriting Properties. Interestingly, our transformation naturally preserves useful properties of the underlying linearly-homomorphic scheme. Here we highlight three properties: proof of knowledge, threshold decryption and multikey homomorphic encryption.

First, we show that if the underlying linearly-homomorphic scheme admits a *proof of plaintext knowledge*, then so does the transformed scheme.

Theorem 3. *If $\hat{\mathcal{HE}}$ admits a zero knowledge proof of plaintext knowledge protocol Σ, then so does $\mathcal{HE}$ (for level-1 ciphertexts).*

Proof. Assume that that a prover P wants to prove knowledge of a plaintext m corresponding to a level-1 ciphertext (a, β). To do this the prover simply uses Σ to prove knowledge of the random pad b encrypted in β. Notice that a prover knowing m also knows such a value b. Completeness and Honest Verifier Zero Knowledge immediately follow from the (corresponding) properties of Σ. Moreover, since Σ is a proof of knowledge, there exists an efficient extractor E that, given two different accepting conversations sharing a common initial message, provides the plaintext. Thus we can build an extractor E' out of E by simply running E on the two different accepting transcripts as above. Once E outputs b, E' outputs $m \leftarrow a + b$. [10]

As a second property, we consider *multikey homomorphic encryption.* This is a scheme which is capable of performing homomorphic operations on inputs encrypted under multiple unrelated keys, and that allows decryption only if, roughly speaking, all the corresponding secret keys involved in the computation are used. This notion has been formally proposed in the context of multiparty computation by López-Alt *et al.* [25] who also proposed a fully-homomorphic realization. We note that a multikey linearly-homomorphic encryption scheme was earlier given by Bresson *et al.* [6] (albeit not explicitly formalized). The following theorem shows that the schemes obtained via our transformation inherit the multikey property. Therefore, by plugging for instance the scheme of Bresson *et al.* [6] we obtain a multikey HE scheme supporting one multiplication.

[10] Notice that E might work in expected polynomial time but this only means that a similar constraint applies to E' as well.

Theorem 4. *If $\hat{\mathcal{HE}}$ is a* multikey *linearly-homomorphic encryption, then so is $\mathcal{HE}$.*

Proof (Sketch). Informally, this holds because a level-1 $\mathcal{HE}$ ciphertext consists of a $\hat{\mathcal{HE}}$ ciphertext together with a ring element which is unrelated to any key. Similarly, level-2 ciphertexts consist solely of $\hat{\mathcal{HE}}$ ciphertexts.

As a third property inherited by our transformed schemes, we consider *threshold decryption.*

Theorem 5. *If $\hat{\mathcal{HE}}$ has a threshold decryption protocol associated with it, then $\mathcal{HE}$ has threshold decryption as well.*

The theorem follows by observing that the threshold variant of $\mathcal{HE}$ is basically the same threshold variant of $\hat{\mathcal{HE}}$.

Extensions. In Appendix B we show how to generalize our construction to boost a level-d homomorphic encryption scheme to compute multiplications up to degree $2d$. As an interesting result, such a generalization for instance shows how to use the BGN cryptosystem [3] to evaluate polynomials of degree up to 4 on ciphertexts.

Instantiations. By instantiating our transformation with a variety of existing schemes (e.g., [19,9,10,28,27,29,12,6,2,22]) we obtain several HE schemes capable of evaluating one multiplication whose security relies on a variety of assumptions, such as standard DDH, quadratic/composite residuosity, or Decision Linear. Furthermore, when considering our extension applied to BGN, we obtain a homomorphic encryption scheme that supports computations of degree up to 4, and whose security relies on a number-theoretic assumption.

5 Two-Server Delegation of Computation on Encrypted Data

In this section we introduce a new primitive that we call *two-server delegation of computation on encrypted data* (2S-DCED, for short) and we show how to realize it building on our technique of Section 4. Using a 2S-DCED protocol, a client can outsource the computation of a function f on some input m to two distinct servers in such a way that: it is possible to recover the output $f(m)$ from the servers' outputs, and (2) the two servers do *not* need to interact (nor to know of their mutual existence). For security, the protocol preserves the confidentiality of the client's inputs as long as the two servers do not collude.

This new notion is somehow related to the one of server-aided secure function evaluation [23]. Indeed, in Section 5.4 we show how to use two-server delegation of computation on encrypted data to build double-server-aided secure function evaluation in the semi-honest model.

5.1 2S-DCED Definition

A protocol for two-server delegation of computation on encrypted data consists of a tuple of algorithms $\mathsf{2S.DCED} = (\mathsf{2S.KeyGen}, \mathsf{2S.Enc}, \mathsf{2S.Dec}, \mathsf{2S.Eval}_1, \mathsf{2S.Eval}_2)$ working as follows.

$\mathsf{2S.KeyGen}(1^\lambda)$**:** the key generation algorithm takes as input the security parameter λ and produces a secret key sk and a public key pk.

$\mathsf{2S.Enc}(\mathsf{pk}, m)$**:** the encryption algorithm takes as input pk and a message $m \in \mathcal{M}$ and outputs a ciphertext C consisting of two components $(C^{(1)}, C^{(2)})$.

$\mathsf{2S.Dec}(\mathsf{sk}, C^{(1)}, C^{(2)})$**:** the decryption algorithm takes as input sk and a ciphertext $(C^{(1)}, C^{(2)})$ and returns a message m.

In addition, there exist two evaluation algorithms $\mathsf{2S.Eval}_1, \mathsf{2S.Eval}_2$ (basically, one for each server):

$\mathsf{2S.Eval}_i(\mathsf{pk}, f, C_1^{(i)}, \ldots, C_t^{(i)})$**:** the i-th evaluation algorithm takes as input pk, an arithmetic circuit $f : \mathcal{M}^t \to \mathcal{M}$ in a class $\mathcal{F}$ of "permitted" circuits, and t ciphertexts $C_1^{(i)}, \ldots, C_t^{(i)}$, all of the same component $i = 1, 2$. The algorithm outputs a ciphertext $C^{(i)}$ (of the same component).

As a basic property, a 2S-DCED protocol must be correct and compact.

Definition 9 (Correctness and Compactness of 2S-DCED). *A 2S-DCED protocol* $\mathsf{2S.DCED} = (\mathsf{2S.KeyGen}, \mathsf{2S.Enc}, \mathsf{2S.Eval}_1, \mathsf{2S.Eval}_2, \mathsf{2S.Dec})$ correctly evaluates *a family of circuits $\mathcal{F}$ if for all honestly generated keys* $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{2S.KeyGen}(1^\lambda)$*, for all $f \in \mathcal{F}$ and for all messages $m_1, \ldots, m_t \in \mathcal{M}$ we have that if* $(C_i^{(1)}, C_i^{(2)}) \leftarrow \mathsf{2S.Enc}(\mathsf{pk}, m_i)$ $\forall i \in [t]$*, then*

$$\Pr[\mathsf{2S.Dec}(\mathsf{sk}, \mathsf{2S.Eval}_1(\mathsf{pk}, f, C_1^{(1)}, \ldots, C_t^{(1)}), \mathsf{2S.Eval}_2(\mathsf{pk}, f, C_1^{(2)}, \ldots, C_t^{(2)})) = f(m_1, \ldots, m_t)] = 1 - \mathsf{negl}(\lambda)$$

where the probability is taken over all the algorithms' random choices.

Furthermore, $\mathsf{2S.DCED}$ compactly evaluates $\mathcal{F}$ *if above the running time of the decryption algorithm* $\mathsf{2S.Dec}$ *is bounded by a fixed polynomial in λ, independent of f.*

Security. Informally, a 2S-DCED protocol should guarantee that any adversary who has access to only one component of a ciphertext $(C^{(1)}, C^{(2)})$ should not learn any information about the underlying plaintext. We formalize this property using the approach of semantic security. Intuitively, our notion says that as long as the two servers do not collude, each of them does not learn anything about the encrypted messages.

Definition 10 (2S-DCED Semantic Security). *Let* $\mathsf{2S.DCED}$ *be a 2S-DCED protocol as defined above, and $\mathcal{A}$ be a PPT adversary. Consider the following experiment:*

```
Experiment $\mathbf{Exp}^{\mathsf{2S.SS}}_{\mathsf{2S.DCED},\mathcal{A}}(\lambda)$
    $b \xleftarrow{\$} \{0,1\}$; $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{2S.KeyGen}(1^\lambda)$
    $(m_0, m_1, i) \leftarrow \mathcal{A}(\mathsf{pk})$
    $(C^{(1)}, C^{(2)}) \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, m_b)$
    $b' \leftarrow \mathcal{A}(C^{(i)})$
    If $b' = b$ return 1. Else return 0.
```

and define $\mathcal{A}$'s advantage as $\mathbf{Adv}^{\mathsf{2S.SS}}_{\mathsf{2S.DCED},\mathcal{A}}(\lambda) = \Pr[\mathbf{Exp}^{\mathsf{2S.SS}}_{\mathsf{2S.DCED},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}$. *We say that* $\mathsf{2S.DCED}$ *is* semantically-secure *if for any PPT $\mathcal{A}$ it holds* $\mathbf{Adv}^{\mathsf{2S.SS}}_{\mathsf{2S.DCED},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$.

We remark that the notion extends in a straightforward way to the case in which the adversary submits *multiple* triples $\{(m_{0,j}, m_{1,j}, i_j)\}$ and receives the corresponding ciphertext components $\{C_j^{(i_j)}\}$ (all generated using the same bit b).

In addition to semantic security we consider another security notion that we call *context hiding*. The motivation is that in the outsourcing setting the party who decrypts may be different from the one who provides the inputs of the computation. Hence, the decryptor who receives a ciphertext $(C^{(1)}, C^{(2)})$ encrypting the result of a computation f must learn nothing about all the inputs of f (that it did not provide), beyond what the result trivially reveals, i.e., $f(m_1, \ldots, m_t)$.

Definition 11 (Context Hiding). *We say that a protocol* $\mathsf{2S.DCED}$ *for two-server delegation of computation on encrypted data satisfies* context-hiding *for a family of circuits* $\mathcal{F}$ *if there exists a PPT simulator* Sim *and a negligible function* $\epsilon(\lambda)$ *such that the following holds. For any* $\lambda \in \mathbb{N}$, *any pair of keys* $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{2S.KeyGen}(1^\lambda)$, *any circuit* $f \in \mathcal{F}$ *with* t *inputs, any two tuples of messages* $\boldsymbol{m}_1 \in \mathcal{M}^{t_1}, \boldsymbol{m}_2 \in \mathcal{M}^{t_2}$ *such that* $t = t_1 + t_2$ *and corresponding ciphertexts* $\boldsymbol{C}_1, \boldsymbol{C}_2$ *such that* $\boldsymbol{C}_k = (\boldsymbol{C^{(1)}}_k, \boldsymbol{C^{(2)}}_k) \xleftarrow{\$} \mathsf{2S.Enc}(\mathsf{pk}, \boldsymbol{m}_k)$ *for* $k = 1, 2$, *and a ciphertext* $(C^{(1)}, C^{(2)})$ *where* $C^{(1)} = \mathsf{2S.Eval}_1(\mathsf{pk}, f, \boldsymbol{C^{(1)}}_1, \boldsymbol{C^{(1)}}_2)$ *and* $C^{(2)} = \mathsf{2S.Eval}_2(\mathsf{pk}, f, \boldsymbol{C^{(2)}}_1, \boldsymbol{C^{(2)}}_2)$ *it holds*

$$\mathsf{SD}[(C^{(1)}, C^{(2)}),\ \mathsf{Sim}(1^\lambda, \boldsymbol{C}_1, f, \mathsf{pk}, f(\boldsymbol{m}_1, \boldsymbol{m}_2))] \leq \epsilon(\lambda)$$

5.2 A 2S-DCED Protocol for Degree-2 Polynomials from Public-Space Linearly-Homomorphic Encryption

In this section, we propose the construction of a protocol for two-server delegation of computation on encrypted data that supports the evaluation of *all* degree-2 multivariate polynomials. Our construction builds upon a public-space linearly-homomorphic encryption scheme (cf. Definition 8), and its interesting feature is to "boost" the linear-only homomorphism in order to compute functions up to degree 2. We stress that in contrast to the result of Section 4, here the ciphertext remains always compact and thus the scheme can support the evaluation of *all* degree-2 polynomials.

Furthermore, our protocol achieves two interesting properties that we call *unbalanced efficiency* and *unbalanced security*. The former says that one of the two servers can perform the computation essentially "for free", i.e., as if it computes over the original inputs in $\mathcal{M}$. The unbalanced security property instead says that with respect to such efficient server the semantic security of the protocol holds information-theoretically.

The precise description of our scheme follows:

$\mathsf{2S.KeyGen}(1^\lambda)$**:** On input 1^λ, where λ is a security parameter, it runs $\hat{\mathsf{KeyGen}}(1^\lambda)$ to get $(\mathsf{pk}, \mathsf{sk})$, and outputs $(\mathsf{pk}, \mathsf{sk})$. We assume that pk implicitly contains a description of the message space $\mathcal{M}$ and the ciphertext space $\hat{\mathcal{C}}$.

$\mathsf{2S.Enc}(\mathsf{pk}, m)$**:** The randomized encryption algorithm chooses a random value $b \xleftarrow{\$} \mathcal{M}$ and sets $a \leftarrow (m - b) \in \mathcal{M}$ and $\beta \leftarrow \hat{\mathsf{Enc}}(\mathsf{pk}, b)$. The output is $C^{(1)} = (a, \beta) \in \mathcal{M} \times \hat{\mathcal{C}}$ and $C^{(2)} = b \in \mathcal{M}$. The ciphertexts of the first component $C^{(1)}$ are "leveled", i.e., the ones of the form $(a, \beta) \in \mathcal{M} \times \hat{\mathcal{C}}$ are of level 1, whereas $C^{(1)} = \alpha \in \hat{\mathcal{C}}$ are of level 2.

$\mathsf{2S.Eval}_1(\mathsf{pk}, f, C_1^{(1)}, \ldots, C_t^{(1)})$**:** the evaluation algorithm for the ciphertexts of the first component essentially consists of the basic procedures for performing the homomorphic operations: $\mathsf{Add}_1, \mathsf{Mult}, \mathsf{Add}_2$. Informally, Add_1 and Mult operate over pairs of level-1 ciphertexts whereas Add_2 operates over pairs of level-2 ciphertexts.

Add_1 : On input two level-1 ciphertexts $C_1^{(1)}, C_2^{(1)} \in \mathcal{M} \times \hat{\mathcal{C}}$ where, for $i = 1, 2$, $C_i^{(1)} = (a_i, \beta_i)$ this algorithm produces a (level-1) ciphertext $C = (a, \beta) \in \mathcal{M} \times \hat{\mathcal{C}}$ computed as:

$$a = a_1 + a_2, \qquad \beta = \beta_1 \boxplus \beta_2$$

Mult : On input two level-1 ciphertexts $C_1^{(1)}, C_2^{(1)} \in \mathcal{M} \times \hat{\mathcal{C}}$ this algorithm outputs a level-2 ciphertext $C^{(1)} = \alpha \in \hat{\mathcal{C}}$ computed as:

$$\alpha = \hat{\mathsf{Enc}}(\mathsf{pk}, a_1 \cdot a_2) \boxplus a_1 \cdot \beta_2 \boxplus a_2 \cdot \beta_1$$

Add_2 : On input two level-2 ciphertexts $C_1^{(1)} = \alpha_1, C_2^{(1)} = \alpha_2 \in \hat{\mathcal{C}}$ this algorithm returns a level-2 ciphertext $C^{(1)} = \alpha \in \hat{\mathcal{C}}$ computed as

$$\alpha = \alpha_1 \boxplus \alpha_2$$

Finally, note that given a ciphertext $C^{(1)}$, a multiplication by a known constants $c \in \mathcal{M}$ is achieved straightforwardly: if $C = (a, \beta)$, simply return $(c \cdot a, c \cdot \beta)$; if $C = \alpha$ return $c \cdot \alpha$.

$\mathsf{2S.Eval}_2(\mathsf{pk}, f, C_1^{(2)}, \ldots, C_t^{(2)})$**:** let $C_i^{(2)} = b_i \in \mathcal{M}$ for all $i \in [t]$. The algorithm simply executes f over the b_i's. Namely, output $C^{(2)} = b = f(b_1, \ldots, b_t) \in \mathcal{M}$.

$\mathsf{2S.Dec}(\mathsf{sk}, C^{(1)}, C^{(2)})$**:** the decryption proceeds slightly differently according to whether the ciphertext component $C^{(1)}$ is of level-1 or level-2.
Given $C^{(1)} = (a, \beta)$ and $C^{(2)} \in \mathcal{M}$, output $m \leftarrow a + C^{(2)}$.
Given $C^{(1)} = \alpha \in \hat{\mathcal{C}}$ and $C^{(2)} \in \mathcal{M}$, output $m \leftarrow \hat{\mathsf{Dec}}(\mathsf{sk}, \alpha) + C^{(2)}$.
We notice that the β component of $C^{(1)}$ is not necessary for decryption. That is, in order to save bandwidth one may require the first server not to transmit β.

Proof of Security. First, we prove that our protocol is semantically-secure according to our Definition 10.

Theorem 6. *If $\hat{\mathcal{HE}}$ is semantically-secure, then* $\mathsf{2S.DCED}$ *is a semantically-secure protocol for two-server delegation of computation on encrypted data.*

The proof is rather straightforward. We only provide a sketch. First, in the case of adversaries $\mathcal{A}$ who submit a triple $(m_0, m_1, 2)$ it is clear that $C^{(2)}$ is uniformly distributed in $\mathcal{M}$ and perfectly hides the bit b. Second, for an adversary who submits a triple $(m_0, m_1, 1)$, the security follows from the semantic security of the linearly-homomorphic scheme $\hat{\mathcal{HE}}$ in the same way as in Theorem 1.

To continue, we show that the protocol $\mathsf{2S.DCED}$ satisfies context-hiding as per Definition 11.

Theorem 7. *If $\hat{\mathcal{HE}}$ is circuit-private, then* $\mathsf{2S.DCED}$ *is context-hiding.*

Proof. Let $\hat{\mathsf{Sim}}$ be the simulator for the circuit privacy of $\hat{\mathcal{HE}}$. For an arithmetic circuit f of degree 1 the simulator $\mathsf{Sim}(1^\lambda, \boldsymbol{C}_1, f, \mathsf{pk}, m = f(\boldsymbol{m}_1, \boldsymbol{m}_2))$ can simply output $\widehat{C^{(1)}} = (m - b, \hat{\mathsf{Sim}}(1^\lambda, \mathsf{pk}, b))$ and $\widehat{C^{(2)}} = b$ where $b = f(\boldsymbol{b}_1, \boldsymbol{b}_2)$ for a randomly sampled $\boldsymbol{b}_2 \in \mathcal{M}^{t_2}$ and for $\boldsymbol{b}_1 = \boldsymbol{C^{(2)}}_1$. If f is of degree 2, the simulator does the same except that $\widehat{C^{(1)}} = \hat{\mathsf{Sim}}(1^\lambda, \mathsf{pk}, m - b)$. It is straightforward to see that by the circuit privacy of $\hat{\mathcal{HE}}$ $(\widehat{C^{(1)}}, \widehat{C^{(2)}})$ is indistinguishable from the pair $(C^{(1)}, C^{(2)})$ produced by the algorithms $\mathsf{2S.Eval}_1$ and $\mathsf{2S.Eval}_2$ respectively.

Unbalanced Efficiency and Unbalanced Security. Our 2S-DCED protocol described above achieves two interesting properties. The first one, that we call *unbalanced efficiency*, says, very roughly, that one of the two servers needs to invest much fewer computational resources. More in detail, the second server – the one executing $\mathsf{2S.Eval}_2$ – can run much faster than the first server since it does not have to do any public key operation. Essentially, it can perform as if it computes over the messages in the plain. Moreover, the storage overhead at such second server, i.e., the ratio $|C^{(2)}|/|m|$, is basically null. This property is particularly relevant in cloud scenarios in which clients have to pay for the servers' storage space and CPU cycles. In our solution the cost to pay to the second server is indeed significantly smaller.

The second property achieved by our construction is called *unbalanced security*, and it says that the semantic security of the protocol holds information-theoretically with respect to the second server[11], the same one who can run faster.

Comparison with other possible solutions. We note that by using a linear secret sharing scheme it is possible to construct a 2S-DCED protocol which uses only two servers, though supporting only degree-1 computations. To support degree-2 computations using secret sharing, one would need at least three distinct non-colluding servers (see e.g., [11]). On the other hand, a solution (with succinct communication) based on only one server can be achieved using a somewhat homomorphic encryption scheme supporting one multiplication. In contrast, our solution can achieve the same result with only two servers and by using only a linearly-homomorphic encryption scheme.

5.3 A 2S-DCED Protocol for Degree-3 Polynomials

In this section, we show a construction of a 2S-DCED protocol that uses a public-space level-2 homomorphic encryption as underlying building block, in order to support the evaluation of all multivariate polynomials of degree 3.

In what follows, $\hat{\mathcal{HE}} = (\hat{\mathsf{KeyGen}}, \hat{\mathsf{Enc}}, \hat{\mathsf{Eval}}, \hat{\mathsf{Dec}})$ denotes a public-space level-2 HE scheme. For compact notation, we indicate with $\boxplus, \boxtimes$ and $\cdot$ the procedures used by $\hat{\mathsf{Eval}}$ to perform, respectively, (homomorphic) additions, multiplications and multiplications by constants. Note that a possible instantiation of $\hat{\mathcal{HE}}$ is the BGN encryption scheme (with the small adaptation that we show in Appendix A.1).

$\mathsf{2S.KeyGen}^{\mathsf{deg3}}(1^\lambda)$**:** On input 1^λ, where λ is a security parameter, it runs $\hat{\mathsf{KeyGen}}(1^\lambda)$ to get $(\mathsf{pk}, \mathsf{sk})$, and outputs $(\mathsf{pk}, \mathsf{sk})$. We assume that pk implicitly contains a description of the message space $\mathcal{M}$ and the ciphertext space $\hat{\mathcal{C}}$.

$\mathsf{2S.Enc}^{\mathsf{deg3}}(\mathsf{pk}, m)$**:** The randomized encryption algorithm chooses a random value $b \xleftarrow{\$} \mathcal{M}$ and sets $a \leftarrow (m - b) \in \mathcal{M}$ and $\beta \leftarrow \hat{\mathsf{Enc}}(\mathsf{pk}, b)$. The output is $C^{(1)} = (a, \beta) \in \mathcal{M} \times \hat{\mathcal{C}}$ and $C^{(2)} = b \in \mathcal{M}$.

$\mathsf{2S.Eval}^{\mathsf{deg3}}{}_1(\mathsf{pk}, f, C_1^{(1)}, \ldots, C_t^{(1)})$**:** the evaluation algorithm for the ciphertext of the first component essentially consists of the basic procedures for performing the homomorphic operations: $\mathsf{Add}_1, \mathsf{Mult}_1, \mathsf{Add}_2, \mathsf{Mult}_2, \mathsf{Add}_3$. Informally, Add_1 and Mult_1 operate over level-1 ciphertexts whereas Add_2 operates over level-2 ciphertexts (i.e., ones containing "multiplied" messages). Mult_2 allows to multiply a level-1 ciphertext with a level-2 one to produce a ciphertext of level 3. Finally, Add_3 allows to make additions over level-3 ciphetexts (i.e., ones produced either by Mult_2 or by Add_3 itself).

Add_1 : On input two level-1 ciphertexts $C_1^{(1)}, C_2^{(1)} \in \mathcal{M} \times \hat{\mathcal{C}}$ where, for $i = 1, 2$, $C_i^{(1)} = (a_i, \beta_i)$ this algorithm produces a (level-1) ciphertext $C = (a, \beta) \in \mathcal{M} \times \hat{\mathcal{C}}$ computed as follows:

$$a = a_1 + a_2, \qquad \beta = \beta_1 \boxplus \beta_2$$

Mult_1 : On input two level-1 ciphertexts $C_1^{(1)}, C_2^{(1)} \in \mathcal{M} \times \hat{\mathcal{C}}$ this algorithm computes a level-2 ciphertext $C^{(1)} = (\alpha, \Gamma) \in \hat{\mathcal{C}} \times \hat{\mathcal{C}}$ as follows:

$$\alpha = \hat{\mathsf{Enc}}(\mathsf{pk}, a_1 \cdot a_2) \boxplus (a_1 \cdot \beta_2) \boxplus (a_2 \cdot \beta_1)$$
$$\Gamma = \beta_1 \boxtimes \beta_2$$

[11] With respect to the first server the security still holds in a computational sense.

Add_2 : On input a pair of level-2 ciphertexts $C_1^{(1)} = (\alpha_1, \Gamma_1), C_2^{(1)} = (\alpha_2, \Gamma_2) \in \hat{\mathcal{C}} \times \hat{\mathcal{C}}$ this algorithm returns a level-2 ciphertext $C^{(1)} = (\alpha, \Gamma) \in \hat{\mathcal{C}} \times \hat{\mathcal{C}}$ computed as

$$\alpha = \alpha_1 \boxplus \alpha_2, \quad \Gamma = \Gamma_1 \boxplus \Gamma_2$$

Notice that adding level-2 ciphertexts with level-1 ciphertexts can be easily achieved by first "moving" the latter up to level 2, which in turn can be done by performing a multiplication by a level-1 encryption of the unity $1 \in \mathcal{M}$.

Mult_2 : On input a level-1 ciphertext $C_1^{(1)} = (a_1, \beta_1) \in \mathcal{M} \times \hat{\mathcal{C}}$ and a level-2 ciphertext $C_2^{(1)} = (\alpha_2, \Gamma_2) \in \hat{\mathcal{C}} \times \hat{\mathcal{C}}$, this algorithm returns a level-3 ciphertext $C^{(1)} = \Delta \in \hat{\mathcal{C}}$ computed as

$$\Delta = (a_1 \cdot \alpha_2) \boxplus (a_1 \cdot \Gamma_2) \boxplus (\alpha_2 \boxtimes \beta_1)$$

Add_3 : On input a pair of level-3 ciphertexts $C_1^{(1)} = \Delta_1 \in \hat{\mathcal{C}}$ and $C_2^{(1)} = \Delta_2 \in \hat{\mathcal{C}}$ it outputs a level-3 ciphertext $C^{(1)} = \Delta$, computed as

$$\Delta = \Delta_1 \boxplus \Delta_2$$

$\mathsf{2S.Eval}^{\mathsf{deg3}}{}_2(\mathsf{pk}, f, C_1^{(2)}, \ldots, C_t^{(2)})$**:** simply output $C^{(2)} = f(C_1^{(2)}, \ldots, C_t^{(2)}) \in \mathcal{M}$.

$\mathsf{2S.Dec}^{\mathsf{deg3}}(\mathsf{sk}, C^{(1)}, C^{(2)})$ We distinguish three different decryption procedures according to whether the ciphertext component $C^{(1)}$ is of level 1, 2 or 3.

Level-1 Decryption: On input a level-1 ciphertext $C^{(1)} = (a, \beta)$ and $C^{(2)} \in \mathcal{M}$, simply output $m \leftarrow a + C^{(2)}$.

Level-2 Decryption: On input a level-2 ciphertext ($C^{(1)} = (\alpha, \Gamma) \in \hat{\mathcal{C}} \times \hat{\mathcal{C}}$, $C^{(2)} \in \mathcal{M}$) and the secret key sk, the algorithm outputs

$$m \leftarrow \hat{\mathsf{Dec}}(\alpha) + C^{(2)}.$$

Level-3 Decryption: On input two level-3 ciphertexts $C^{(1)} = \Delta \in \hat{\mathcal{C}}$, $C^{(2)} \in \mathcal{M}$ and the secret key sk, the algorithm outputs

$$m \leftarrow \hat{\mathsf{Dec}}(\Delta) + C^{(2)}.$$

Remark 1. Similarly as in the scheme of Section 5.2, in level-1 and level-2 decryption, the second component of $C^{(1)}$ is not used and thus, to save bandwidth, one might require the server not to send it.

Proof of Security. Here we show that our protocol is semantically-secure and context-hiding.

Theorem 8. *If $\hat{\mathcal{HE}}$ is semantically-secure, then* $\mathsf{2S.Deg3.HE}$ *is a semantically-secure protocol for two-server delegation of computation on encrypted data.*

The proof is the same as that of Theorem 1.

Theorem 9. *If $\hat{\mathcal{HE}}$ is circuit-private, then* $\mathsf{2S.Deg3.HE}$ *is context-hiding.*

The proof is essentially the same as that of Theorem 7. The only difference is that here the simulator also needs to consider the case of level-3 ciphertexts which, however, is analogous.

5.4 Application to Double-Server-Aided Two-Party Secure Function Evaluation.

While our definition of 2S-DCED considers a *single* client who outsources the computation and learns the result, it is possible to extend this to the case of multiple clients who provide the inputs. We can indeed use our new notion to build a protocol for *double-server-aided two-party secure function evaluation* (2-SFE).

In standard 2-SFE there are two parties[12], Alice holding an input x and Bob holding an input y, who want to learn (i.e., jointly compute) $f(x, y)$ without disclosing their respective inputs. In the server-aided setting, the goal is to enable 2-SFE with the help of a set of servers who do not provide any input to the computation but are willing to do most of the work. In other words, Alice and Bob delegate the computation to the servers so that their work in the protocol does not depend on the size of the circuit f. The server-aided setting for SFE and multiparty computation was considered earlier in [15,11,1], while it has been formally studied more recently by [23] who, in particular, revisited the notion of collusion in such a scenario. Slightly more in detail, instead of the classical *monolithical* adversary who can corrupt several parties and share information (i.e., it models several different adversaries that collude), they considered a setting in which adversaries corrupt single parties and do not necessarily collude.

Here we consider this model of corruption. In particular, our basic 2-SFE protocol is proven secure against semi-honest adversaries who do not collude.

It is also worth noting that in general in server-aided SFE the servers are just a subset of the parties who do not provide inputs, and thus the servers can interact. In contrast, when building double-server-aided SFE using our notion of 2S-DCED we obtain a solution in which the two servers do *not* need to communicate at all.

Model and Definition. Here we recall the security model for securely realizing an ideal functionality in the presence of non-colluding semi-honest adversaries. For simplicity, we do it for the specific scenario of our functionality which involve two parties, Alice and Bob, and two servers S_1 and S_2. We refer the reader to [23] for the general case definitions.

Let $\mathcal{P} = (Alice, Bob, S_1, S_2)$ be the set of all protocol parties. We consider four adversaries $(\mathcal{A}_{Alice}, \mathcal{A}_{Bob}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$ that corrupt respectively $Alice, Bob, S_1$ and S_2. In the real world, *Alice* and *Bob* run on input x and y respectively (plus additional auxiliary input z_A and z_B), while S_1 and S_2 receive auxiliary inputs z_1, z_2. Let $H \subseteq \mathcal{P}$ be the set of *honest* parties. Then, for every $P \in H$, let out_P be the output of party P, whereas if P is *corrupted*, i.e., $P \in \mathcal{P} \setminus H$ then out_P denotes the view of P during the protocol Π.

For every $P^* \in \mathcal{P}$, the *partial view* of P^* in a real-world execution of protocol Π in the presence of adversaries $\mathcal{A} = (\mathcal{A}_{Alice}, \mathcal{A}_{Bob}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$ is defined as

$$REAL^{P^*}_{\Pi,\mathcal{A},H,\boldsymbol{z}}(\lambda, x, y) \equiv \{out_P : P \in H\} \cup out_{P^*}$$

In the ideal world, there is an ideal functionality f for a function f and the parties interact only with f. Here Alice and Bob sends x and y respectively to f. If any of x or y is $\perp$ then f returns $\perp$. Otherwise, f asks both S_1 and S_2 which party (Alice and or Bob) should receive the output and which not. Finally, f returns $f(x, y)$ to Alice and Bob according to the decision of S_1, S_2 (which must be unanimous). As before, let $H \subseteq \mathcal{P}$ be the set of honest parties. Then, for every $P \in H$, let out_P be the output returned by f to party P, whereas if P is *corrupted*, out_P is the same value returned by P.

[12] We consider the two party case only for simplicity and leave the multiparty extension for future work.

For every $P^* \in \mathcal{P}$, the *partial view* of P^* in an ideal-world execution in the presence of independent simulators $\mathsf{Sim} = (\mathsf{Sim}_{Alice}, \mathsf{Sim}_{Bob}, \mathsf{Sim}_{S_1}, \mathsf{Sim}_{S_2})$ is defined as

$$IDEAL^{P^*}_{\mathsf{f},\mathsf{Sim},H,\boldsymbol{z}}(\lambda, x, y) \equiv \{out_P : P \in H\} \cup out_{P^*}$$

Informally, a protocol $\mathit{\Pi}$ is considered secure against non-colluding semi-honest adversaries if it *partially emulates*, in the real world, an execution of f in the ideal world. More formally,

Definition 12. *Let* f *be a deterministic functionality among parties in* $\mathcal{P}$*. Let* $H \subseteq \mathcal{P}$ *be the subset of honest parties in* $\mathcal{P}$*. We say that* $\mathit{\Pi}$ *securely realizes* f *if there exists a set* $\mathsf{Sim} = (\mathsf{Sim}_{Alice}, \mathsf{Sim}_{Bob}, \mathsf{Sim}_{S_1}, \mathsf{Sim}_{S_2})$ *of PPT transformations (where* $\mathsf{Sim}_{Alice} = \mathsf{Sim}_{Alice}(\mathcal{A}_{Alice})$ *and so on) such that for all semi-honest PPT adversaries* $\mathcal{A} = (\mathcal{A}_{Alice}, \mathcal{A}_{Bob}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$*, for all inputs* x, y *and auxiliary inputs* $\boldsymbol{z}$*, and for all parties* $P^* \in \mathcal{P}$ *it holds*

$$\{REAL^{P^*}_{\mathit{\Pi},\mathcal{A},H,\boldsymbol{z}}(\lambda, x, y)\}_{\lambda \in \mathbb{N}} \approx_c \{IDEAL^{P^*}_{\mathsf{f},\mathsf{Sim},H,\boldsymbol{z}}(\lambda, x, y)\}_{\lambda \in \mathbb{N}}$$

where $\approx_c$ *compactly denotes computational indistinguishability.*

Our Protocol. Our protocol is described in Fig. 1 and shows how to use 2S-DCED for realizing double-server-aided 2-SFE. We assume that both Alice and Bob had run $\mathsf{2S.KeyGen}$ to generate their keys, and that all parties (including the two servers) know the public keys $\mathsf{pk}_A, \mathsf{pk}_B$ of Alice and Bob respectively. Moreover, we assume private and authenticated channels.

By plugging our 2S-DCED construction of Section 5.2 in the following protocol, we obtain a server-aided 2-SFE protocol for multivariate polynomials of degree-2. Such a protocol enables interesting practical applications. For instance, Alice and Bob can jointly compute statistical functions (e.g., mean, variance, covariance, etc.) on joint data sets without disclosing their respective data, or obtain the multiplication of two matrices A and B held by Alice and Bob respectively while keeping A and B private to each other and to the servers. Also, Alice and Bob can compute distance measures (e.g., the euclidean distance): for example Bob can have a large matrix X stored on the cloud (the two servers), and Alice may wish to learn the euclidean distance between a vector y (that she only knows) and each row of X (let us compactly denote this as $z = dist(X, y)$). Using our solution, all the work to compute $dist(X, y)$ can be outsourced to the two servers so that neither of them learns anything about X and y, and Alice does not learn anything about X beyond $z = dist(X, y)$.

Theorem 10. *If* $\mathsf{2S.DCED}$ *is a secure protocol for 2S-DCED for the class of functions in* $\mathcal{F}$*, then the protocol in Fig. 1 securely realizes any* $f \in \mathcal{F}$ *in the presence of semi-honest (non-colluding) adversaries* $(\mathcal{A}_{Alice}, \mathcal{A}_{Bob}, \mathcal{A}_{S_1}, \mathcal{A}_{S_2})$*.*

Proof (Sketch). We only provide a proof sketch. Essentially, we have to show four independent simulators Sim_{Alice}, Sim_{Bob}, Sim_{S_1}, Sim_{S_2}:

Sim_{Alice} receives x and sends x to the ideal functionality f who replies with $z = f(x, y)$. Next, Sim_{Alice} simulates $\mathcal{A}_{Alice}$ as follows. First, it generates encryption $C_{A,x} = \mathsf{2S.Enc}(\mathsf{pk}_A, x)$ of x (and does analogously to produce $C_{B,x}$). Then it runs the context-hiding simulator $\mathsf{Sim}(1^\lambda, C_{A,x}, f, \mathsf{pk}_A, z)$ to obtain a ciphertext $(\widehat{C^{(1)}}, \widehat{C^{(2)}})$ which encrypts the result z. $\widehat{C^{(1)}}, \widehat{C^{(2)}}$ are then given to $\mathcal{A}_{Alice}$ as messages coming from S_1 and S_2 respectively. Finally, it outputs $\mathcal{A}_{Alice}$'s entire view.

Inputs: Alice's input is $x \in \mathcal{M}$ and Bob's input is $y \in \mathcal{M}$. The servers S_1 and S_2 do not have any input. Extending the protocol to the case where inputs are multiple messages in $\mathcal{M}$ is straightforward.
Outputs: Both Alice and Bob are supposed to output $z = f(x, y)$.

1. Alice on input x computes $(C^{(1)}_{A,x}, C^{(2)}_{A,x}) \xleftarrow{\$} \mathsf{2S.Enc}(\mathsf{pk}_A, x)$ and $(C^{(1)}_{B,x}, C^{(2)}_{B,x}) \xleftarrow{\$} \mathsf{2S.Enc}(\mathsf{pk}_B, x)$, and she sends $C^{(1)}_{A,x}, C^{(1)}_{B,x}$ to S_1 and $C^{(2)}_{A,x}, C^{(2)}_{B,x}$ to S_2.
   Bob does the same with y, i.e., he computes $(C^{(1)}_{A,y}, C^{(2)}_{A,y}) \xleftarrow{\$} \mathsf{2S.Enc}(\mathsf{pk}_A, y)$ and $(C^{(1)}_{B,y}, C^{(2)}_{B,y}) \xleftarrow{\$} \mathsf{2S.Enc}(\mathsf{pk}_B, y)$, and he sends $C^{(1)}_{A,y}, C^{(1)}_{B,y}$ to S_1 and $C^{(2)}_{A,y}, C^{(2)}_{B,y}$ to S_2.
2. S_1 computes $C^{(1)}_A \leftarrow \mathsf{2S.Eval}_1(\mathsf{pk}_A, C^{(1)}_{A,x}, C^{(1)}_{A,y})$ and $C^{(1)}_B \leftarrow \mathsf{2S.Eval}_1(\mathsf{pk}_B, C^{(1)}_{B,x}, C^{(1)}_{B,y})$, and it sends $C^{(1)}_A$ to Alice and $C^{(1)}_B$ to Bob.
   S_2 computes $C^{(2)}_A \leftarrow \mathsf{2S.Eval}_2(\mathsf{pk}_A, C^{(2)}_{A,x}, C^{(2)}_{A,y})$ and $C^{(2)}_B \leftarrow \mathsf{2S.Eval}_2(\mathsf{pk}_B, C^{(2)}_{B,x}, C^{(2)}_{B,y})$, and it sends $C^{(2)}_A$ to Alice and $C^{(2)}_B$ to Bob.
3. Alice runs $z_A \leftarrow \mathsf{2S.Dec}(\mathsf{sk}_A, C^{(1)}_A, C^{(2)}_A)$. Bob runs $z_B \leftarrow \mathsf{2S.Dec}(\mathsf{sk}_B, C^{(1)}_B, C^{(2)}_B)$.

Fig. 1. Our protocol for double-server-aided 2-SFE.

Note that $\mathcal{A}_{Alice}$'s view consists of the encryptions of x—$C_{A,x}, C_{B,x}$—it creates and the encrypted result $z = f(x, y)$—$(\widehat{C^{(1)}}, \widehat{C^{(2)}})$—it receives. In the real world such distribution is guaranteed by that the servers and Bob are honest and by the correctness of the $\mathsf{2S.DCED}$ protocol. Therefore, by the context-hiding of $\mathsf{2S.DCED}$ the view of $\mathcal{A}_{Alice}$ in the real and ideal executions is indistinguishable.

Sim_{Bob} adopts the same strategy of Sim_{Alice}.

Sim_{S_1} runs $\mathcal{A}_{S_1}$. First, it computes (fake) encryptions of the inputs $C_{A,x}, C_{A,y}, C_{B,x}, C_{B,y}$ by running $\mathsf{2S.Enc}(\cdot,\cdot)$ on randomly chosen $\tilde{x}, \tilde{y}$. Sim_{S_1} sends the $C^{(1)}$ components of all these ciphertexts to $\mathcal{A}_{S_1}$. If $\mathcal{A}_{S_1}$ replies with $\perp$, then Sim_{S_1} returns $\perp$.
Note that in the real execution $\mathcal{A}_{S_1}$ receives encryptions of the correct inputs x and y whereas in the ideal execution, Sim_{S_1} gives to $\mathcal{A}_{S_1}$ encryptions of random $\tilde{x}, \tilde{y}$. Moreover, in both the real and ideal execution the semi-honest $\mathcal{A}_{S_1}$ does not abort since he is given valid encryptions (in the real execution this is guaranteed by that Alice and Bob are honest). Therefore, the semantic security of $\mathsf{2S.DCED}$ guarantees that the encryptions in the two executions, and thus the entire $\mathcal{A}_{S_1}$'s views, are indistinguishable.

Sim_{S_2} adopts the same strategy of Sim_{S_1}.

References

1. Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In Roger Dingledine and Philippe Golle, editors, *FC 2009*, volume 5628 of *LNCS*, pages 325–343. Springer, February 2009.
2. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, August 2004.
3. Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 325–341. Springer, February 2005.
4. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

5. Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524. Springer, August 2011.
6. Emmanuel Bresson, Dario Catalano, and David Pointcheval. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In Chi-Sung Laih, editor, *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 37–54. Springer, November / December 2003.
7. Christian Cachin, Jan Camenisch, Joe Kilian, and Joy Muller. One-round secure computation and secure autonomous mobile agents. In Ugo Montanari, José D. P. Rolim, and Emo Welzl, editors, *ICALP 2000*, volume 1853 of *LNCS*, pages 512–523. Springer, July 2000.
8. Dario Catalano, Rosario Gennaro, and Nick Howgrave-Graham. The bit security of Paillier's encryption scheme and its applications. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 229–243. Springer, May 2001.
9. Josh D. Cohen and Michael J. Fischer. A robust and verifiable cryptographically secure election scheme (extended abstract). In *26th FOCS*, pages 372–382. IEEE Computer Society Press, October 1985.
10. Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 103–118. Springer, May 1997.
11. Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 378–394. Springer, August 2005.
12. Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 119–136. Springer, February 2001.
13. Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 643–662. Springer, August 2012.
14. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, August 1984.
15. Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In *26th ACM STOC*, pages 554–563. ACM Press, May 1994.
16. David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 44–61. Springer, May 2010.
17. Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.
18. Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. i-Hop homomorphic encryption and rerandomizable Yao circuits. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 155–172. Springer, August 2010.
19. Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, pages 365–377, New York, NY, USA, 1982. ACM.
20. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
21. Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 575–594. Springer, February 2007.
22. Marc Joye and Benoît Libert. Efficient cryptosystems from 2^k-th power residue symbols. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 76–92. Springer, May 2013.
23. Seny Kamara, Payman Mohassel, and Mariana Raykova. Outsourcing multi-party computation. Cryptology ePrint Archive, Report 2011/272, 2011. `http://eprint.iacr.org/2011/272`.
24. Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *38th FOCS*, pages 364–373. IEEE Computer Society Press, October 1997.
25. Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 1219–1234. ACM Press, May 2012.
26. Carlos Aguilar Melchor, Philippe Gaborit, and Javier Herranz. Additively homomorphic encryption with d-operand multiplications. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 138–154. Springer, August 2010.
27. David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In *ACM CCS 98*, pages 59–66. ACM Press, November 1998.

28. Tatsuaki Okamoto and Shigenori Uchiyama. A new public-key cryptosystem as secure as factoring. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 308–318. Springer, May / June 1998.
29. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, May 1999.
30. R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.
31. Tomas Sander, Adam Young, and Moti Yung. Non-interactive cryptocomputing for NC1. In *40th FOCS*, pages 554–567. IEEE Computer Society Press, October 1999.
32. Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 420–443. Springer, May 2010.
33. Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 24–43. Springer, May 2010.
34. Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982.

A Adapting Known HE Schemes to be Public-Space

In this section we show that those few encryption schemes which do not fit our notion of public-space, can be easily modified to do so. This is the case for the additively-homomorphic variants (aka "in the exponent") of the ElGamal [14,10] and the BBS Linear Encryption [2] schemes, as well as for BGN [3]. The case of Okamoto-Uchiyama [28] is slightly different: we show how to tweak the scheme so that it can still be used in our construction of Section 4.

A.1 Adapting the BGN Cryptosystem to be Public-Space

The basic BGN [3] does not fit our requirement of public-space encryption scheme. This is because, in order to cope with the decryption procedure the messages are treated as (small) integers rather than elements in a ring. This difficulty can be easily overcome by forcing the messages to be treated as elements in some appropriate ring.

Below we give a short description of the adapted scheme that we call $\mathsf{BGN}^{(\mathsf{pub})}$.

Let $\mathcal{G}(1^\lambda)$ be an algorithm that on input a security parameter λ outputs a tuple $(p, q, \mathbb{G}, \mathbb{G}_T, e)$ where $\mathbb{G}$ and $\mathbb{G}_T$ are groups of order $N = pq$ and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map. Slightly more in detail, on input λ, $\mathcal{G}(1^\lambda)$ works ad follows:

- Generate two random λ-bit primes p, q and set $N = pq$.
- Generate a bilinear group $\mathbb{G}$ of order N (see [3] for details). Let g be a generator of $\mathbb{G}$, and $e : \mathbb{G} \times \mathbb{G} \leftarrow \mathbb{G}_T$ be the bilinear map.
- Output $(p, q, \mathbb{G}, \mathbb{G}_T, e)$.

$\mathsf{KeyGen}(1^\lambda)$**:** On input 1^λ where λ is a security parameter, run $\mathcal{G}(1^\lambda)$ to obtain $(p, q, \mathbb{G}, \mathbb{G}_T, e)$. Set $N = pq$. Pick two random generators $g, g' \in_R \mathbb{G}$ and set $h = (g')^q$. The public key is set as $\mathsf{pk} = (N, \mathbb{G}, \mathbb{G}_T, e, g, h)$, the private key is $\mathsf{sk} = p$. For the message space, take some small integer $t << q$ and set $\mathcal{M} = \mathbb{Z}_t$.

$\mathsf{Enc}(\mathsf{pk}, m)$**:** To encrypt $m \in \mathcal{M}$, the algorithm picks a random $r \overset{\$}{\leftarrow} \{1, ..., N\}$ and outputs

$$C = g^m h^r \in \mathbb{G}$$

$\mathsf{Eval}(\mathsf{pk}, f, C_1, \ldots, C_n)$**:** Given two ciphertexts $C_1, C_2 \in \mathbb{G}$, homomorphic addition is performed by merely multiplying – over $\mathbb{G}$ – (and re-randomizing) the ciphertexts, i.e., one takes a random $s \xleftarrow{\$} \{1, ..., N\}$ and sets $C = C_1 \cdot C_2 \cdot h^s = g^{m_1+m_2} \cdot h^{r_1+r_2+s}$. If $C_1, C_2 \in \mathbb{G}_T$, one does the same over $\mathbb{G}_T$ by using $e(g, h)$ in place of h. The (single) homomorphic multiplication can be performed by using the bilinear map as follows:

$$C = e(C_1, C_2)e(g, h)^s = e(g, g)^{m_1 m_2} e(g, h)^{\hat{s}} \in \mathbb{G}_T$$

Notice that the system remains additively homomorphic in $\mathbb{G}_T$.

$\mathsf{Dec}(\mathsf{sk}, C)$**:** To decrypt a ciphertext C using the private key p, one first computes $C^p = (g^m h^r)^p = (g^p)^m$. Next, one extracts the discrete log m of C^p in base g^p. Finally, return $(m \bmod t)$.

As in [3], in order for decryption to be efficient one needs t to be small enough so that it is possible to execute in polynomial time the extraction of discrete logs of at most T bits, where 2^T is an upper bound on the maximum value reachable by applying an admissible computation on messages of $\log t$ bits. For instance, $T = \log(n \cdot t^3)$ when considering degree-2 polynomials with $n = \mathsf{poly}(\lambda)$ monomials and coefficients in $\mathbb{Z}_t$.

The same proof of security in [3] trivially applies to the scheme described above.

Theorem 11. *If the subgroup decision assumption holds for $\mathcal{G}$, then the scheme $\mathsf{BGN}^{(\mathsf{pub})}$ described above is a semantically secure, public-space level-2 homomorphic encryption.*

Moreover, note that our adaptation applies also to the variant of BGN based on the Decision Linear assumption which can be obtained via the transformation in [16].

A.2 Adapting ElGamal and BBS "in the Exponent" to be Public-Space

The idea to adapt the additively-homomorphic variants of the ElGamal [14,10] and Boneh-Boyen-Shacham [2] encryption schemes to become public-space is essentially the same as the one described for BGN in Appendix A.1. Namely, we force messages to be treated as integers in a small ring $\mathbb{Z}_t$. In particular, this means that the discrete logarithm m extracted in the last step of the decryption procedure is then reduced $\bmod t$.

We call these two (adapted) schemes $\mathsf{ElGamal}^{(\mathsf{pub})}$ and $\mathsf{BBS}^{(\mathsf{pub})}$ respectively. For completeness, in what follows we describe this technique for the case of ElGamal "in the exponent" [10]. The same idea immediately applies to BBS.

$\mathsf{KeyGen}(1^\lambda)$ On input the security parameter λ, generate a group $\mathbb{G}$ of order p where p is a sufficiently large prime, and let $g \in \mathbb{G}$ be a generator. Sample a random $z \xleftarrow{\$} \mathbb{Z}_p^*$ and set $h = g^z$. The public key is $\mathsf{pk} = (p, \mathbb{G}, g, h)$, the private key is $\mathsf{sk} = z$. For the message space, take some small integer $t << p$ and set $\mathcal{M} = \mathbb{Z}_t$.

$\mathsf{Enc}(\mathsf{pk}, m)$**:** To encrypt $m \in \mathcal{M}$, choose a random $r \xleftarrow{\$} \mathbb{Z}_p^*$ and output $C = (U, V) \in \mathbb{G}^2$

$$U = g^m h^r, \quad V = g^r$$

$\mathsf{Eval}(\mathsf{pk}, f, C_1, \ldots, C_n)$**:** Given two ciphertexts $C_1, C_2 \in \mathbb{G}$, homomorphic addition is performed by merely multiplying – over $\mathbb{G}$ – (and re-randomizing) the ciphertexts component-wise, i.e., take a random $s \xleftarrow{\$} \mathbb{Z}_p^*$ and set

$$U = U_1 \cdot U_2 \cdot h^s = g^{m_1+m_2} \cdot h^{r_1+r_2+s}, \quad V = V_1 \cdot V_2 \cdot g^s = g^{r_1+r_2+s}$$

$\mathsf{Dec}(\mathsf{sk}, C)$ To decrypt a ciphertext $C = (U, V) \in \mathbb{G}^2$ using the private key z, compute $M = U/V^z = g^m$. Next, one extracts the discrete log m of M in base g. Finally, return $(m \bmod t)$.

In order for decryption to be efficient one needs t to be small enough so that it is possible to execute in polynomial time the extraction of discrete logs of at most T bits, where 2^T is an upper bound on the maximum value reachable by applying an admissible computation on messages of $\log t$ bits. For instance, $T = \log(n \cdot t^2)$ when considering degree-1 polynomials with $n = \mathsf{poly}(\lambda)$ monomials and coefficients in $\mathbb{Z}_t$.

It is straightforward to see that the scheme is secure under the DDH assumption.

Theorem 12. *If the Decisional Diffie-Hellman assumption holds in $\mathbb{G}$, then the scheme $\mathsf{ElGamal}^{(\mathsf{pub})}$ described above is a semantically secure, public-space linearly-homomorphic encryption.*

By applying the same adaptation idea to the BBS scheme we obtain a public-space linearly-homomorphic encryption scheme based on the Decision Linear assumption.

Theorem 13. *If the Decisional Linear assumption holds in $\mathbb{G}$, then the scheme $\mathsf{BBS}^{(\mathsf{pub})}$ is a semantically secure, public-space linearly-homomorphic encryption.*

A.3 The modified Okamoto-Uchiyama Cryptosystem

The encryption scheme of Okamoto-Uchiyama [28] does not fit our public-space requirement. However, we show that with a small tweak this scheme too can be used in our basic construction of Section 4.

Let us first recall the scheme.

$\mathsf{KeyGen}(1^\lambda)$**:** On input 1^λ where λ is a security parameter, the algorithm generates large primes p and q, such that $|p| = |q| = \lambda$ and set $N = p^2 q$. Next, it chooses $g \in (\mathbb{Z}_N)^*$ such that g is a generator of $\mathbb{Z}^*_{p^2}$. Finally set $h = g^N mod N$. Let k be such that 2^k is (slightly) less than p. The message space is $\mathcal{M} = \{0, 1\}^k$ The public key is (N, h, g) while the secret key is p.

$\mathsf{Enc}(\mathsf{pk}, m)$**:** On input a message $m \in \mathcal{M}$, choose a random $r \in \mathbb{Z}_N$ and output $C = g^m h^r \bmod N$.

$\mathsf{Dec}(\mathsf{sk}, C)$**:** On input a ciphertext C and the secret key, one first computes $D = C^{p-1} \bmod p^2 = (g^m)^{p-1} \bmod p^2$. Note that D has order p in $\mathbb{Z}^*_{p^2}$. Let $L(x) = \frac{x-1}{p}$, over the integers. The plaintext is retrieved as

$$m \leftarrow \frac{L(D)}{L\left(g^{p-1} \bmod p^2\right)} \bmod p$$

The problem with the above construction is that the scheme is homomorphic modulo p while p is not public. We overcome this difficulty by working directly modulo the (public!) value N and by performing reductions modulo p only at decryption time. This leads to the level-2 HE scheme that we call $\mathsf{OU}^{(\mathsf{level}-2)}$ sketched below.

$\mathsf{KeyGen}(1^\lambda)$**:** This is exactly as above with $\mathcal{M} = \mathbb{Z}_{2^k}$.

$\mathsf{Enc}(\mathsf{pk}, m)$**:** on input a message $m \in \mathcal{M}$, choose random $b, r \xleftarrow{\$} \mathbb{Z}_N$ and output $C = (a, \beta)$ where

$$a = (m - b) \bmod N, \quad \beta = g^b h^r \bmod N$$

$\mathsf{Eval}(\mathsf{pk}, f, C_1, \ldots, C_n)$**:** to homomorphically add two level-1 ciphertexts $(a_1, \beta_1), (a_2, \beta_2)$ compute

$$a = a_1 + a_2 \bmod N, \quad \beta = \beta_1 \cdot \beta_2 \bmod N$$

Multiplications and level-2 additions are done exactly as explained in Section 4.

$\mathsf{Dec}(\mathsf{sk}, C)$**:** to decrypt a level-1 ciphertext $(a, \beta) \in \mathbb{Z}_N \times \mathbb{Z}_N$, first "decrypt" β as described above to obtain $b \in \mathbb{Z}_p$ and then output $a + b \bmod p$. To decrypt a level-2 ciphertext $(\alpha, \beta) \in \mathbb{Z}_N \times \mathbb{Z}_N^{2\times\ell}$, first "decrypt" α as described above to obtain $a \in \mathbb{Z}_p$, second do the same with all the β's to obtain $b \in \mathbb{Z}_p^{2\times\ell}$ and finally output

$$m \leftarrow a + \sum_{i=1}^{\ell} b_{1,i} \cdot b_{2,i} \bmod p$$

It is straightforward to see that by applying the security proof of [28] the above scheme is semantically secure under the p-subgroup decision assumption.

Theorem 14. *If the p-subgroup assumption holds, then the scheme* $\mathsf{OU}^{(\mathsf{level}-2)}$ *described above is a semantically-secure homomorphic encryption that correctly and compactly evaluates polynomials in* $\mathcal{F}_2^\star$*.*

B A Generalized Construction: from Level-d to level-$2d$ Homomorphic Encryption

In this section, we show how to generalize the results from Section 4 in order to transform a homomorphic encryption supporting up to d multiplications (i.e., a level-d HE) into an HE scheme capable of (homomorphically) evaluating arithmetic circuits of degree up to $2d$. Precisely, let $\mathcal{F}_d$ be the set of (multi-variate) polynomials over a ring $\mathcal{M}$ that have total degree at most d. Then the schemes obtained via our transformation can support the evaluation of polynomials in the class $\mathcal{F}_{2d}^\star = \{f(\boldsymbol{m})\} \subset \mathcal{F}_{2d}$ where $f(\boldsymbol{m})$ is a polynomial over $\mathcal{M}$ of form $P(\boldsymbol{m}) + \sum_{i=1}^{L} Q_i(\boldsymbol{m}) \cdot R_i(\boldsymbol{m})$ where $P(\boldsymbol{m}), \{Q_i(\boldsymbol{m}), R_i(\boldsymbol{m})\}$ are all polynomials in $\mathcal{F}_d$. Essentially we enable the evaluation of one multiplication between terms of degree $\leq d$, which yields terms of degree $\leq 2d$.

Note that an instantiation of this result can be obtained by using the BGN [3] cryptosystem (with the small adaptation that we show in Appendix A.1). Interestingly, such instantiation yields a homomorphic encryption that can evaluate computations of degree up to 4 on ciphertexts and is based on a number-theoretic assumption.

Let $\mathcal{HE}^{(d)} = (\mathsf{KeyGen}^{(\mathsf{d})}, \mathsf{Enc}^{(\mathsf{d})}, \mathsf{Eval}^{(\mathsf{d})}, \mathsf{Dec}^{(\mathsf{d})})$ be a public-space level-d HE scheme. Similarly as before, and without loss of generality, we assume that $\mathsf{Eval}^{(\mathsf{d})}$ is composed of subroutines for performing homomorphic addition, multiplication and multiplication by constant, which we compactly denote by $\boxplus, \boxtimes$ and $\cdot$ respectively.

$\mathsf{KeyGen}(1^\lambda)$**:** On input 1^λ, where λ is a security parameter, it runs $\mathsf{KeyGen}^{(\mathsf{d})}$ to get $(\mathsf{pk}, \mathsf{sk})$. The algorithm outputs $(\mathsf{pk}, \mathsf{sk})$. We assume that pk implicitly contains a description of the message space $\mathcal{M}$ and the ciphertext space $\hat{\mathcal{C}}$.

$\mathsf{Enc}(\mathsf{pk}, m)$**:** The randomized encryption algorithm chooses a random value $b \overset{\$}{\leftarrow} \mathcal{M}$ and sets $a \leftarrow (m - b) \in \mathcal{M}$ and $\beta \overset{\$}{\leftarrow} \mathsf{Enc}^{(\mathsf{d})}(\mathsf{pk}, b)$. The output is $C = (a, \beta) \in \mathcal{M} \times \hat{\mathcal{C}}$.

$\mathsf{Eval}(\mathsf{pk}, f, C_1, \ldots, C_t)$**:** We describe this algorithm in terms of four different procedures: $\mathsf{Add}^{(\leq \mathsf{d})}$, $\mathsf{Mult}^{(\leq \mathsf{d})}$, $\mathsf{Add}^{(>\mathsf{d})}$, $\mathsf{Mult}^{(>\mathsf{d})}$. Informally $\mathsf{Add}^{(\leq \mathsf{d})}$ and $\mathsf{Mult}^{(\leq \mathsf{d})}$ operate over ciphertexts of level at most d (i.e., ciphertexts that were obtained either from $\mathsf{Enc}^{(\mathsf{d})}$ or from $\mathsf{Eval}^{(\mathsf{d})}$), whereas $\mathsf{Add}^{(>\mathsf{d})}$ and $\mathsf{Mult}^{(>\mathsf{d})}$ operate on ciphertexts encoding higher degree terms. Multiplications by known constants are achieved by straightforwardly adapting the corresponding procedure of $\mathcal{HE}^{(d)}$ to the new setting.

$\mathsf{Add}^{(\leq \mathsf{d})}$: On input two ciphertexts of level $\leq d$, $C_1, C_2 \in \mathcal{M} \times \hat{\mathcal{C}}$ where, for $i = 1, 2$, $C_i = (a_i, \beta_i)$ with $a_i = (m_i - b_i)$ and $\beta_i \in \mathsf{Enc}^{(\mathsf{d})}(\mathsf{pk}, b_i)$ for some $b_i \in \mathcal{M}$, this algorithm produces a ciphertext $C = (a, \beta) \in \mathcal{M} \times \hat{\mathcal{C}}$ at the same level by computing

$$\begin{aligned} a &= a_1 + a_2 = (m_1 + m_2) - (b_1 + b_2) \\ \beta &= \beta_1 \boxplus \beta_2 \in \mathsf{Enc}^{(\mathsf{d})}(\mathsf{pk}, b_1 + b_2) \end{aligned}$$

$\mathsf{Mult}^{(\leq \mathsf{d})}$: On input a level-d_1 ciphertext C_1 and a level d_2 ciphertext C_2 where, for $i = 1, 2$, $C_i = (a_i, \beta_i)$ with $a_i = (m_i - b_i)$ and $\beta_i \in \mathsf{Enc}^{(\mathsf{d})}(\mathsf{pk}, b_i)$ for some $b_i \in \mathcal{M}$, if $d_1 + d_2 \leq d$ this algorithm outputs a level-$(d_1 + d_2)$ ciphertext $C = (a, \beta)$ by computing

$$\begin{aligned} a &= a_1 \times a_2 - r = (m_1 \times m_2) - (b_1 \times m_2 + b_2 \times m_1 - b_1 \times b_2 + r) \\ \beta &= (\beta_1 \boxtimes \beta_2) \boxplus a_1\beta_2 \boxplus a_2\beta_1 \boxplus \mathsf{Enc}^{(\mathsf{d})}(\mathsf{pk}, r) \\ &\quad \in \mathsf{Enc}^{(\mathsf{d})}(\mathsf{pk}, b_1 \times m_2 + b_2 \times m_1 - b_1 \times b_2 + r) \end{aligned}$$

where $r \xleftarrow{\$} \mathcal{M}$ is randomly chosen.

$\mathsf{Mult}^{(> \mathsf{d})}$: On input a ciphertext C_1 of level $d_1 \leq d$ and a ciphertext C_2 of level $d_2 \leq d$ where, for $i = 1, 2$, $C_i = (a_i, \beta_i)$ with $a_i = (m_i - b_i)$ and $\beta_i \in \mathsf{Enc}^{(\mathsf{d})}(\mathsf{pk}, b_i)$ for some $b_i \in \mathcal{M}$, if $d_1 + d_2 > d$ this algorithm outputs a level-$(d_1 + d_2)$ ciphertext $C = (\alpha, \beta) \in \hat{\mathcal{C}} \times \hat{\mathcal{C}}^2$ by computing

$$\begin{aligned} \alpha &= \mathsf{Enc}^{(\mathsf{d})}(\mathsf{pk}, a_1 \cdot a_2) \boxplus a_1\beta_2 \boxplus a_2\beta_1 \\ &\quad \in \mathsf{Enc}^{(\mathsf{d})}\left(\mathsf{pk}, (m_1m_2 - b_1m_2 - b_2m_1 + b_1b_2) + (b_2m_1 - b_1b_2) + (b_1m_2 - b_1b_2)\right) \\ &\quad = \mathsf{Enc}^{(\mathsf{d})}(\mathsf{pk}, m_1m_2 - b_1b_2) \\ \beta &= [\beta_1, \beta_2]^\top \in (\mathsf{Enc}^{(\mathsf{d})}(\mathsf{pk}, b_1), \mathsf{Enc}^{(\mathsf{d})}(\mathsf{pk}, b_2))^\top \end{aligned}$$

$\mathsf{Add}^{(> \mathsf{d})}$: On input two ciphertexts C_1, C_2, both at level $D > d$,[13] where $C_i = (\alpha_i, \beta_i) \in \hat{\mathcal{C}} \times \hat{\mathcal{C}}^{2 \times \ell_i}$ such that $\beta_i = [(\beta_{1,1}, \beta_{2,1})^\top, \ldots, (\beta_{1,\ell_i}, \beta_{2,\ell_i})^\top]$, this algorithm returns a level-D ciphertext $C = (\alpha, \beta) \in \hat{\mathcal{C}} \times \hat{\mathcal{C}}^{2 \times (\ell_1 + \ell_2)}$ computed as follows:

$$\alpha = \alpha_1 \boxplus \alpha_2, \quad \beta = [\beta_1, \beta_2]$$

$\mathsf{Dec}(\mathsf{sk}, C)$**:** As before, we distinguish two different decryption procedures depending on whether the ciphertext C is at level $D \leq d$ or not:

Up to level d Decryption: On input a ciphertext $(a, \beta) \in \mathcal{M} \times \hat{\mathcal{C}}$ and the secret key sk, the algorithm outputs $m \leftarrow a + \mathsf{Dec}^{(\mathsf{d})}(\mathsf{sk}, \beta)$.

Beyond level d Decryption On input a ciphertext $(\alpha, \beta) \in \hat{\mathcal{C}} \times \hat{\mathcal{C}}^{2 \times \ell}$ and the secret key sk, the algorithm outputs

$$m \xleftarrow{\$} \mathsf{Dec}^{(\mathsf{d})}(\mathsf{sk}, \alpha) + \left(\sum_{i=1}^{\ell} \mathsf{Dec}^{(\mathsf{d})}(\mathsf{sk}, \beta_{1,i}) \cdot \mathsf{Dec}^{(\mathsf{d})}(\mathsf{sk}, \beta_{2,i}) \right)$$

A ReRand procedure, almost identical to that described in Section 4, can be adapted to work for the construction presented above.

Proof of Security. Using the same ideas as for the construction of Section 4, we obtain the following theorems showing the security of the scheme $\mathcal{HE}$.

[13] Notice that addition of a level $\leq d$ ciphertext with a level $>\mathsf{d}$ ciphertext can be done by first "turning" the first ciphertext into one of $>\mathsf{d}$ type.

Theorem 15. *If $\mathcal{HE}^{(d)}$ is semantically-secure, then $\mathcal{HE}$ is semantically secure.*

Theorem 16. *$\mathcal{HE}$ is a leveled circuit-private homomorphic encryption.*