Packed Ciphertexts in LWE-based Homomorphic Encryption

Zvika Brakerski Craig Gentry Shai Halevi

September 13, 2012

Abstract

In this short note we observe that the Peikert-Vaikuntanathan-Waters (PVW) method of packing many plaintext elements in a single Regev-type ciphertext, can be used for performing SIMD homomorphic operations on packed ciphertext. This provides an alternative to the Smart-Vercauteren (SV) ciphertext-packing technique that relies on polynomial-CRT. While the SV technique is only applicable to schemes that rely on ring-LWE (or other hardness assumptions in ideal lattices), the PVW method can be used also for cryptosystems whose security is based on standard LWE (or more broadly on the hardness of "General-LWE").

Although using the PVW method with LWE-based schemes leads to worse asymptotic efficiency than using the SV technique with ring-LWE schemes, the simplicity of this method may still offer some practical advantages. Also, the two techniques can be used in tandem with "general-LWE" schemes, suggesting yet another tradeoff that can be optimized for different settings.

Acknowledgments

The first author is sponsored by DARPA under agreement number FA8750-11-C-0096. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

The second and third authors are sponsored by DARPA and ONR under agreement number N00014-11C-0390. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, or the U.S. Government. Distribution Statement "A" (Approved for Public Release, Distribution Unlimited).

1 Introduction

Homomorphic Encryption (HE) [Gen09] supports arbitrarily computation on encrypted data, even by parties that do not have the secret decryption key. Despite rapid recent advances [BV11b, BV11a, BGV12, Bra12], HE is still quite expensive. In a nutshell, this is because security considerations dictate that the ciphertexts be large, making homomorphic operations slow (as they have to manipulate these large ciphertexts). The main technique for dealing with this problem is to work with *packed ciphertexts*, namely ciphertexts that encrypt a vector of plaintext values, not just a single value. Homomorphic operations are applied to these vectors component-wise in a SIMD fashion (Single Instruction Multiple Data). Smart and Vercauteren described a ciphertext-packing technique based on polynomial-CRT [SV11], and Gentry et al. [GHS12a] used that technique to achieve a nearly optimal homomorphic evaluation (upto polylogarithic factors). These techniques rely on working over polynomial rings, and their security is based on the assumed hardness of problems in ideal lattices (such as ring-LWE [LPR10]).

Although Brakerski et al. [BV11a, BGV12, Bra12] describe also how to apply homomorphic operations the Regev cryptosystem [Reg09] (whose security is based on standard LWE), and Peikert et al. [PVW08] show how to pack many plaintext elements in one Regev ciphertext, so far the literature does not contain a description of how to use PVW packed ciphertexts to perform SIMD-type operations. Applying basic homomorphic operations to PVW packed ciphertexts is straightforward, but applying the re-linearizion technique from [BV11a] (which is needed to get more than a constant-degree homomorphism) takes some care. In this short note we describe how this is done, and discuss some practical considerations regarding using this technique.

1.1 Overview

Recall that in Regev's cryptosystem [Reg09] there is a system parameter $q \in Z$, plaintexts are bits, and secret keys and ciphertexts are vectors in $\mathbb{Z}^n$. Decryption of ciphertext $\mathbf{c}$ with secret key $\mathbf{s}$ is done by taking the inner product, reducing modulo q to the interval $[-q/2, +q/2)$, then outputting 0 if the result is smaller than $q/4$ in magnitude and outputting 1 otherwise. In a few more details, the integer $z = \langle \mathbf{s}, \mathbf{c} \rangle$ is of the form $z = k \cdot q + b \cdot \frac{q}{2} + e$, where b is the plaintext bit and k, e, are small integers (and where $\langle \cdot, \cdot \rangle$ denotes inner-product).

Homomorphic operations for Regev's cryptosystem. It is well known that Regev's cryptosystem supports additive homomorphism (for "appropriate choice" of parameters), just by adding the ciphertext vectors modulo q. Moreover, Brakerski and Vaikuntanathan observed in [BV11a] that it can be made to support also multiplicative homomorphism, using tensor products.[1] For two vectors $\mathbf{c}_1, \mathbf{c}_2$, denote by $\mathbf{c}_1 \otimes \mathbf{c}_2$ the tensor product of $\mathbf{c}_1$ and $\mathbf{c}_2$ (arranged as a vectors). That is, $\mathbf{c}_1 \otimes \mathbf{c}_2$ has dimension n^2, with each entry obtained as a product of one entry from $\mathbf{c}_1$ by one entry from $\mathbf{c}_2$. It is easy to see that for four vectors $\mathbf{s}_1, \mathbf{s}_2, \mathbf{c}_1, \mathbf{c}_2$, we always have $\langle \mathbf{s}_1, \mathbf{c}_1 \rangle \cdot \langle \mathbf{s}_2, \mathbf{c}_2 \rangle = \langle \mathbf{s}_1 \otimes \mathbf{s}_2, \mathbf{c}_1 \otimes \mathbf{c}_2 \rangle$.

Given two ciphertext vectors $\mathbf{c}_1, \mathbf{c}_2$, that encrypt the bits b_1, b_2 (relative to secret key $\mathbf{s}$), we construct a product ciphertext by first computing $\mathbf{c}_1 \otimes \mathbf{c}_2$ (over the integers), then scaling down by a $(2/q)$-factor and rounding to the nearest integer vector. Namely, $\mathbf{c}^* \leftarrow \left\lceil \frac{2}{q} \cdot (\mathbf{c}_1 \otimes \mathbf{c}_2) \right\rfloor$. It can be seen that for "appropriate choice" of parameters, if $z_i = \langle \mathbf{s}, \mathbf{c}_i \rangle$ is of the form $z_i = k_i \cdot q + b_i \cdot \frac{q}{2} + e_i$ for $i = 1, 2$ (with $\mathbf{s}$, k_i and e_i small enough), then $z^* = \langle \mathbf{s} \otimes \mathbf{s}, \mathbf{c}^* \rangle$ is of the form $z^* = k^* \cdot q + b_1 b_2 \cdot \frac{q}{2} + e^*$ where k^*, e^* also rather small. Hence the product ciphertext $\mathbf{c}^*$ can be decrypted using the tensor product $\mathbf{s} \otimes \mathbf{s}$ to the product of the plaintext bits $b_1 b_2$.

Of course, using only tensor product would have led to dimension explosion: the dimension of $\mathbf{c}^*$ is n^2, if two such ciphertexts are multiplied then the dimension becomes n^4, etc. To overcome this problem, Brakerski and Vaikuntanathan introduced in [BV11a] a re-linearization technique that shrinks the dimension from n^2 back to n. In fact that technique is more general: given any two keys $\mathbf{s}$ and $\mathbf{s}'$, one can generate a key-switching gadget that can be added to the public key, allowing conversion of ciphertexts relative to $\mathbf{s}'$ to ciphertexts relative to $\mathbf{s}$. (Roughly speaking, the key-switching gadget consists of an encryption of $\mathbf{s}$ under $\mathbf{s}'$,

[1]The observation in [BV11a] was initially made about a slightly different scheme that encodes the plaintext in the least significant bit, but here we use the variant of Brakerski [Bra12] that recovers the original form of Regev's cryptosystem.

which is a matrix because encrypting each entry in $\mathbf{s}$ takes a vector). Re-linearization is then obtained by putting in the public key a switching gadget from $(\mathbf{s} \otimes \mathbf{s})$ to $\mathbf{s}$.

Packing Regev ciphertexts. In [PVW08], Peikert et al. observed (roughly) that the same Regev ciphertext vector $\mathbf{c}$ can be used to encrypt many bits, by having many secret-key vectors. Decrypting the i'th bit is done by applying the same decryption procedure as above, using the i'th secret-key. Putting all these secret keys $\mathbf{s}_i$ as the rows of a matrix S, we have a ciphertext $\mathbf{c}$ encrypting a plaintext bit vector $\mathbf{b}$ if the integer vector $\mathbf{z} = S \cdot \mathbf{c}$ is of the form $\mathbf{z} = \mathbf{k} \cdot q + \mathbf{b} \cdot \frac{q}{2} + \mathbf{e}$ for some small integer vector $\mathbf{k}$ and $\mathbf{e}$.

Computing on packed ciphertexts. Since a packed Regev ciphertext as above is essentially the same as a standard ciphertext (except viewed relative to several different keys), then the basic homomorphic operation still work as before, i.e., homomorphic addition by adding ciphertexts (mod q) and homomorphic multiplication via tensor products.

Applying re-linearization to packed ciphertexts takes a little care, however. Although for each i we can put in the public key a switching gadget from $(\mathbf{s}_i \otimes \mathbf{s}_i)$ to $\mathbf{s}_i$, this will still not give us what we need: If $\mathbf{c}^*$ is a packed high-dimension ciphertext that for each i encrypts the product $b_i b'_i$ relative to the high-dimension key $\mathbf{s}_i \otimes \mathbf{s}_i$, then using all the $(\mathbf{s}_i \otimes \mathbf{s}_i)$-to-$\mathbf{s}_i$ gadgets will only yield a collection of non-packed ciphertexts, the i'th of which encrypts $b_i b'_i$ relative to $\mathbf{s}_i$. Instead, we need *a single* key-switching gadget, that simultaneously does all the translations $(\mathbf{s}_i \otimes \mathbf{s}_i)$-to-$\mathbf{s}_i$.

To this end, we recall that a $(\mathbf{s}_i \otimes \mathbf{s}_i)$-to-$\mathbf{s}_i$ key-switching gadget is roughly an encryption of $(\mathbf{s}_i \otimes \mathbf{s}_i)$ under $\mathbf{s}_i$. We thus will use packed ciphertexts also for this gadget, obtaining a single matrix that encrypts $(\mathbf{s}_i \otimes \mathbf{s}_i)$ under $\mathbf{s}_i$, simultaneously for all the i's. This gives us exactly what we need, letting us translate a packed ciphertext relative to the keys $(\mathbf{s}_i \otimes \mathbf{s}_i)$ to another packed ciphertext relative to the $\mathbf{s}_i$'s.

Other homomorphic operations. The above techniques are sufficient to implement SIMD-type homomorphic computation, where we compute the same function over many different inputs at once. However, we would like to use the techniques of Gentry et al. from [GHS12a] to also get efficient evaluation of a single copy. For that purpose, we need to be able to move plaintext elements between slots. For example, we need a way to transform a ciphertext that encrypts a vector $\mathbf{b}$ into another ciphertext that encrypts a cyclic shift of $\mathbf{b}$ (or any other known permutation of the entries of $\mathbf{b}$).

Moving elements between slots turns out to be very easy for this ciphertext packing method: To implement a permutation π over the slots of the plaintext vector, all we need is a packed ciphertext matrix, encrypting the $\mathbf{s}_{\pi(i)}$ under $\mathbf{s}_i$ simultaneously for all i. This is an advantage of the PVW packing method over the SV packing method using polynomial-CRT: In the SV method, plaintext slots movements are implemented using automorphisms, but only a small set of permutations can be implemented this way, hence additional work is needed to implement general permutations from this limited set (see details in [GHS12a]). In the PVW method, any permutation can be implemented directly by adding to the public key a corresponding key-switching gadget.

2 Background

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, l_1, and l_∞ norms of a vector by $\|\mathbf{v}\|$, $\|\mathbf{v}\|_1$, $\|\mathbf{v}\|_\infty$, respectively.

For an integer q we identify $\mathbb{Z}_q$ with the set of representatives from the interval $[-\frac{q}{2}, +\frac{q}{2})$, and denote by $[a]_q$ the reduction of a modulo q into this interval.[2] By $\lceil a \rfloor$ we denote the rounding of the rational a to the nearest integer, and $\lceil a \rfloor_q$ is a shorthand for $[\lceil a \rfloor]_q$. These notations are extended to vectors and matrices in the natural way.

2.1 Learning with errors (LWE)

The LWE problem was introduced by Regev [Reg09] as a generalization of "learning parity with noise". This problem has parameters the security parameter n, another integer $q \geq \mathrm{poly}(n)$, and a probability distribution χ on $\mathbb{Z}_q$, that outputs integers of magnitude much smaller than q with overwhelming probability. (Typically, χ is a discrete Gaussian distribution with zero mean and standard deviation q/β for some parameter $\beta = \mathrm{poly}(n)$.)

The search version of this problem is to discover a "hidden" vector $\mathbf{s}$ given polynomially many samples of the form $(\mathbf{a}_i, b_i)$, where the $\mathbf{a}_i$'s are chosen at random in $\mathbb{Z}_q^n$, some "error terms" $e_i \leftarrow \chi$ are drawn from χ, and the b_i's are set as $b_i = [\langle \mathbf{s}, \mathbf{a}_i \rangle + e_i]_q$.

The decision variant of the LWE problem is to distinguish a sequence of such pairs $\{(\mathbf{a}_i, [\langle \mathbf{s}, \mathbf{a}_i \rangle + e_i]_q)\}_i$ from a sequence of uniform random pairs in $\mathbb{Z}_q^n \times \mathbb{Z}_q$. As defined by Regev, the hidden vector $\mathbf{s}$ is chosen uniformly at random in $\mathbb{Z}_q^n$, but Applebaum et al. proved in [ACPS09] that this is equivalent (in terms of hardness) to the variant where $\mathbf{s}$ is chosen from the error distribution, $\mathbf{s} \leftarrow \chi^n$. It is this latter variant that we use for homomorphic encryption.

Evidence for the hardness of the LWE problem follows from results of Regev [Reg09] who gave *quantum* reductions from approximation of certain problems on n-dimensional lattices in the worst case to solving LWE with some Gaussian error distributions, and Peikert [Pei09] who gave *classical* reductions for some other problems with similar parameters.

2.2 Regev's Cryptosystem

Regev described in [Reg09] a public-key encryption scheme whose security relies on the hardness of decision-LWE. To simplify the presentation, here we describe this cryptosystem as a symmetric (shared-key) encryption scheme. (Since this scheme supports homomorphic computation, one can use generic transformations to obtain a public-key scheme, see, e.g., [Rot11].) Below we also assume for simplicity that q is even, and concentrate on the case where the plaintext space is $\mathbb{Z}_2 = \{0, 1\}$. The extensions to larger plaintext spaces and arbitrary moduli q are straightforward. Also, below we denote the security parameter by n' and let $n = n' + 1$.

In this symmetric-key variant, the secret key is the LWE hidden vector, chosen (say) as $\mathbf{s}' \leftarrow \chi^{n'}$. Encrypting a bit σ is done by choosing a vector $\mathbf{a} \in \mathbb{Z}_q^{n'}$ uniformly at random and a small error term $e \leftarrow \chi$, setting $b = [\sigma \frac{q}{2} - \langle \mathbf{s}', \mathbf{a} \rangle + e]_q$, and outputting $(b, \mathbf{a})$. To decrypt, compute $d = [b + \langle \mathbf{s}', \mathbf{a} \rangle]_q$ and output 1 if d has magnitude more than $q/4$ and 0 otherwise. Decryption succeeds because the error term e had magnitude smaller than $q/4$ (as it is chosen from the error distribution) and we have $d = [\sigma \frac{q}{2} + e]_q$.

Considering the n-vectors $\mathbf{s} = (1|\mathbf{s}')$ and $\mathbf{c} = (b|\mathbf{a})$, the integer d is just the inner product $\langle \mathbf{s}, \mathbf{c} \rangle$ modulo q, and decryption can be expressed using the formula $\sigma = \lceil \, [\langle \mathbf{s}, \mathbf{c} \rangle]_q \, / \, (q/2) \, \rfloor_2$. Also, since $\mathbf{s}'$ was chosen from the error distribution χ and therefore has entries much smaller then q, then the integer $\langle \mathbf{s}, \mathbf{c} \rangle$ (without reduction modulo q) has magnitude at most $q/2 \cdot \|\mathbf{s}'\| \ll q^2$. It follows that a valid encryption of the bit σ

[2]The only exception is $\mathbb{Z}_2$, which we identify with $\{0, 1\}$ rather than $\{-1, 0\}$.

relative to $\mathbf{s}$ is a vector $\mathbf{c}$ such that the inner product of $\mathbf{s}$ and $\mathbf{c}$ is of the form

$$\langle \mathbf{s}, \mathbf{c} \rangle \;=\; kq + \sigma\frac{q}{2} + e,$$

where k, e are of magnitude much smaller than q.

For the basic Regev cryptosystem we only need $|e| < q/4$ and the size of k does not matter. However for the homomorphic operations that are described below we need $k, e \ll q$. Typically q is set to be super-polynomial (or even sub-exponential) in n, and e, k are bounded by some polynomial in n.

2.3 Homomorphic Computation

Let $\mathbf{c}_1, \mathbf{c}_2$ be two valid encryptions of the bits b_1, b_2, respectively, relative to the same key $\mathbf{s}$. By the above, this means that we have $\langle \mathbf{s}, \mathbf{c}_i \rangle \;=\; k_i q + b_i\frac{q}{2} + e_i$ for small k_i, e_i. It therefore follows that their sum, $\mathbf{c}' = [\mathbf{c}_1 + \mathbf{c}_2]_q$, satisfies $\langle \mathbf{s}, \mathbf{c}' \rangle = k' q + (\sigma_1 \oplus \sigma_2)\frac{q}{2} + e'$, where $e' = e_1 + e_2$ and k' is either $k_1 + k_2$ or $k_1 + k_2 \pm 1$. Hence $\mathbf{c}'$ is a valid encryption of the bit $\sigma_1 \oplus \sigma_2$.

More interesting is the observation from [BV11a, Bra12] that Regev's scheme also supports multiplication via tensor products. Let $\mathbf{c}^* = \mathbf{c}_1 \otimes \mathbf{c}_2$ denote the dimension-n^2 vector whose entries are all the products of one entry from $\mathbf{c}_1$ and one from $\mathbf{c}_2$ (without any modular reduction), and similarly denotes $\mathbf{s}^* = \mathbf{s} \otimes \mathbf{s}$. Then over the rationals we have:

$$\begin{aligned}\left\langle \mathbf{s}^*,\ \tfrac{2}{q} \cdot \mathbf{c}^* \right\rangle \;&=\; \tfrac{2}{q} \cdot \langle \mathbf{s}, \mathbf{c}_1 \rangle \cdot \langle \mathbf{s}, \mathbf{c}_2 \rangle \;=\; \tfrac{2}{q} \cdot \big(k_1 q + b_1(q/2) + e_1\big) \cdot \big(k_2 q + b_2(q/2) + e_2\big) \\ &=\; (2k_1k_2 + k_1b_2 + k_2b_1) \cdot q \;+\; b_1b_2 \cdot (q/2) \;+\; \underbrace{(2k_1 + b_1)e_2 + (2k_2 + b_2)e_1 + \tfrac{2e_1e_2}{q}}_{e''}\end{aligned}$$

Note that the error term e'' above is only polynomially (in n) larger than e_1, e_2 themselves, because k_1, k_2 are bounded by $\mathrm{poly}(n)$.

Rounding $\frac{2}{q}\mathbf{c}^*$ to an integer vector we have $\left\lceil \frac{2}{q}\mathbf{c}^* \right\rfloor = \frac{2}{q}\mathbf{c}^* + \mathbf{e}$ for some rounding-error vector $\mathbf{e}$ with $\|\mathbf{e}\|_\infty \le \frac{1}{2}$. Hence we get

$$\left\langle \mathbf{s}^*,\ \left\lceil \tfrac{2}{q}\mathbf{c}^* \right\rfloor \right\rangle \;=\; \left\langle \mathbf{s}^*,\ \tfrac{2}{q}\mathbf{c}^* \right\rangle + \langle \mathbf{s}^*, \mathbf{e} \rangle \;=\; k''q + b_1b_2(q/2) + e^*,$$

where $e^* = (2k_1 + b_1)e_2 + (2k_2 + b_2)e_1 + \frac{2e_1e_2}{q} + \langle \mathbf{s}^*, \mathbf{e} \rangle$ and k'' is some integer. Since both $\mathbf{s}^* = \mathbf{s} \otimes \mathbf{s}$ and $\mathbf{e}$ have small entries then the added term $\langle \mathbf{s}^*, \mathbf{e} \rangle$ is insignificant, and we have $|e^*| \le \mathrm{poly}(n) \cdot (|e_1| + |e_2|) \ll q$. Finally, reducing the rounded vector modulo q we get a vector $\mathbf{c}'' = \left\lceil \frac{2}{q}\mathbf{c}^* \right\rfloor_q$ satisfying

$$\langle \mathbf{s}^*, \mathbf{c}'' \rangle \;=\; k^*q + b_1b_2(q/2) + e^*,$$

for the same small error term e^* are above. The factor k^* can be bounded by

$$k^* \;\le\; 1 + \frac{|\langle \mathbf{s}^*, \mathbf{c}'' \rangle|}{q} \;\le\; 1 + \frac{\|\mathbf{s}^*\| \cdot \|\mathbf{c}''\|}{q} \;\overset{(\star)}{\le}\; 1 + \frac{\|\mathbf{s}\|^2 \cdot nq/2}{q} \;=\; 1 + \frac{\|\mathbf{s}\|^2 \cdot n}{2} \;\ll\; q. \qquad (1)$$

The Inequality $(\star)$ follow since $\|\mathbf{s}^*\| = \|\mathbf{s} \otimes \mathbf{s}\| = \|\mathbf{s}\|^2$, and $\mathbf{c}''$ is an n^2-vector with entries of magnitude no larger than $q/2$. We conclude that $\mathbf{c}''$ is a valid encryption of the bit b_1b_2 relative to key $\mathbf{s}^* = \mathbf{s} \otimes \mathbf{s}$.

Key-switching. The multiplication-via-tensoring technique from above comes with the unpleasant side-effect that the dimension of product ciphertext is squared. To overcome this problem, Brakerski and Vaikuntanathan introduced in [BV11a] a key-switching technique. They added to the public key a gadget to enable mapping ciphertexts relative to the high-dimension $\mathbf{s}^*$ into ciphertexts that encrypt the same thing relative to the lower-dimension $\mathbf{s}$. Below we describe a variant similar to the key-switching technique of Gentry et al. [GHS12b], which is a little easier to explain (and more efficient to implement) than the variants from [BV11a, Bra12].

On a high level, the $\mathbf{s}^*$-to-$\mathbf{s}$ key-switching gadget is a (slightly twisted) encryption of $\mathbf{s}^*$ under $\mathbf{s}$. In more detail, for each entry $\mathbf{s}^*[i]$ we put in the public key a *rational* "ciphertext" vector $\mathbf{w}_i$ (say, with $\ell = \Theta(\log q)$ bits of precision to the right of the binary point). This vector satisfies the equality $\langle \mathbf{s}, \mathbf{w}_i \rangle = k_i q + \mathbf{s}^*[i] + e_i$ over the rationals, where the factor k_i is an integer and the magnitude of the error term is bounded by $|e_i| \leq \mathrm{poly}(n)/q$. It can be shown that assuming hardness of decision-LWE with modulus $2^\ell q$ (and a circular-security assumption), these vectors $\mathbf{w}_i$ are pseudo-random. Putting all these vectors as the columns of a matrix, we get an n-by-n^2 rational matrix $\mathbf{W}$ such that over the rationals

$$\mathbf{s} \times \mathbf{W} = \mathbf{k}q + \mathbf{s}^* + \mathbf{e},$$

with $\mathbf{k}$ an integer vector and $\|\mathbf{e}\|_\infty \leq \mathrm{poly}(n)/q$.

Given the n^2-dimension vector $\mathbf{c}^*$ satisfying $\langle \mathbf{s}^*, \mathbf{c}^* \rangle = k'q + b(q/2) + e'$ (for small integers $k', e' \ll q$ and a bit b), we multiply $\mathbf{c}^*$ by $\mathbf{W}$, then round and reduce mod q, getting $\mathbf{c} = \lceil \mathbf{W}\mathbf{c}^* \rfloor_q$. We can express $\mathbf{c}$ as $\mathbf{c} = \mathbf{W}\mathbf{c}^* + \mathbf{e}^* + \mathbf{k}^* q$, with $\mathbf{e}^*$ the rounding error and $\mathbf{k}^*$ the integer factor from reduction modulo q. Then we have

$$\begin{aligned} \langle \mathbf{s}, \mathbf{c} \rangle &= \langle \mathbf{s},\ \mathbf{W}\mathbf{c}^* + \mathbf{e}^* + \mathbf{k}^* q \rangle \ = \ \mathbf{s}\mathbf{W}\mathbf{c}^* + \langle \mathbf{s}, \mathbf{e}^* \rangle + \langle \mathbf{s}, \mathbf{k}^* \rangle q \\ &= (\langle \mathbf{k}, \mathbf{c}^* \rangle q + \langle \mathbf{s}^*, \mathbf{c}^* \rangle + \langle \mathbf{e}, \mathbf{c}^* \rangle) + \langle \mathbf{s}, \mathbf{e}^* \rangle + \langle \mathbf{s}, \mathbf{k}^* \rangle q \\ &= \underbrace{(\langle \mathbf{s}, \mathbf{k}^* \rangle + \langle \mathbf{s}^*, \mathbf{k} \rangle + k')}_{\tilde{k}} q + b(q/2) + \underbrace{\langle \mathbf{e}, \mathbf{c}^* \rangle + \langle \mathbf{s}, \mathbf{e}^* \rangle + e'}_{\tilde{e}} \end{aligned}$$

The magnitude of the error term $\tilde{e}$ can be bounded by noticing the following:

- $\mathbf{e}^*$ is the rounding error, so $\|\mathbf{e}^*\|_\infty \leq \frac{1}{2}$, and therefore $|\langle \mathbf{s}, \mathbf{e}^* \rangle| < \|\mathbf{s}\|_1 \ll q$;
- We have $\|\mathbf{e}\|_\infty \leq \mathrm{poly}(n)/q$ and $\|\mathbf{c}^*\|_\infty \leq q/2$, hence $|\langle \mathbf{e}, \mathbf{c}^* \rangle| \leq n^2 \cdot \|\mathbf{e}\|_\infty \cdot \|\mathbf{c}^*\|_\infty = \mathrm{poly}(n) \ll q$.

It thus follows that $|\tilde{e}| \leq |\langle \mathbf{s}, \mathbf{e}^* \rangle| + |\langle \mathbf{e}, \mathbf{c}^* \rangle| + e' \ll q$. As for the size of the factor $\tilde{k}$, here we have similarly to Equation (1):

$$\tilde{k} \ \leq \ 1 + \frac{|\langle \mathbf{s}, \mathbf{c} \rangle|}{q} \ \leq \ 1 + \frac{\|\mathbf{s}\| \cdot \|\mathbf{c}\|}{q} \ \leq \ 1 + \frac{\|\mathbf{s}\| \cdot q\sqrt{n}/2}{q} \ = \ 1 + \frac{\|\mathbf{s}\| \cdot \sqrt{n}}{2} \ \ll \ q.$$

Summing up, we obtained a dimension-n ciphertext vector $\mathbf{c}$ satisfying $\langle \mathbf{s}, \mathbf{c} \rangle = \tilde{k}q + b(q/2) + \tilde{e}$ with $\tilde{k}, \tilde{e} \ll q$, so this is a valid encryption of b relative to $\mathbf{s}$.

2.4 Packed Ciphertexts in Regev's Cryptosystem

As described above, we need a dimension-$(n'+1)$ ciphertext to encrypt a single plaintext bit. Peikert et al. observed in [PVW08] that this ciphertext-expansion ratio can be reduced by packing many plaintext bits in a single ciphertext. Specifically, we can encrypt m' plaintext bits in a ciphertext of dimension $n' + m'$. Below we denote $m = n' + m'$.

To this end, we choose m' (rather than one) secret vectors of dimension n', $\mathbf{s}'_i \leftarrow \chi^{n'}$, and store them as the rows of an m'-by-n' secret matrix $\mathbf{S}'$. Rather than using a dimension-$(n'+1)$ secret-key vector $\mathbf{s} = (1|\mathbf{s}')$ as before, we now use an m'-by-m secret-key matrix $\mathbf{S} = (\mathbf{I}|\mathbf{S}')$, where $\mathbf{I}$ is the m'-by-m' identity matrix.

Recall that for simplicity we describe this cryptosystem as a symmetric encryption scheme. To encrypt a vector of bits $\mathbf{b} \in \{0,1\}^{m'}$, we choose a random vector $\mathbf{a} \in \mathbb{Z}_q^{n'}$ and a random error vector $\mathbf{x} \leftarrow \chi^m$, set $\mathbf{d} = [\mathbf{b} \cdot \frac{q}{2} - \mathbf{S}'\mathbf{a} + \mathbf{x}]_q$ and output the ciphertext vector $\mathbf{c} = (\mathbf{d}|\mathbf{a}) \in \mathbb{Z}_q^m$. To decrypt the m-ciphertext $\mathbf{c}$, we multiply it by $\mathbf{S}$ modulo q, then for each entry of the result output 1 if that entry has magnitude larger than $q/4$ and 0 otherwise. This works because $\mathbf{Sc} = \mathbf{d} + \mathbf{S}'\mathbf{a} = \mathbf{b} \cdot \frac{q}{2} + \mathbf{x} \pmod q$, and the entries of $\mathbf{x}$ are all much smaller then q. Decryption can be expressed using the formula $\mathbf{b} = \lceil\, [\mathbf{Sc}]_q \,/\, (q/2) \,\rfloor_2$. Also, a valid ciphertext relative to $\mathbf{S}$ is a vector $\mathbf{c}$ such that $\mathbf{Sc}$ is of the form

$$\mathbf{Sc} \;=\; \mathbf{k} \cdot q + \mathbf{b} \cdot \frac{q}{2} + \mathbf{e},$$

where $\|k\|_\infty$ and $\|\mathbf{e}\|_\infty$ are of much smaller than q. In other words, the same vector $\mathbf{c}$ is a valid ciphertext relative to all the rows of $\mathbf{S}$, encrypting the i'th bit of $\mathbf{b}$ relative to the i'th row of $\mathbf{S}$.

3 Computing on Packed Ciphertexts

The techniques from Section 2 can be combined "right out of the box" to provide homomorphic evaluation of polynomial of constant degree on packed ciphertexts: Let $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{Z}_q^m$ be two packed ciphertexts, encrypting the bit vectors $\mathbf{b}_1, \mathbf{b}_2 \in \{0,1\}^{m'}$, respectively, relative to the secret key $\mathbf{S} \in \mathbb{Z}_q^{m' \times m}$. Denote the i'th row of $\mathbf{S}$ by $\mathbf{s}_i$, then for all i we have that $\mathbf{c}_1, \mathbf{c}_2$ encrypt the i'th bits of $\mathbf{b}_1, \mathbf{b}_2$, respectively, relative to $\mathbf{s}_i$.

Just like in Section 2.3, this means that for all i, the vector $\mathbf{c}' = [\mathbf{c}_1 + \mathbf{c}_2]_q$ is a valid ciphertext relative to $\mathbf{s}_i$, encrypting the XOR of the i'th bits of $\mathbf{b}_1$ and $\mathbf{b}_2$. In other words, $\mathbf{c}'$ is a valid encryption of the vector $[\mathbf{b}_1 + \ vb_2]_2$, relative to the secret key $\mathbf{S}$. Similarly, setting $\mathbf{c}'' = \left\lceil \frac{2}{q}\mathbf{c}_1 \otimes \mathbf{c}_2 \right\rfloor_q$, we get that for all i, the vector $\mathbf{c}''$ is a valid ciphertext relative to $\mathbf{s}_i^* = \mathbf{s}_i \otimes \mathbf{s}_i$, encrypting the product of the i'th bits of $\mathbf{b}_1$ and $\mathbf{b}_2$. In other words, denoting by $\mathbf{S}^*$ the $m' \times m^2$ matrix with the $\mathbf{s}_i^*$'s as rows, the vector $\mathbf{c}''$ is a valid ciphertext relative to $\mathbf{S}^*$, encrypting the bitwise product $\mathbf{b}_1 \boxdot \mathbf{b}_2 \in \{0,1\}^{m'}$.

3.1 Key-Switching with Packed Ciphertexts

Performing key-switching/relinearization on packed ciphertext takes some care. Clearly, one can put in the public key key-switching gadgets $\mathbf{W}_i$ that for each i enable switching from $\mathbf{s}_i^*$ back to $\mathbf{s}_i$. However, this will only let us convert the single high-dimension ciphertext $\mathbf{c}''$ (which is a valid ciphertext relative to all the $\mathbf{s}_i^*$'s) into a collection of m' low-dimension ciphertexts, the i'th of which is valid with respect to $\mathbf{s}_i$. Instead, we would like to have *a single gadget* that lets us convert the single ciphertext $\mathbf{c}''$ into a single low-dimension ciphertext which is valid relative to all the $\mathbf{s}_i$'s.

Recalling that a key-switching gadget from one key to another is roughly an encryption of the first key under the second, we use our ability to pack many plaintext into one ciphertext to "encrypt all the keys $\mathbf{s}_i^*$ in a single ciphertext" relative to the $\mathbf{s}_i$'s: Our key-switching gadget from $\mathbf{S}^*$ to $\mathbf{S}$ will be a rational matrix $\mathbf{W}$ such that $\mathbf{SW} \equiv \mathbf{S}^* + E \pmod q$ for a sufficiently small error matrix E. In more detail, denoting $m = n' + m'$, then for a secret key $\mathbf{S} = (\mathbf{I}|\mathbf{S}') \in \mathbb{Z}^{m' \times m}$ and an "extended secret key" $\mathbf{S}^* \in \mathbb{Z}^{m' \times m^2}$, we choose the key-switching matrix $\mathbf{W} \in \mathbb{Q}^{m \times m^2}$ as follows:

Let $\ell = \lceil \log q \rceil$, and let χ be an error distribution (over $\mathbb{Z}$) for which the decision-LWE problem with modulus $Q = 2^\ell q$ is hard, and such that with overwhelming probability, elements drawn from χ are much smaller in magnitude than q. Say $|e| \le \text{poly}(n) \ll q$ whp for $e \leftarrow \chi$. Note that we require hardness relative to the larger modulus Q, even though the error is much smaller than the small modulus q. Since $Q \approx q^2$ then all the known hardness results for LWE carry also to this case. (However, since we have larger modulus-to-noise ratio then we would need a larger dimension n' to get the same level of concrete security.)

For each $j \in \{1, 2, \ldots, m^2\}$, we denote the j'th column of the matrix $\mathbf{S}^*$ by $\tilde{\mathbf{s}}_j \in \mathbb{Z}^{m'}$. The j'th column of $\mathbf{W}$ is set by drawing a random vector $\mathbf{a}_j \in \mathbb{Z}_Q^{n'}$ uniformly at random, drawing an error vector $\mathbf{e}_j \leftarrow \chi^{m'}$, computing $\mathbf{d}_j = [2^\ell \tilde{\mathbf{s}}_j - \mathbf{S}'\mathbf{a}_j + \mathbf{e}_j]_Q$, then dividing by 2^ℓ and outputting the rational column vector (with ℓ bits of precision), $\mathbf{w}_j = (\mathbf{d}_j|\mathbf{a}_j)^T/2^\ell \in \mathbb{Q}^m$.

Let us denote by $\mathbf{k}_j \in \mathbb{Z}^m$ the integer vector containing the factors of the reduction modulo Q from above, so $\mathbf{d}_j = [2^\ell \tilde{\mathbf{s}}_j - \mathbf{S}'\mathbf{a}_j + \mathbf{e}_j]_Q = 2^\ell \tilde{\mathbf{s}}_j - \mathbf{S}'\mathbf{a}_j + \mathbf{e}_j + \mathbf{k}Q$. We thus have for every column $\mathbf{w}_j$:

$$\begin{aligned}\mathbf{S}\mathbf{w}_j &= \frac{(\mathbf{I}|\mathbf{S}') \cdot (\mathbf{d}_j|\mathbf{a}_j)^T}{2^\ell} = \frac{\mathbf{d}_j + \mathbf{S}'\mathbf{a}_j}{2^\ell} = \frac{(2^\ell \tilde{\mathbf{s}}_j - \mathbf{S}'\mathbf{a}_j + \mathbf{e}_j + \mathbf{k}Q) + \mathbf{S}'\mathbf{a}_j}{2^\ell} = \mathbf{k}_j q + \tilde{\mathbf{s}}_j + \tfrac{\mathbf{e}_j}{2^\ell} \\ &\equiv \tilde{\mathbf{s}}_j \pm \text{poly}(n)/q \pmod q\end{aligned}$$

In other words, multiplying $\mathbf{S}$ by the rational matrix $\mathbf{W}$ (without any modular reduction), we have

$$\mathbf{S}\mathbf{W} = \mathbf{K}q + \mathbf{S}^* + \mathbf{E}, \tag{2}$$

for some integer matrix $\mathbf{K}$ and an error matrix $\mathbf{E}$ satisfying $\|\mathbf{E}\|_\infty \le \text{poly}(n)/q \ll 1$ (whp).

Functionality of W. Given a valid high-dimension ciphertext $\mathbf{c}^*$ relative to the secret key $\mathbf{S}^*$, we key-switch it to $\mathbf{S}$ by multiplying by $\mathbf{W}$, then rounding and reducing mod q. Namely, we set $\mathbf{c} = \lceil \mathbf{W}\mathbf{c}^* \rfloor_q$. This yields a valid low-dimension ciphertext that encrypt the same thing, but relative to $\mathbf{S}$. Namely, if $\mathbf{S}^*\mathbf{c}^* = \mathbf{k}^* q + \mathbf{b}(q/2) + \mathbf{e}^*$ for a bit vector $\mathbf{b}$ and integer vectors $\mathbf{k}^*, \mathbf{e}^*$ of magnitude much smaller than q, then also $\mathbf{S}\mathbf{c} = \mathbf{k}q + \mathbf{b}(q/2) + \mathbf{e}$ for the same $\mathbf{b}$ and where also the magnitude of $\mathbf{k}, \mathbf{e}$ is much smaller than q. The analysis is identical to that in Section 2.3.

Of course, there is nothing special about $\mathbf{S}^*$ and $\mathbf{S}$ above. For every two secret-key matrices $\mathbf{S}_1, \mathbf{S}_2$ we can similarly generate a key-switching gadget $W[\mathbf{S}_1 \to \mathbf{S}_2]$ to enable switching ciphertext relative to $\mathbf{S}_1$ into ciphertexts relative to $\mathbf{S}_2$.

Security of W. It is immediate to show that when $\mathbf{S}_2$ is drawn according to the error distribution χ and independently from $\mathbf{S}_1$, then the key-switching matrix $W[\mathbf{S}_1 \to \mathbf{S}_2]$ is pseudo-random, assuming the hardness of decision-LWE relative to error distribution χ and modulus $Q = 2^\ell q$. To see this, it is enough to observe that each vector $(\mathbf{d}_j|\mathbf{a}_j)$ (before the division by 2^ℓ) is pseudorandom in $\mathbb{Z}_Q$.

Lemma 1. *If the decision LWE problem with error distribution χ and modulus $Q = 2^\ell q$ is hard, then for a random secret key $\mathbf{S}_2 \leftarrow \chi^{m' \times m}$, the key-switching matrix $W[\mathbf{S}_1 \to \mathbf{S}_2]$ as above is indistinguishable from a uniformly random matrix with all the entries drawn independently at random from $[-q/2, q/2)$ with ℓ bits of precision to the right of the binary point. The indistinguishability holds even if the distinguisher gets as input the old secret key $\mathbf{S}_1$.*

Of course, the above lemma does not hold when $\mathbf{S}_1, \mathbf{S}_2$ are related, as in our case where each row of $\mathbf{S}^*$ is the tensor product of the corresponding row of $\mathbf{S}$ with itself. This issue is routinely addresses in one of two ways: One option is to construct a *leveled* HE scheme, where we choose many independent secret key matrices $\mathbf{S}_k$, $k = 1, 2, \ldots$, then put in the public key only the key-switching gadgets $W[\mathbf{S}_k^* \to \mathbf{S}_{k+1}]$. This requires that we switch to a new key after every multiplication, hence the multiplication depth of the circuits that we can handle is bounded by the number of key-switching gadgets in the public key. The other common option of dealing with this issue is to use related $\mathbf{S}_1, \mathbf{S}_2$ anyway, call it circular security, then wave our hands emphatically, saying that we think that the scheme remains secure nonetheless. (Indeed, there are no attackes in the literature that use this relation between $\mathbf{S}_1, \mathbf{S}_2$ to break the scheme.)

3.2 Moving Values Between Plaintext Slot

Using the techniques thus far we can implement SIMD-type homomorphic operations on packed ciphertexts, where the same function is applied to m' different inputs at once. However, Gentry et al. pointed out in [GHS12a] that more is needed if we are to apply these techniques for efficient evaluation of (wide enough) circuits on just one input. To take advantage of internal parallelism opportunities within a circuit, we must also be able to move values between different plaintext slots, so as to move a value from the output of one gate in level i of the circuit to the input of another gate in level $i + 1$.

Specifically, following [GHS12a] we seek an efficient procedure for permuting the plaintext slots according to any given permutation. Below we denote by $\pi(\mathbf{b})$ the permutation according to π of the entries of the vector $\mathbf{b}$ (i.e., the vector whose $\pi(i)$'th entry equals the i'th entry of $\mathbf{b}$). Similarly, for a matrix $\mathbf{S}$ we denote by $\pi(\mathbf{S})$ the permutation of the rows of $\mathbf{S}$ according to π. What we seek, therefore, is a transformation that given a ciphertext $\mathbf{c}$ encrypting a binary vector $\mathbf{b} \in \{0, 1\}^{m'}$ relative to $\mathbf{S}$, outputs another ciphertext $\mathbf{c}'$ encrypting the permuted vector $\pi(\mathbf{b})$ relative to the same $\mathbf{S}$.

The technique from above for key-switching can be easily adopted to this purpose. Indeed, it follows from the decryption formula that if $\mathbf{c}$ is a valid encryption of $\mathbf{b}$ relative to $\mathbf{S}$, then the same $\mathbf{c}$ is also a valid encryption of $\pi(\mathbf{b})$ relative to $\pi(\mathbf{S})$. All we need, therefore, is to switch $\mathbf{c}$ from the key $\pi(\mathbf{S})$ back to the key $\mathbf{S}$, which we can do if we have in the public key a key-switching gadget $\mathbf{W} = W[\pi(\mathbf{S}) \to \mathbf{S}]$. Namely, permuting the plaintext slots according to π is done by setting $\mathbf{c}' = \lceil \mathbf{W}\mathbf{s} \rfloor_q$.

3.3 Discussion

In this note we described a very simple method of computing on packed ciphertext in the PVW variant of Regev's cryptosystem. This provides an efficiency boost for LWE-based HE schemes, similar to the boost that we get by using the techniques of [SV11, GHS12a] in RLWE-based schemes.

We stress, however, that schemes over polynomial rings still offer much better asymptotic efficiency than schemes over the ring of integers. The size of ciphertexts in both cases is roughly the same (upto a constant factor), since ciphertexts in polynomial-ring schemes consist of a constant number of ring elements, each element represented by $O(n)$ integers. However, the tensor product multiplication increases the ciphertext size over the integers to $O(n^2)$ integers, whereas over polynomial rings we still only need $O(n)$ integers to describe even the product ciphertext. Even more, the re-linearizion gadget over the integers is a $O(n)$-by-$O(n^2)$ matrix, which takes $O(n^3)$ integers to represent. In the polynomial-ring setting, this matrix is still only an $O(1)$-by-$O(1)$ matrix over the ring, so it still only takes $O(n)$ integers to represent. As a consequence, whereas in the polynomial-ring setting [GHS12a] were able to reduce the overhead of homomorphic evaluation to only polylogarithmic factor (for wide enough circuits), using the same techniques over the integers (with the ciphertext-packing tools from the current paper) would yield a quasi-quadratic overhead.

On the other hand, the security of the integer schemes is based on the hardness of standard LWE, which is arguably better understood than the hardness of ring-LWE (or the NTRU problem) which underly the security of schemes over polynomial rings. In addition, the techniques that we described in this note are perhaps more flexible and less "algebraically heavy" than their polynomial-ring counterparts.

For example, for polynomial-ring schemes, the number of plaintext slots in each ciphertext is determined by the algebra of the underlying ring, and thus not every number of slots is achievable. For example, Gentry et al reported in [GHS12b] on homomorphic evaluation of the AES circuit, that used only 16 plaintext slots (for the 16 bytes in the AES state). However, security considerations dictated that the ring be very large, which resulted in several hundred unused plaintext slots. In contrast, the number of plaintext slots in PVW ciphertext packing is a free parameter that can be set to any desired value.

Perhaps the main advantage of the packed-ciphertext computation techniques for integer-based schemes over their counterparts for polynomial-ring schemes is the simplicity of implementing data movement over plaintext slots. In polynomial rings, these operations are implemented using automorphism, but only a small set of permutations (determined by the ring algebra) can be implemented this way. In [GHS12a] it was shown how to implement any data movement pattern using the small set that we get from the ring algebra, but that procedure requires a logarithmic number of basic automorphisms to implement a given data-movement pattern. In contrast, the technique from Section 3.2 lets us directly implement any data-movement pattern, just by putting in the public key the corresponding key-switching gadget. Hence if we know ahead of time the circuit that we want to evaluate homomorphically (e.g., the AES circuit), we can prepare the corresponding gadgets to enable computing the data-movement patterns of that circuit directly. Of course, if we do not know the circuit ahead of time, we can put in the public key the gadgets for just a few permutations, and then use the techniques from [GHS12a] to implement arbitrary patterns, incurring the same logarithmic slowdown.

Another thing which is easier to do in integer-based schemes than in polynomial-based scheme is to gradually reduce the dimension as the computation progresses: Fresh ciphertexts in all these schemes must have a very large modulus/noise ratio to enable computation, which implies that we need fairly high dimension (or fairly high ring-size) to get security. However, larger noise (and hence smaller modulus/noise ratio) is used as the computation progresses, so from a security standpoint it is permissible to switch to lower dimension (or smaller ring), thus speeding up further homomorphic operations. Recently, it was shown in [GHPS12] how to do this for schemes in polynomial rings, but this operation is highly constrained by the relevant algebra. Specifically, if the dimension of the initial ring is some m, then it is only possible to switch to other rings of dimension that divides m. In particular it means that the first time we can perform this transformation is when it is safe to switch to a ring of size $m/2$ (or less), which means that at least half the computation has to be done relative to the original large ring. In contrast, switching to a lower dimension is nearly trivial in LWE-based schemes: All we need is key-switching from the initial dimension-n key to a lower-dimension key, exactly as is done for re-linearization (with the noise magnitude in the key-switching gadget increased to provide adequate security relative to the lower dimension).

Finally, we mansion that the techniques described in this note can also be used in conjunction with HE schemes over polynomial rings (with security based on the "general LWE" problem), as suggested, e.g., in [BGV12]. This setting offers a tradeoff, with schemes over the integers on one end and schemes over large polynomial rings on the other. In the middle we have schemes over smaller polynomial rings, in which ciphertexts are low-dimension vectors over these rings. (For example, the ring may have dimension $n/\log n$, and then ciphertexts and secret key can have dimension $O(\log n)$ over that ring.) This opens yet another avenue for tradeoffs and optimizations, for example we can choose the ring with best algebraic properties, even if it is too small to provide the security level that we seek, then use slightly longer vectors over that ring to recover security. In this context, it is possible to use both ciphertext packing techniques: pack many plaintext values relative to each secret-key vector using the polynomial-CRT techniques, and use many secret key vectors to reduce the ciphertext expansion ratio as described in this work.

References

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO'09*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618. Springer, 2009.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. In *Innovations in Theoretical Computer Science (ITCS'12)*, 2012. Available at `http://eprint.iacr.org/2011/277`.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP, 2012. `http://eprint.iacr.org/2012/078`.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS'11*. IEEE Computer Society, 2011.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *Advances in Cryptology - CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, 2011.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC'09*, pages 169–178. ACM, 2009.

[GHPS12] Craig Gentry, Shai Halevi, Chris Peikert, and Nigel Smart. Ring switching in bgv-style homomorphic encryption. In *SCN'12*, volume 7485 of *Lecture Notes in Computer Science*, pages 19–37. Springer, 2012. Full version at `http://eprint.iacr.org/2012/240`.

[GHS12a] Craig Gentry, Shai Halevi, and Nigel Smart. Fully homomorphic encryption with polylog overhead. In *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 465–482. Springer, 2012. Full version at `http://eprint.iacr.org/2011/566`.

[GHS12b] Craig Gentry, Shai Halevi, and Nigel Smart. Homomorphic evaluation of the AES circuit. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867. Springer, 2012. Full version at `http://eprint.iacr.org/2012/099`.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *STOC'09*, pages 333–342. ACM, 2009.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO'08*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2008.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.

[Rot11] Ron Rothblum. Homomorphic encryption: From private-key to public-key. In *TCC'11*, volume 6597 of *Lecture Notes in Computer Science*, pages 219–234. Springer, 2011.

[SV11] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. Manuscript at http://eprint.iacr.org/2011/133, 2011.