A Simple BGN-type Cryptosystem from LWE

Craig Gentry Shai Halevi Vinod Vaikuntanathan

IBM T.J. Watson Research Center, NY, USA

March 30, 2010

Abstract

We construct a simple public-key encryption scheme that supports polynomially many additions and one multiplication, similar to the cryptosystem of Boneh, Goh, and Nissim (BGN). Security is based on the hardness of the learning with errors (LWE) problem, which is known to be as hard as certain worst-case lattice problems.

Some features of our cryptosystem include support for large message space, an easy way of achieving formula-privacy, a better message-to-ciphertext expansion ratio than BGN, and an easy way of multiplying two encrypted polynomials. Also, the scheme can be made identity-based and leakage-resilient (at the cost of a higher message-to-ciphertext expansion ratio).

1 Introduction

In this work we describe an encryption scheme which is additively homomorphic, and in addition also supports one multiplication. Our scheme is based on the trapdoor function proposed by Gentry, Peikert and Vaikuntanathan [10] (henceforth referred to as the GPV trapdoor function). Recall that the "public key" in the GPV trapdoor function is a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ (for parameters p and $m > n$), and the corresponding trapdoor is a full rank integer matrix with small entries $\mathbf{T} \in \mathbb{Z}^{m \times m}$ such that $\mathbf{TA} = 0 \pmod q$. The public and secret keys in our cryptosystem are exactly the same as in the GPV trapdoor function. We encrypt a square binary matrix $\mathbf{B} \in \mathbb{Z}_2^{m \times m}$ by setting

$$\mathbf{C} = \mathbf{AS} + 2\mathbf{X} + \mathbf{B} \bmod q$$

where $\mathbf{S}$ is a random "coefficient matrix" $\mathbf{S} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{X}$ is a "noise matrix" with small entries $\mathbf{X} \in Z^{m \times m}$. Ciphertext matrices can be added, and a single matrix multiplication $\mathbf{C}' = \mathbf{C}_1 \cdot \mathbf{C}_2^t \bmod q$ is also supported. ($\mathbf{C}^t$ is the transpose of $\mathbf{C}$.) To decrypt, we set

$$\mathbf{B} = \mathbf{T}^{-1} \cdot (\mathbf{TCT}^t \bmod q) \cdot (\mathbf{T}^t)^{-1} \bmod 2$$

The security of our scheme is equivalent to the hardness of learning with errors (LWE). This problem, which is related to the well-known "learning parity with noise", has become standard in the study of lattice-based cryptography. The problem was first proposed by Regev [14], and shown by Regev [14] and Peikert [13] to be as hard as *worst-case instances* of various problems in integer lattices.

1.1 An Abridged History of Homomorphic Encryption

Encryption schemes that support operations on encrypted data (aka homomorphic encryption) are very useful for secure computation. Many public-key cryptosystems supports either addition or multiplication of encrypted data, but obtaining both at the same time seems harder.

It is known that computing arbitrary functions on encrypted data can be implemented, e.g., using Yao's "garbled circuit" technique [16, 12], but the size of the ciphertext and complexity of decryption grow at least linearly with the number of gates in the circuit being computed. Also, Sander, Young and Yung [15] described a technique that permits evaluation of arbitrary circuits, but the ciphertext size grows exponentially with the circuit depth. Both of these methods can be implemented using only "general hardness assumptions" (e.g., the existence of two-flow Oblivious-Transfer protocols etc.)

Boneh, Goh, and Nissim described a cryptosystem that permitted arbitrary number of additions and one multiplication, without growing the ciphertext size [5]. Below we refer to this scheme as the BGN cryptosystem. Security of the BGN cryptosystem is based on the subgroup-membership problem in composite-order groups that admit bilinear maps. This cryptosystem immediately implies an efficient protocol for evaluating 2DNF formula (or more generally bilinear forms). Boneh et al. also described in [5] applications of the BGN cryptosystem to improving the efficiency of private information retrieval schemes (PIR) and for a voting protocol.

More recently, Aguilar Melchor, Gaborit, and Herranz described in [2] a "template" for converting some additively homomorphic encryption into a cryptosystem that permits both additions and multiplications. They show how to use this template to combine the BGN cryptosystem with the cryptosystem of Kawachi et al. [11], thus obtaining a cryptosystem that supports two multiplications and arbitrary additions, based on the hardness of both the subgroup membership problem and the unique-shortest vector problem in lattices. They also show how to use this template with the cryptosystem of Aguilar Melchor et al. [1] in order to obtain unlimited multiplication depth, where the ciphertext size grows exponentially with the multiplication depth but additions are supported without increasing the size. (Security of this last realization is based on a relatively unstudied hardness assumption, called the "Differential Knapsack Vector Problem.")

Very recently, Gentry described a fully homomorphic cryptosystem [9], supporting polynomially many additions and multiplications without increasing the ciphertext size, with security based on the hardness of finding short vectors in ideal lattices [8].

1.2 Our Contributions

Even given the great advances in homomorphic encryption over the last year, our scheme still offers some advantages over prior schemes in the literature. Below we list some of these advantages, mostly in comparison to the BGN cryptosystem.

Perhaps the main difference between our scheme and previous work is the underlying hardness assumption. In particular, ours is the first reported cryptosystem based on LWE that has more than just additive homomorphism. Also, our scheme is very efficient: it can encrypt a matrix of m^2 elements in time $\tilde{O}(m^3)$, and decryption takes comparable time.

One important difference between our scheme and the BGN cryptosystem is that the BGN cryptosystem can only encrypt messages from a small space (since on decryption one only recovers a group element g^m, and then need to search for the message m). In our scheme, we can replace the binary matrices by matrices over $\mathbb{Z}_p$ for any p, as long as the ciphertext is defined over $\mathbb{Z}_q$

where q is sufficiently larger than p. A related advantage is that by choosing a large modulus p, our scheme can be made to have ciphertext expansion of O(1) (whereas the BGN cryptosystem expands $O(\log n)$ bits of plaintext to $O(n)$ ciphertext bits.)[1]

We also note that the modulus p that defines the message space in our scheme can be chosen dynamically by the encryptor: the same public/secret key pair can be used to encrypt/decrypt messages modulo many different fields (or rings). Our scheme also support ciphertext blinding (a given ciphertext is converted into a random ciphertext that encrypts the same thing), and also the stronger property of modular blinding: Given a ciphertext that encrypts a matrix $\mathbf{B} \in \mathbb{Z}_p^{m \times n}$, and given some divisor p' of p, we can produce a random ciphertext that encrypts $\mathbf{B} \bmod p'$. For example, if the original plaintext matrix had numbers in $\mathbb{Z}_{2^n}$, we can blind the ciphertext so as to erase all but the least-significant bits of the entries in $\mathbf{B}$.

One consequence of the (standard) blinding property and the flexibility of choosing the message space is that our system provide a very simple procedure for formula-private secure computation. Namely, it is very easy to compute a 2DNF formulas (or a general bilinear form) on ciphertexts, while at the same time hiding from the holder of the secret key everything about the formula itself (other than the result of applying it on the given inputs).

Finally, our scheme inherits much of the flexibility that comes with LWE-based cryptosystems. In particular, it can be made identity-based (in the random-oracle model) using the construction of Gentry et al. [10], and it can be made leakage resilient using a recent result of Dodis et al. [6]. Both of these applications follow from the observation that the "dual Regev cryptosystem" from [10] can be described as a special case of our cryptosystem.

Relation to the AMGH transformation. It turns out that our cryptosystem fits "right out of the box" in the template of Aguilar Melchor et al. [2]. Their transformation apply to any additively homomorphic cryptosystem for which you can embed the ciphertexts back into the plaintext space while maintaining the semantics of addition, which is easy in our case. See the appendix for a brief description of their transformation and how it applies to our cryptosystem.

Combining our cryptosystem with the AMGH transformation yields a homomorphic encryption scheme for circuits of logarithmic multiplication depth (with arbitrary additions), whose security is based on the hardness of LWE.[2] We point out that even in this context, using our native multiplication operation will be advantageous, since it does not increase the ciphertext size (or the decryption time). Thus we can get either one more multiplication level for a given complexity bound, or a more efficient scheme for the same circuit depth.

Applications. Clearly, our scheme can be used as a drop-in replacement in the applications to voting and PIR that were discussed in the paper of Boneh et al. [5]. In addition, since out scheme encrypts matrices natively, it is a good match for applications that can benefit from batching, or when efficient linear algebra is important. Some examples of batching include applications that need to multiply polynomials (whose coefficients are to be encoded in the entries of the plaintext matrix) or large integers (whose bit representation is to be encoded in the entries of the plaintext matrix).

[1] To achieve such bandwidth efficient encryption, an application would have to encode its input as a matrix. Although this can always be done, it is not clear that such encoding will maintain the semantics of multiplication that the application needs. See some examples of this point in Section 5.

[2] We comment that the AMGH transformation appears to be inherently "non private", in that the holder of the secret key can deduce the multiplication structure of the circuit that was used to generate a given ciphertext. This can be addressed using generic techniques such as Yao's garbled circuits.

In Section 5.3 we describe how these can be encoded in a matrix so that a single multiplication of $m \times m$ matrices can be used to multiply two degree-$(m-1)$ polynomials (or two m-bit integers), so that the result does not leak anything about the inputs other than their product.

2 Preliminaries

Notations. We denote scalars by lower-case letters $(a, b, \ldots)$, vectors by lower-case bold letters $(\mathbf{a}, \mathbf{b}, \ldots)$, and matrices by upper-case bold letters $(\mathbf{A}, \mathbf{B}, \ldots)$. We denote the Euclidean norm of a vector $\mathbf{v}$ by $\|\mathbf{v}\|$, and the largest entry in a vector or a matrix is denoted $\|\mathbf{v}\|_\infty$ or $\|\mathbf{M}\|_\infty$, respectively. We consider the operation $(a \bmod q)$ as mapping the integer a into the interval $(-q/2, +q/2]$.

2.1 Learning with errors (LWE)

The LWE problem was introduced by Regev [14] as a generalization of "learning parity with noise". For positive integers n and $q \geq 2$, a vector $\mathbf{s} \in \mathbb{Z}_q^n$, and a probability distribution χ on $\mathbb{Z}_q$, let $A_{\mathbf{s},\chi}$ be the distribution obtained by choosing a vector $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random and a noise term $x \leftarrow \chi$, and outputting $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + x) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.

Definition 1 (LWE) *For an integer $q = q(n)$ and an error distribution $\chi = \chi(n)$ over $\mathbb{Z}_q$, the learning with errors problem $\mathsf{LWE}_{n,m,q,\chi}$ is defined as follows: Given m independent samples from $A_{\mathbf{s},\chi}$ (for some $\mathbf{s} \in \mathbb{Z}_q^n$), output $\mathbf{s}$ with noticeable probability.*

The decision variant of the LWE problem, denoted $\mathsf{distLWE}_{n,m,q,\chi}$, is to distinguish (with non-negligible advantage) m samples chosen according to $A_{\mathbf{s},\chi}$ (for uniformly random $\mathbf{s} \in_R \mathbb{Z}_q^n$), from m samples chosen according to the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

For cryptographic applications we are primarily interested in the decision problem $\mathsf{distLWE}$. Regev [14] showed that for a prime modulus q, $\mathsf{distLWE}$ can be reduced to worst-case LWE, with a loss of up to a $q \cdot \mathsf{poly}(n)$ factor in the parameter m.

At times, we find it convenient to describe the LWE problem $\mathsf{LWE}_{n,m,q,\chi}$ using a compact matrix notation: given $(\mathbf{A}, \mathbf{As} + \mathbf{x})$ where $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}$ is uniformly random, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ is the LWE secret, and $\mathbf{x} \leftarrow \chi^m$, find $\mathbf{s}$. We also use similar matrix notation for the decision version $\mathsf{distLWE}$.

Gaussian error distributions $\overline{\Psi}_\beta$. We are primarily interested in the LWE and $\mathsf{distLWE}$ problems where the error distribution χ over $\mathbb{Z}_q$ is derived from a Gaussian. For any $\beta > 0$, the density function of a Gaussian distribution over the reals is give by $D_\beta(x) = 1/\beta \cdot \exp(-\pi(x/\beta)^2)$. For an integer $q \geq 2$, define $\overline{\Psi}_\beta(q)$ to be the distribution on $\mathbb{Z}_q$ obtained by drawing $y \leftarrow D_\beta$ and outputting $\lfloor q \cdot y \rceil \pmod q$. We write $\mathsf{LWE}_{n,m,q,\beta}$ as an abbreviation for $\mathsf{LWE}_{n,m,q,\overline{\Psi}_\beta(q)}$.

Here we state some basic facts about Gaussians (tailored to the error distribution $\overline{\Psi}_\beta$); see, e.g. [7]. (In what follows overwhelming probability means probability $1 - \delta$ for δ which is negligible in n.)

Fact 1 *Let $\beta > 0$ and $q \in \mathbb{Z}$, and let the vector $\mathbf{x}$ be chosen as $\mathbf{x} \leftarrow \overline{\Psi}_\beta(q)^n$. Also let $\mathbf{y} \in \mathbb{Z}^n$ be an arbitrary vector and let $g = \omega(\sqrt{\log n})$. Then with overwhelming probability $|\langle \mathbf{x}, \mathbf{y} \rangle| \leq \beta q \cdot g \cdot \|\mathbf{y}\|$.*

Fact 2 *Let $y \in \mathbb{R}$ be arbitrary. The statistical distance between the distributions $\overline{\Psi}_\beta$ and $\overline{\Psi}_\beta + y$ is at most $|y|/(\beta q)$.*

Evidence for the hardness of $\mathsf{LWE}_{n,m,q,\beta}$ follows from results of Regev [14], who gave a *quantum* reduction from approximating certain problems on n-dimensional lattices in the worst case to within $\widetilde{O}(n/\beta)$ factors to solving $\mathsf{LWE}_{n,m,q,\beta}$ for any desired $m = \mathsf{poly}(n)$, when $\beta \cdot q \geq 2\sqrt{n}$. Recently, Peikert [13] also gave a related *classical* reduction for some other problems with similar parameters.

2.2 Trapdoor sampling

The basis of our encryption scheme is a trapdoor sampling algorithm first constructed by Ajtai [3], and later improved by Alwen and Peikert [4]. The trapdoor sampling procedure generates an (almost) uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, together with $\mathbf{T} \in \mathbb{Z}^{m \times m}$ such that (a) $\mathbf{T} \cdot \mathbf{A} = 0 \pmod q$, (b) $\mathbf{T}$ is invertible, and (c) the entries of $\mathbf{T}$ are small (say, of size $O(n \log q)$).

The trapdoor $\mathbf{T}$ can be used to solve the LWE problem relative to $\mathbf{A}$, i.e., given $\mathbf{y} = \mathbf{As} + \mathbf{x}$ where $\mathbf{x}$ is any "sufficiently short" vector, it can be used to recover $\mathbf{s}$. This is done as follows: compute

$$\mathbf{Ty} \;=\; \mathbf{T}(\mathbf{As} + \mathbf{x}) = \mathbf{TAs} + \mathbf{Tx} = \mathbf{Tx} \pmod q$$

where the last equality follows since the rows of $\mathbf{T}$ belong to lattice $\Lambda^\perp(\mathbf{A})$. Now, since both $\mathbf{T}$ and $\mathbf{x}$ contain small entries, each entry of the vector $\mathbf{Tx}$ is smaller than q, and thus $\mathbf{Tx} \bmod q$ is $\mathbf{Tx}$ itself! Finally, multiplying by $\mathbf{T}^{-1}$ (which is well-defined since $\mathbf{T}$ is a basis and therefore has full rank) gives us $\mathbf{x}$. The LWE secret $\mathbf{s}$ can then be recovered by Gaussian elimination. We state the result of Alwen and Peikert [4] below.

Lemma 1 ([3, 4]) *There is a probabilistic polynomial-time algorithm $\mathsf{TrapSamp}$ that, on input 1^n, a positive integer $q \geq 2$, and a $\mathsf{poly}(n)$-bounded positive integer $m \geq 8n \log q$, outputs matrices $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and $\mathbf{T} \in \mathbb{Z}^{m \times m}$ such that:*

- *$\mathbf{A}$ is statistically close to uniform over $\mathbb{Z}_q^{m \times n}$,*
- *the rows of $\mathbf{T}$ form a basis of the lattice $\Lambda^\perp(\mathbf{A}) \stackrel{\text{def}}{=} \{\mathbf{w} \in \mathbb{Z}^m \;:\; \mathbf{w} \cdot \mathbf{A} = 0 \pmod q\}$,*
- *the Euclidean norm of all the rows is $\mathbf{T}$ (and therefore also $\|\mathbf{T}\|_\infty$) is bounded by $O(n \log q)$. (Alwen and Peikert assert that the constant hidden in the $O(\cdot)$ is no more than 20.)*

We note that since the rows of $\mathbf{T}$ span the lattice $\Lambda^\perp(\mathbf{A})$, it follows that $\det(\mathbf{T}) = q^n$, hence for odd q we know that $\mathbf{T}$ is invertible mod 2.

3 The Encryption Scheme

For ease of presentation, we focus below on the case of encrypting binary matrices. The extension for encrypting matrices mod p for $p > 2$ is straightforward, and is discussed in Section 5.1.

Below we let n denote the security parameter. Other parameters of the system are two numbers $m, q = \mathsf{poly}(n)$ (with q an odd prime), and a Gaussian error parameter $\beta = 1/\mathsf{poly}(n)$. See Section 3.2 for concrete instantiations of these parameters. For these parameters, the message space is the set of binary m-by-m matrices, i.e., $\mathbf{B} \in \mathbb{Z}_2^{m \times m}$. Public keys are matrices $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, secret key are matrices $\mathbf{T} \in \mathbb{Z}_q^{m \times m}$, and ciphertexts are matrices $\mathbf{C} \in \mathbb{Z}_q^{m \times m}$.

$\mathsf{KeyGen}(1^n)$: Run the trapdoor sampling algorithm $\mathsf{TrapSamp}$ of Lemma 1 to obtain a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ together with the trapdoor matrix $\mathbf{T} \in \mathbb{Z}^{m \times m}$, $(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapSamp}(1^n, q, m)$. The public key is $\mathbf{A}$ and the secret key is $\mathbf{T}$.

$\mathsf{Enc}(\mathbf{A}, \mathbf{B} \in \{0,1\}^{m \times m})$: Choose a uniformly random matrix $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and an "error matrix" $\mathbf{X} \xleftarrow{\$} \overline{\Psi}_\beta(q)^{m \times m}$. Output the ciphertext

$$\mathbf{C} \leftarrow \mathbf{AS} + 2\mathbf{X} + \mathbf{B} \pmod{q}$$

(Here, $2\mathbf{X}$ means multiplying each entry of the matrix $\mathbf{X}$ by 2.)

$\mathsf{Dec}(\mathbf{T}, \mathbf{C})$: Set $\mathbf{E} \leftarrow \mathbf{TCT}^t \bmod q$, and then output $\mathbf{B} \leftarrow \mathbf{T}^{-1}\mathbf{E}(\mathbf{T}^t)^{-1} \bmod 2$.

To see that decryption works, recall that $\mathbf{T} \cdot \mathbf{A} = 0 \pmod{q}$ and therefore $\mathbf{TCT}^t = \mathbf{T}(2\mathbf{X}+\mathbf{B})\mathbf{T}^t \pmod{q}$. If in addition all the entries of $\mathbf{T}(2\mathbf{X}+\mathbf{B})\mathbf{T}^t$ are smaller than q then we also have the equality over the integers $\mathbf{E} = (\mathbf{TCT}^t \bmod q) = \mathbf{T}(2\mathbf{X}+\mathbf{B})\mathbf{T}^t$, and hence $\mathbf{T}^{-1}\mathbf{E}(\mathbf{T}^t)^{-1} = \mathbf{B} \pmod{2}$. This means that we have correct decryption as long as we set the parameter β small enough so that with high probability all the entries of $\mathbf{T}(2\mathbf{X}+\mathbf{B})\mathbf{T}^t$ are smaller than $q/2$.

Remark 1 *Note that the right-multiplication by $\mathbf{T}^t$ and $(\mathbf{T}^t)^{-1}$ on decryption are redundant here, we can instead just compute $\mathbf{B} \leftarrow \mathbf{T}^{-1}(\mathbf{TC} \bmod q) \bmod 2$. The right-multiplication is needed to decrypt product ciphertexts, as described below. As opposed to the BGN cryptosystem, in our scheme the "normal ciphertexts" and "product ciphertexts" live in the same space, and we can use the same decryption procedure to decrypt both.*

Also, we can optimize away the need to multiply by $\mathbf{T}^{-1}$ and $(\mathbf{T}^t)^{-1}$ by using the modified trapdoor $\mathbf{T}' = (\mathbf{T}^{-1} \bmod 2) \cdot \mathbf{T}$ (product over the integers). Clearly we have $\mathbf{T}'\mathbf{A} = 0 \pmod{q}$, and the entries of $\mathbf{T}'$ are not much larger than those of $\mathbf{T}$ (since $(\mathbf{T}^{-1} \bmod 2)$ is a 0-1 matrix).

3.1 Homomorphic operations

Addition. Given two ciphertexts $\mathbf{C}_1, \mathbf{C}_2$ that decrypt to $\mathbf{B}_1, \mathbf{B}_2$, respectively, it is easy to see that the matrix $\mathbf{C} = \mathbf{C}_1 + \mathbf{C}_2 \bmod q$ would be decrypted to $\mathbf{B}_1 + \mathbf{B}_2 \bmod 2$, as long as there is no "overflow" in any entry. Specifically, if we have $\mathbf{C}_1 = \mathbf{AS}_1 + 2\mathbf{X}_1 + \mathbf{B}_1$ and $\mathbf{C}_1 = \mathbf{AS}_2 + 2\mathbf{X}_2 + \mathbf{B}_2$ then

$$\mathbf{C} = \mathbf{C}_1 + \mathbf{C}_2 = \mathbf{A}(\mathbf{S}_1 + \mathbf{S}_2) + 2(\mathbf{X}_1 + \mathbf{X}_2) + (\mathbf{B}_1 + \mathbf{B}_2)$$

which would be decrypted as $\mathbf{B}_1 + \mathbf{B}_2$ as long as all the entries in $\mathbf{T}(2(\mathbf{X}_1 + \mathbf{X}_2) + \mathbf{B}_1 + \mathbf{B}_2)\mathbf{T}^t$ are smaller than $q/2$. See Section 3.2 for the exact parameters.

Multiplication. Given two ciphertexts $\mathbf{C}_1, \mathbf{C}_2$ that encrypt $\mathbf{B}_1, \mathbf{B}_2$, respectively, we compute the product ciphertext as $\mathbf{C} = \mathbf{C}_1 \cdot \mathbf{C}_2^t \bmod q$. If we have $\mathbf{C}_1 = \mathbf{AS}_1 + 2\mathbf{X}_1 + \mathbf{B}_1$ and $\mathbf{C}_2 = \mathbf{AS}_2 + 2\mathbf{X}_2 + \mathbf{B}_2$ then

$$\begin{aligned}\mathbf{C} = \mathbf{C}_1 \cdot \mathbf{C}_2^t &= (\mathbf{AS}_1 + 2\mathbf{X}_1 + \mathbf{B}_1)(\mathbf{AS}_2 + 2\mathbf{X}_2 + \mathbf{B}_2)^t \\ &= \mathbf{A} \cdot \underbrace{(\mathbf{S}_1\mathbf{C}_2^t)}_{\mathbf{S}} + 2\underbrace{(\mathbf{X}_1(2\mathbf{X}_2 + \mathbf{B}_2) + \mathbf{B}_1\mathbf{X}_2^t)}_{\mathbf{X}} + \underbrace{\mathbf{B}_1\mathbf{B}_2^t}_{\mathbf{B}} + \underbrace{(2\mathbf{X}_1 + \mathbf{B}_1)\mathbf{S}_2^t}_{\mathbf{S}'} \cdot \mathbf{A}^t \pmod{q}.\end{aligned}$$

Hence the product ciphertext has the form $\mathbf{AS} + 2\mathbf{X} + \mathbf{B} + \mathbf{S}'\mathbf{A}^t$.

As before, we see that $\mathbf{TCT}^t = \mathbf{T}(2\mathbf{X}+\mathbf{B})\mathbf{T}^t \pmod q$, and if all the entries of $\mathbf{T}(2\mathbf{X}+\mathbf{B})\mathbf{T}^t$ are smaller than $q/2$ then we have $\mathbf{E} = (\mathbf{TCT}^t \bmod q) = \mathbf{T}(2\mathbf{X}+\mathbf{B})\mathbf{T}^t$ over the integers, and therefore $\mathbf{T}^{-1}\mathbf{E}(\mathbf{T}^t)^{-1} = \mathbf{B} \pmod 2$. Below we establish the parameters that we need for this to work.

Remark 2 *We remark that the $AS + 2X + B$ format for ciphertexts, which is borrowed from [9], seem particularly conducive for homomorphic encryption. When applied in a commutative ring as in [9], it supports a large number of additions and multiplications on ciphertexts. In our case we use it in the ring of matrices, which is not commutative, but multiplying by the transpose offers a partial workaround, supporting one level of multiplication.*

3.2 Setting the parameters

Theorem 2 *Fix the security parameter n and any $c = c(n) > 0$. Let q, m, β be set as*

$$\begin{aligned} q &> 2^{20}(c+4)^3 n^{3c+4}\log^5 n, \quad q \text{ is a prime} \\ m &= \lfloor 8n\log q \rfloor \\ \beta &= \frac{1}{27n^{1+(3c/2)}\log n \log q\sqrt{qm}} \end{aligned}$$

Then the encryption scheme from above with parameters n, m, q, β supports n^c additions and one multiplication (in any order) over the matrix ring $\mathbb{Z}_2^{m\times m}$.

Remark 3 *Note that in Theorem 2 we can allow n^c additions for a non-constant c. The reason that this may be needed is for taking linear combinations of ciphertexts with large coefficients. Specifically, if we have ciphertext matrices $\mathbf{C}_1, \mathbf{C}_2, \ldots$, we can homomorphically compute $\sum \alpha_i \mathbf{C}_i$ as long as $|\sum \alpha_i| < n^c$.*

Proof First, let $\mathbf{C}$ be a matrix that was obtained by adding $\ell \le n^c$ ciphertexts, $\mathbf{C} = \sum_{i=1}^{\ell}(\mathbf{AS}_i + 2\mathbf{X}_i + \mathbf{B}_i)$. Denote $\mathbf{X} = \sum_{i=1}^{\ell}\mathbf{X}_i$, and $\mathbf{B} = \sum_{i=1}^{\ell}\mathbf{B}_i$, and we analyze the size of the entries in the matrix $\mathbf{T}(2\mathbf{X}+\mathbf{B})$. Recall from Lemma 1 that every row of T has Euclidean norm at most $20n\log q$. Applying Fact 1 (with $g = \log n - 1$), with overwhelming probability every entry of $\mathbf{TX}_i$ is at most $20\beta q(\log n - 1)n\log q$, hence every entry of $\mathbf{TX}$ is at most $20\ell\beta q(\log n-1)n\log q$. At the same time, all the $\mathbf{B}_i$'s are binary so each entry of $\mathbf{TB}$ is at most $20\ell n\log q$. Hence the absolute value of each entry in $\mathbf{T}(2\mathbf{X}+\mathbf{B})$ is bounded by

$$\begin{aligned} 20\ell n\log q\cdot(2\beta q(\log n-1)+1) &< 20\ell n\log q\cdot 2\beta q\log n \\ &= \frac{40\ell\cdot n\log n\cdot q\log q}{27n^{1+(3c/2)}\log n\cdot\log q\sqrt{qm}} = \frac{40\ell\sqrt{q}}{27n^{3c/2}\sqrt{m}} \overset{(\star)}{\ll} \sqrt{q/m} \end{aligned}$$

where inequality $(\star)$ uses the fact that $\ell \le n^c$. This in particular means that each entry in $\mathbf{T}(2\mathbf{X}+\mathbf{B})\mathbf{T}^t$ is bounded by $m\cdot 20n\log q\cdot\sqrt{q/m} = 20n\log q\sqrt{qm} \ll q/2$. Since $\mathbf{TA} = 0 \pmod q$ then $\mathbf{TCT}^t = \mathbf{T}(2\mathbf{X}+\mathbf{B})\mathbf{T}^t \pmod q$, and as all the entries in $\mathbf{T}(2\mathbf{X}+\mathbf{B})$ are less than $q/2$ in absolute value, we have the equality over the integers $(\mathbf{TCT}^t \bmod q) = \mathbf{T}(2\mathbf{X}+\mathbf{B})\mathbf{T}^t$, hence $\mathbf{T}^{-1}(\mathbf{TCT}^t \bmod q)(\mathbf{T}^t)^{-1} = \mathbf{B} \pmod 2$.

Next, consider a circuit with one ℓ_1-fan-in addition layer, followed by a multiplication layer of fan-in two, and another ℓ_2-fan-in layer of addition, where $\ell_1 + \ell_2 \le n^c$. We have shown above that

when multiplying two matrices of the form $\mathbf{AS}_i + 2\mathbf{X}_i + \mathbf{B}_i$ $(i = 1, 2)$, the result is of the form $\mathbf{AS} + 2\mathbf{X} + \mathbf{B} + \mathbf{S}'\mathbf{A}^t$. Hence all the matrices at the output of the multiplication layer are of this form, and therefore so is the output ciphertext that results from adding them all together. We now proceed to show that for that final ciphertext $\mathbf{C} = \mathbf{AS} + 2\mathbf{X} + \mathbf{B} + \mathbf{S}'\mathbf{A}^t$, it holds that every entry in $\mathbf{T}(2\mathbf{X} + \mathbf{B})\mathbf{T}^t$ is less that $q/2$ in absolute value.

Consider one particular ciphertext at the output of the multiplication layer, this ciphertext is of the form $\mathbf{C}_i = \mathbf{AS} + (2\mathbf{X}_1 + \mathbf{B}_1)(2\mathbf{X}_2^t + \mathbf{B}_2^t) + \mathbf{S}'\mathbf{A}^t$, and the matrices $(2\mathbf{X}_i + \mathbf{B}_i)$ were obtained from adding upto ℓ_1 encryptions. By the analysis from above, each entry in $\mathbf{T}(2\mathbf{X}_1 + \mathbf{B}_1)$ is bounded by $\frac{40\ell_1\sqrt{q}}{27n^{3c/2}\sqrt{m}}$, and the same bound apply also to each entry in $(2\mathbf{X}_2^t + \mathbf{B}_2^t)\mathbf{T}^t$. Hence each entry in the product $\mathbf{T}(2\mathbf{X}_1 + \mathbf{B}_1)(2\mathbf{X}_2^t + \mathbf{B}_2^t)\mathbf{T}^t$ is bounded by

$$m \cdot \left(\frac{40\ell_1\sqrt{q}}{27n^{3c/2}\sqrt{m}}\right)^2 \;=\; \left(\frac{40}{27}\right)^2 \cdot \frac{\ell_1^2}{n^{3c}} \cdot q$$

Adding $\ell_2 \leq n^c - \ell_1$ such matrices, the entry in the result is bounded by

$$\left(\frac{40}{27}\right)^2 \cdot \frac{\ell_1^2(n^c - \ell_1)}{n^{3c}} \cdot q \;\overset{(\star)}{\leq}\; \left(\frac{40}{27}\right)^2 \cdot \frac{2}{9} \cdot q \;<\; q/2$$

where the inequality $(\star)$ follows since the function $f(x) = x^2(a - x)$ obtains its maximum at $x = 2a/3$, where $f(x) = 2a^3/9$.

Once again, since each entry in $\mathbf{T}(2\mathbf{X} + \mathbf{B})\mathbf{T}^t$ is less than $q/2$ in absolute value, and since $\mathbf{TCT}^t = \mathbf{T}(2\mathbf{X} + \mathbf{B})\mathbf{T}^t \pmod q$, we have the equality over the integers $(\mathbf{TCT}^t \bmod q) = \mathbf{T}(2\mathbf{X} + \mathbf{B})\mathbf{T}^t$, which means that $\mathbf{T}^{-1}(\mathbf{TCT}^t \bmod q)(\mathbf{T}^t)^{-1} = \mathbf{B} \pmod 2$. ■

4 Security

The CPA security of the encryption scheme follows directly from the hardness of the decision LWE problem, as we now prove.

Theorem 3 *Any distinguishing algorithm with advantage ϵ against the CPA security of the scheme with parameters n, m, q, β, can be converted to a distinguisher against $\mathsf{distLWE}_{n,m,q,\beta}$ with roughly the same running time and advantage at least $\epsilon/2m$.*

Proof Let $\mathcal{A}$ be a CPA-adversary that distinguishes between encryptions of messages of its choice with advantage ϵ. We first construct a distinguisher $\mathcal{D}$ with advantage at least $\epsilon/2$ between the two distributions

$$\left\{(\mathbf{A}, \mathbf{AS} + \mathbf{X}) : \mathbf{A} \leftarrow \mathbb{Z}_q^{m\times n}, \mathbf{S} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{X} \leftarrow \overline{\Psi}_\beta(q)^{m\times m}\right\} \quad \text{and} \quad \left\{\mathsf{Unif}(\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^{m\times m})\right\}$$

The distinguisher $\mathcal{D}$ takes as input a pair of matrices $(\mathbf{A} \in \mathbb{Z}_q^{m\times n}, \mathbf{C} \in \mathbb{Z}_q^{m\times m})$, and runs the adversary $\mathcal{A}$ with $\mathbf{A}$ as the public key. Upon receiving message $\mathbf{B}_0, \mathbf{B}_1$ from the adversary, $\mathcal{D}$ chooses at random $i \in_R \{0, 1\}$, returns the challenge ciphertext $2\mathbf{C} + \mathbf{B}_i \bmod q$, then outputs 1 if the adversary $\mathcal{A}$ guesses the right i, and 0 otherwise.

On the one hand, if $\mathbf{C}$ is a uniformly random matrix then the challenge ciphertext is also uniformly random, regardless of the choice of i. Hence in this case $\mathcal{D}$ outputs 1 with probability at

most 1/2. On the other hand, if $\mathbf{C} = \mathbf{AS} + \mathbf{X} \bmod q$, then the challenge ciphertext is $2\mathbf{C} + \mathbf{B} = \mathbf{AS}' + 2\mathbf{X} + \mathbf{B} \bmod q$, where $\mathbf{S}' = 2\mathbf{S} \bmod q$ is uniformly distributed (since q and 2 are relatively prime). This is identical to the output distribution of $\mathsf{Enc}(PK, \mathbf{B}_i)$, hence by assumption $\mathcal{A}$ will guess the right i with probability $(1+\epsilon)/2$, which means that $\mathcal{D}$ outputs 1 with the same probability. Hence $\mathcal{D}$ has advantage at least $\epsilon/2$.

Finally, a standard hybrid argument can be used to convert the distinguisher $\mathcal{D}$ from above to a $\mathsf{distLWE}_{n,m,q,\beta}$ distinguisher with advantage $\epsilon/2m$. ■

Worst-case Connection. Regev [14] showed that a PPT algorithm that solves $\mathsf{distLWE}_{n,m,q,\beta}$ implies an $O(q \cdot m)$-time *quantum* algorithm that approximates various lattice problems on n-dimensional lattices in the worst case to within $\widetilde{O}(n/\beta)$ factors, when $\beta \cdot q \geq 2\sqrt{n}$. Recently, Peikert [13] also gave a related *classical* reduction with similar parameters.

Observe that for $n \geq \max\{140,\ 10\sqrt{c+4}\}$, the conditions on q, m, β imply that $\beta q > 2\sqrt{n}$. Plugging in our parameters m, q and β for the scheme that supports n^c additions, we get that breaking semantic security of the scheme is at least as hard as solving worst-case lattice problems to within a factor of $\tilde{O}(n^{3c+7/2})$.

5 Extensions and Applications

5.1 Encrypting matrices over larger rings

As we said in the introduction, we can use the same scheme to encrypt matrices over larger rings and still enjoy the same homomorphic properties, just by working with a larger modulus q. Specifically, we can encrypt matrices over $\mathbb{Z}_p$ for any p by setting $q = \omega(p^2 n^{3c+1} \log^5 n)$ while keeping all the other parameters intact. We then encrypt a matrix $\mathbf{B} \in Z_p^{m \times m}$ as $\mathbf{C} = \mathbf{AS} + p\mathbf{X} + \mathbf{B}$, and decrypt it as $\mathbf{T}^{-1} \cdot (\mathbf{TCT}^t \bmod q) \cdot (\mathbf{T}^t)^{-1} \bmod p$. (We recall again that the determinant of $\mathbf{T}$ is q^n, so $\mathbf{T}$ is invertible mod p.) Using the above with $p \geq n^{3c+1} \log^5 n$, we have $q \leq p^3$ which means that our ciphertext expansion ratio is only three. (The plaintext has $m^2 \log p$ bits while the ciphertext has $m^2 \log q$ bits.)

We comment that once we fix these larger parameters, the choice of the underlying ring can be made adaptively by the encryptor. Namely, with the same public key $\mathbf{A}$ and secret key $\mathbf{T}$, the encryptor can choose the underlying ring as $\mathbb{Z}_r$ for any $r \leq p$ (thereby computing the ciphertext as $\mathbf{C} = \mathbf{AS} + r\mathbf{X} + \mathbf{B}$), and the decryptor can decrypt accordingly.

5.2 Formula Privacy

As described so far, the scheme does not ensure "formula privacy" against the holder of the secret key. For example, given a ciphertext matrix $\mathbf{C}$, the decryptor may be able to distinguish the case where this ciphertext was obtained by multiplying an encryption of the identity with an encryption of the zero matrix from the case where it was obtained by multiplying two encryptions of the zero matrix.

This deficiency can be remedied by standard techniques. We first need to increase the size of the modulus somewhat: switching from q as specified in Theorem 2 to $q' \geq q \cdot 2^{\omega(\log n)}$. Then given a ciphertext matrix $\mathbf{C}^*$, encrypting some plaintext matrix mod p, we blind it by setting

$$\mathbf{C} \ \leftarrow \ \mathbf{C}^* + \mathbf{AS}_1 + p\mathbf{X} + \mathbf{S}_2^t \mathbf{A}^t,$$

where $\mathbf{S}, \mathbf{S}'$ are uniform in $\mathbb{Z}_{q'}^{n\times m}$ and each entry of $\mathbf{X}^*$ is chosen from $\overline{\Psi}_{\beta'}(q)$ with β' super-polynomially larger than the parameter β that is used in the scheme.

Using Fact 2 we can then show that the noise in the added $\mathbf{X}^*$ "drowns" all traces of the origin of this ciphertext. Namely, the resulting ciphertext is of the form $\mathbf{C} = \mathbf{A}\mathbf{S}'_1 + p\mathbf{X}' + \mathbf{B} + (\mathbf{S}'_2)^t\mathbf{A}^t$, where $\mathbf{S}'_1, \mathbf{S}'_2$ are uniformly random, $\mathbf{B}$ is the corresponding plaintext, and the distribution of $\mathbf{X}'$ is nearly independent of the provenance of this ciphertext matrix.

We note that the same blinding technique can be used even if the encrypted plaintext matrix was chosen in a larger ring $\mathbb{Z}_{p'}$, as long as the parameter p that is used in the blinding procedure divides the original p'.

5.3 Encrypting polynomials and large integers

To encrypt polynomials or large numbers, we need to encode them as matrices, in a way that would let us exploit the matrix operations that are supported natively by our scheme to do operations over the these polynomials or numbers.

We begin with polynomials: it is well known how to embed the coefficients of two polynomials in two matrices, so that multiplying these matrices we get all the coefficients of the resulting product polynomial. For example, for two polynomials $\hat{a}(x) = \sum a_i x^i$ and $\hat{b}(x) = \sum b_i x^i$, we can use

$$\mathbf{A} = \begin{pmatrix} a_3 & a_2 & a_1 \\ & a_3 & a_2 \\ & & a_3 \end{pmatrix} \mathbf{B} = \begin{pmatrix} b_1 & b_2 & b_3 \\ & b_1 & b_2 \\ & & b_1 \end{pmatrix} \Rightarrow \mathbf{A}\mathbf{B}^t = \begin{pmatrix} a_1b_3 + a_2b_2 + a3b_1 & a_1b_2 + a_2b_1 & a_1b_1 \\ a_2b_3 + a_3b_2 & \star & \star \\ a_3b_3 & \star & \star \end{pmatrix}$$

Note that the product matrix above is not private, in that it reveals more than just the coefficients of the product polynomial. This can be fixed easily by adding an encryption of a matrix with zero first column and first row and random entries everywhere else. Also, this simple embedding is "wasteful" in that it results in ciphertext expansion ratio of $O(m)$ (we encrypt degree-$(m-1)$ polynomials using $m \times m$ matrices). We do not know if more economical embeddings are possible.

Moving to integer multiplication, an obvious way of multiplying two m-bit integers is to just set the plaintext space to $\mathbb{Z}_p$ for some $p \geq 2^{2m}$, but working with such large plaintext space may be inconvenient. We thus seek a method for implementing large integer multiplication with a small input space. One possibility is to use the same technique as we did for polynomials, viewing the integer with binary representation $a = \sum a_i 2^i$ as a binary polynomial $\hat{a}(x)$ evaluated at $x = 2$. Given two integers a, b, we encrypt the binary coefficients of the corresponding polynomials $\hat{a}, \hat{b}$ *over plaintext space* $\mathbb{Z}_p$ *for some* $p \geq m$. Reading out the coefficients of the product polynomial, we then compute $a \cdot b = (\hat{a} \cdot \hat{b})(2)$ over the integers.

This solution is not private however, it leaks more information about a, b than just their integer product. One approach for making it private is to add random elements $r_i \in \mathbb{Z}_p$ to the first row and column of the product matrix such that $\sum_i 2^i r_i = 0 \pmod{p}$. This will make it possible for the secret key holder to recover $a \cdot b \pmod{p}$. Repeating it several times with different p's, we can then use Chinese remaindering to recover $a \cdot b$ completely.

5.4 Two-out-of-two decryption

We point out a peculiar property of our cryptosystem, which so far we were not able to find applications for. Namely, if we have encryptions of two matrices under *two different public keys*, we can multiply these two ciphertexts, thus obtaining an "ciphertext" corresponding to the product of

the two plaintext matrices". This "ciphertext" can then be decrypted by pulling together the two secret keys.

In more details, suppose that we have two public keys $\mathbf{A}_1, \mathbf{A}_2$ and the corresponding two secret keys $\mathbf{T}_1, \mathbf{T}_2$, with both pairs defined modulo the same prime number q. (We also assume for simplicity that both pairs use the same parameters n and m, but this assumption is not really needed). Then, given two ciphertexts

$$\mathbf{C}_1 = \mathbf{A}_1\mathbf{S}_1 + 2\mathbf{X}_1 + \mathbf{B}_1 \text{ and } \mathbf{C}_2 = \mathbf{A}_2\mathbf{S}_2 + 2\mathbf{X}_2 + \mathbf{B}_2,$$

we can compute the "product ciphertext" $\mathbf{C} = \mathbf{C}_1\mathbf{C}_2^t \pmod q$, corresponding to the plaintext $\mathbf{B}_1\mathbf{B}_2^t \pmod 2$. This plaintext can be recovered if we know both $\mathbf{T}_1$ and $\mathbf{T}_2$, by setting

$$B \leftarrow \mathbf{T}_1^{-1} \cdot (\mathbf{T}_1\mathbf{C}\mathbf{T}_2^t \bmod q) \cdot (\mathbf{T}_2^t)^{-1} \bmod 2$$

5.5 Identity-based and leakage-resilient BGN-type encryption

Next we show how to extend the one-multiplication homomorphism beyond just standard public-key encryption, to get more "advanced features" such as identity-based encryption and leakage-resilience. This follows from the simple observations that the "dual Regev cryptosystem" from [10] (with a different input encoding) can be viewed as a special case of our encryption scheme (for a particular form of matrices), and hence it supports the same homomorphic operations. IBE (in the random-oracle model) follows directly since Gentry et al. showed in [10] how to derive dual Regev keys from a master key, and leakage-resilience follows since Dodis et al. proved in [6] that the dual Regev cryptosystem is leakage resilient.

Recall the "dual Regev cryptosystem" from [10]: The public key is a matrix $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$, and the secret key is one short vector in the dual, namely a short $\vec{u} \in \mathbb{Z}_q^m$ such that $\vec{u}\mathbf{A} = 0 \pmod q$. Moreover, the last entry in $\vec{u}$ is always -1.

In the cryptosystem as described in [10], a bit b is encrypted by choosing a uniform vector $\vec{s} \in \mathbb{Z}_q^n$ and a small error vector $\vec{x} \in \mathbb{Z}_q^m$, and then encoding the bit b in the "most significant bit" of one entry of the ciphertext vector, namely $\vec{c} \leftarrow \mathbf{A}\vec{s}+\vec{x}+\langle 0\ldots 0\ 1\rangle^t \cdot \lceil q/2 \rceil \bmod q$. To get homomorphism, however, we encode the input in the *least significant bit*, setting $\vec{c} \leftarrow \mathbf{A}\vec{s} + 2\vec{x} + \langle 0\ldots 0\ b\rangle^t \bmod q$. With this input encoding, one can view the dual Regev cryptosystem as a special case of our cryptosystem, where the public key is the same matrix $\mathbf{A}$, and the secret key is not a full rank matrix but instead a rank-1 matrix. The matrices $\mathbf{T}, \mathbf{S}, \mathbf{X}, \mathbf{B}$ are defined as

$$\mathbf{T} = \begin{pmatrix} -\vec{u}- \\ 0 \end{pmatrix},\ \mathbf{S} = (\ 0\ \vec{s}\ ),\ \mathbf{X} = (\ 0\ \vec{x}\ ),\ \mathbf{B} = \begin{pmatrix} 0 & \\ & b \end{pmatrix}.$$

(That is, all but the top row of T are zero, all but the rightmost columns of $\mathbf{S}, \mathbf{X}$ are zero, and all but the bottom-right element of $\mathbf{B}$ are zero.)

Although this choice does not follow our input distribution for these matrices, it is nonetheless easy to show that semantic security follows from LWE. Since the key is just the dual Regev key, then the same proof as in [6] shows that it remains secure even in the face of partial leakage of the secret key. Also, it was shown in [10] how this secret key can be computed from a master secret key in an identity-based setting (in the random-oracle model).

With these choices, most of the "ciphertext matrix" is zero, so all we need to output as the ciphertext is indeed the one vector $\vec{c} \leftarrow \mathbf{A}\vec{s} + 2\vec{x} + \langle 0\ldots 0\ b\rangle^t \bmod q$, which implicitly encodes

the matrix $\mathbf{C} = (\ 0\ \vec{c}\ )$. The homomorphic operations are then applied to the implicit matrices, namely addition is just element-wise addition modulo q and multiplication of two vectors is an outer-product operation.

To decrypt a ciphertext matrix, we multiply if from left and right by the secret key vector $\vec{c}$, reducing the result first modulo q and then modulo 2. Due to the special form of the plaintext matrix $\mathbf{B}$, this is the same as multiplying by $\mathbf{T}$ on the left and and $\mathbf{T}^t$ on the right, and then taking only the bottom right element of the result.

Although the matrix $\mathbf{T}$ no longer has an inverse, we can still recover the hidden bit b. This is done simply by setting $b \leftarrow \vec{u}\mathbf{C}\vec{u}^t \bmod q \bmod 2$, without needing to multiply by the inverse, since we have

$$(\vec{u}\mathbf{C}\vec{u}^t \bmod q) \quad = \quad \vec{u} \begin{pmatrix} 0 & \\ & b \end{pmatrix} \vec{u}^t \quad = \quad b \cdot u_m^2 \pmod 2$$

Recalling that $u_m = -1$ in the dual Regev cryptosystem, this procedure indeed gives the right answer.

Acknowledgments. We thank the anonymous Eurocrypt 2010 reviewers for their valuable comments.

References

[1] C. Aguilar Melchor, G. Castagnos, and P. Gaborit. Lattice-based homomorphic encryption of vector spaces. In *IEEE International Symposium on Information Theory, ISIT'2008*, pages 1858–1862, 2008.

[2] C. Aguilar Melchor, P. Gaborit, and H. Javier. Additive Homomorphic Encryption with t-Operand Multiplications. Technical Report 2008/378, IACR ePrint archive, 2008. `http://eprint.iacr.org/2008/378/`.

[3] M. Ajtai. Generating hard instances of the short basis problem. In *ICALP*, pages 1–9, 1999.

[4] J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. In *STACS*, pages 75–86, 2009.

[5] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. pages 325–341, 2005.

[6] Y. Dodis, S. Goldwasser, Y. T. Kalai, C. Peikert, and V. Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In *TCC*, pages 361–381, 2010.

[7] W. Feller. *An Introduction to Probability Theory and Its Applications, Volume 1.* Wiley, 1968.

[8] C. Gentry. *A fully homomorphic encryption scheme.* PhD thesis, Stanford University, 2009. `http://crypto.stanford.edu/craig`.

[9] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC '09*, pages 169–178. ACM, 2009.

[10] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

[11] A. Kawachi, K. Tanaka, and K. Xagawa. Multi-bit Cryptosystems Based on Lattice Problems. In *Public Key Cryptography (PKC'07)*, volume 4450 of *Lecture Notes in Computer Science*, pages 315–329. Springer, 2007.

[12] Y. Lindell and B. Pinkas. A proof of security of yao's protocol for two-party computation. *J. Cryptology*, 22(2), 2009.

[13] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC'09*, pages 333–342. ACM, 2009.

[14] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009. Preliminiary version in STOC'05.

[15] T. Sander, A. Young, and M. Yung. Non-interactive CryptoComputing for NC1. In *40th Annual Symposium on Foundations of Computer Science*, pages 554–567. IEEE, 1999.

[16] A. C. Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science – FOCS '82*, pages 160–164. IEEE, 1982.

A The Aguilar Melchor-Gaborit-Herranz Transformation

Below is a brief description of the Aguilar Melchor-Gaborit-Herranz transformation [2] of an additively-homomorphic cryptosystem to one that supports evaluation of d-degree polynomials with upto m terms (where d, m are parameters). Here we only describe the basic approach, exemplified for the special case of $d = 3$ (since indexing becomes unwieldy for larger d). Aguilar Melchor et al. also describe in [2] some extensions and optimizations.

To evaluate d-degree binary polynomials with upto m terms, we need an encryption scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathbb{Z}_p$ for $p \geq m+1$, that satisfies the following properties:

- The ciphertext is a vector of integers in $[0, q-1]$ (for some parameter q), which we denote by Greek letters, $\mathsf{Enc}(a) = \langle \alpha[1], \ldots, \alpha[n] \rangle \in \mathbb{Z}_q^n$. (We identify Z_q with the set of integers in $[0, q-1]$.) Denote the bit-length of the parameter q by $t \stackrel{\text{def}}{=} \lceil \log(q+1) \rceil$, so the total size of the ciphertext is nt bits. (Note that to support message space $\mathbb{Z}_p$ for $p > m$, we need $nt \geq \Omega(\kappa + \log m)$ for security parameter κ.)

- $\mathcal{E}$ is additively homomorphic, via mod-q addition of the ciphertext vectors. Specifically, what we need is that for any $m' \leq m$ plaintext bits $a_1, \ldots, a_{m'} \in \{0,1\}$ and their encryption $\vec{\alpha_j} \leftarrow \mathsf{Enc}(a_j)$, the vector
$$\vec{\alpha} = \sum_j \vec{\alpha_j} \bmod q$$
whole elements are integers in $[0, q-1]$, is decrypted (with probability one) to the integer $\sum_j a_j$. (Note that since $m < p$ and all the a_j's are bits, then the sum of the a_j's is less than p, and hence addition modulo p is the same as the sum over the integers.)

Consider now a vector of m triples of plaintext bits $\langle (a_1, b_1, c_1), \ldots, (a_m, b_m, c_m) \rangle \in (\{0,1\}^3)^m$, and their encryption $\vec{\alpha_j} \leftarrow \mathsf{Enc}(a_j)$, $\vec{\beta_j} \leftarrow \mathsf{Enc}(b_j)$, $\vec{\gamma_j} \leftarrow \mathsf{Enc}(c_j)$. We now show how to generate a "ciphertext" of size $(nt)^3$ that can be decrypted to the bit $\sum_{j=1}^m a_j b_j c_j \bmod 2$. (More generally,

for degree-d polynomials the size of the ciphertext is at most $(nt)^d$. If the underlying scheme has $nt = O(\kappa + \log m)$ then this would give ciphertext size of $O(\kappa^d + \log^d m)$ for degree-d, m-term polynomials with security parameter κ.)

$\textsc{PolyEval}\Big(\Big\langle(\vec{\alpha_j}, \vec{\beta_j}, \vec{\gamma_j})\Big\rangle_{j=1,\ldots,m}\Big)$:

For $j = 1, \ldots, m$, denote the integers in the ciphertext vectors $\vec{\beta_j}$, $\vec{\gamma_j}$ by

$$\vec{\beta_j} = \langle \beta_j[1], \ldots, \beta_j[n] \rangle, \quad \vec{\gamma_j} = \langle \gamma_j[1], \ldots, \gamma_j[n] \rangle$$

Recall that these are all non-negative t-bit integers, and we denote their bit representations by

$$\beta_j[i] = \sum_{k=0}^{t-1} 2^k \cdot \beta_j^{(k)}[i], \quad \gamma_j[i'] = \sum_{k'=0}^{t-1} 2^{k'} \cdot \gamma_j^{(k')}[i'] \qquad \text{(integer addition)}$$

where each $\beta_j^{(k)}[i]$ and $\gamma_j^{(k')}[i']$ is a bit. The "compound ciphertext" consists of the $(nt)^2$ vectors

$$\vec{\delta}^{(i,k,i',k')} \stackrel{\text{def}}{=} \sum_{j=1}^{m} \underbrace{\vec{\alpha_j}}_{\text{ctxt}} \cdot \underbrace{\beta_j^{(k)}[i]}_{\text{bit}} \cdot \underbrace{\gamma_j^{(k')}[i']}_{\text{bit}} \mod q \tag{1}$$

In other words, each vector $\vec{\delta}^{(i,k,i',k')}$ is computed as a subset-sum (over Z_q) of the m ciphertext vectors $\vec{\alpha_j}$. Since we have $(nt)^2$ such vectors, the total size of the compound ciphertext is $(nt)^3$, as claimed.

$\textsc{Decrypt}\Big(\Big\{\vec{\delta}^{(i,k,i',k')} : \ k, k' \in [0, t-1], i, i' \in [1, n]\Big\}\Big)$:

1. For all k, k', i, i' decrypt $\vec{\delta}^{(i,k,i',k')}$ to get an integer $\lambda^{(i,k,i',k')} \leftarrow \mathsf{Dec}\left(\vec{\delta}^{(i,k,i',k')}\right)$. Due to the additive homomorphism of the underlying scheme, we have that

$$\lambda^{(i,k,i',k')} \quad = \quad \sum_j \underbrace{a_j}_{\text{bit}} \cdot \underbrace{\beta_j^{(k)}[i]}_{\text{bit}} \cdot \underbrace{\gamma_j^{(k')}[i']}_{\text{bit}} \qquad \text{(equality over the integers)}$$

   Note that this equality is over the integers since this is a sum of m bits, and hence must be less than p.

2. For all k', i, i' compute the integer $\lambda^{(i',k')}[i] \leftarrow \sum_{k=0}^{t-1} 2^k \cdot \lambda^{(i,k,i',k')} \bmod q$. By construction we have $\lambda^{(i',k')}[i] \in Z_q$, and by changing the order of summation we see that

$$\begin{aligned} \lambda^{(i',k')}[i] \quad &= \quad \sum_{k=0}^{t-1} 2^k \cdot \sum_j a_j \cdot \beta_j^{(k)}[i] \cdot \gamma_j^{(k')}[i'] \qquad (2) \\ &= \quad \sum_j a_j \cdot \gamma_j^{(k')}[i'] \cdot \sum_{k=0}^{t-1} 2^k \cdot \beta_j^{(k)}[i] \quad = \quad \sum_j \underbrace{a_j}_{\text{bit}} \cdot \underbrace{\gamma_j^{(k')}[i']}_{\text{bit}} \cdot \underbrace{\beta_j[i]}_{\text{integer}} \pmod q \end{aligned}$$

3. For all k', i' denote

$$\vec{\lambda}^{(i',k')} \stackrel{\text{def}}{=} \left\langle \lambda^{(i',k')}[1], \ldots, \lambda^{(i',k')}[n] \right\rangle = \sum_j \underbrace{a_j}_{\text{bit}} \cdot \underbrace{\gamma_j^{(k')}[i']}_{\text{bit}} \cdot \underbrace{\vec{\beta_j}}_{\text{ctxt}} \pmod q$$

Again, each $\vec{\lambda}^{(i',k')}$ is equal to a subset-sum over Z_q of the $\vec{\beta_j}$ ciphertext vectors.

4. For all k', i' decrypt $\vec{\lambda}^{(i',k')}$ to get an integer $\mu^{(i',k')} \leftarrow \mathsf{Dec}\left(\vec{\lambda}^{(i',k')}\right)$. As before, due to the additive homomorphism of the underlying scheme we have that

$$\mu^{(i',k')} = \sum_j \underbrace{a_j}_{\text{bit}} \cdot \underbrace{b_j}_{\text{bit}} \cdot \underbrace{\gamma_j^{(k')}[i']}_{\text{bit}} \quad \text{(equality over the integers)}$$

5. For all i' compute the integer $\mu[i'] \leftarrow \sum_{k'=0}^{t-1} 2^{k'} \cdot \mu^{(i',k')} \bmod q$. Again, we have $\mu[i'] \in Z_q$ and by changing the order of summation we see that

$$\mu[i'] = \sum_{k'=0}^{t-1} 2^{k'} \cdot \sum_j a_j \cdot b_j \cdot \gamma_j^{(k')}[i'] = \sum_j a_j \cdot b_j \cdot \sum_{k'=0}^{t-1} 2^{k'} \cdot \gamma_j^{(k')}[i'] = \sum_j \underbrace{a_j}_{\text{bit}} \cdot \underbrace{b_j}_{\text{bit}} \cdot \underbrace{\gamma_j[i']}_{\text{integer}}$$

6. Denote $\vec{\mu} \stackrel{\text{def}}{=} \langle \mu[1], \ldots, \mu[n] \rangle = \sum_j \underbrace{a_j}_{\text{bit}} \cdot \underbrace{b_j}_{\text{bit}} \cdot \underbrace{\vec{\gamma_j}}_{\text{ctxt}}$

7. Decrypt $\vec{\mu}$ to get the integer $\nu \leftarrow \mathsf{Dec}(\vec{\mu})$, and once again due to the additive homomorphism we have the equality $\nu = \sum_j a_j b_j c_j$ holding over the integers. Finally output ($\nu \bmod 2$) as the decrypted bit.