Scale-Invariant Fully Homomorphic Encryption over the Integers

Jean-Sébastien Coron[1], Tancrède Lepoint[1,2,3], and Mehdi Tibouchi[4]

[1] University of Luxembourg, Luxembourg
jean-sebastien.coron@uni.lu
[2] École Normale Supérieure, France
[3] CryptoExperts, France
tancrede.lepoint@cryptoexperts.com
[4] NTT Secure Platform Laboratories, Japan
tibouchi.mehdi@lab.ntt.co.jp

Abstract. At Crypto 2012, Brakerski constructed a scale-invariant fully homomorphic encryption scheme based on the LWE problem, in which the same modulus is used throughout the evaluation process, instead of a ladder of moduli when doing "modulus switching". In this paper we describe a variant of the van Dijk et al. FHE scheme over the integers with the same scale-invariant property. Our scheme has a single secret modulus whose size is linear in the multiplicative depth of the circuit to be homomorphically evaluated, instead of exponential; we therefore construct a leveled fully homomorphic encryption scheme. This scheme can be transformed into a pure fully homomorphic encryption scheme using bootstrapping, and its security is still based on the Approximate-GCD problem.

We also describe an implementation of the homomorphic evaluation of the full AES encryption circuit, and obtain significantly improved performance compared to previous implementations: about 23 seconds (resp. 3 minutes) per AES block at the 72-bit (resp. 80-bit) security level on a mid-range workstation.

Finally, we prove the equivalence between the (error-free) decisional Approximate-GCD problem introduced by Cheon et al. (Eurocrypt 2013) and the classical computational Approximate-GCD problem. This equivalence allows to get rid of the additional noise in all the integer-based FHE schemes described so far, and therefore to simplify their security proof.

1 Introduction

Fully Homomorphic Encryption. In 2009, Gentry constructed the first fully homomorphic encryption scheme (FHE), i.e. a scheme allowing a worker to evaluate any circuit on plaintext values while manipulating only ciphertexts. The first generation of FHE schemes [Gen09,DGHV10,SV10,GH11,BV11a,BV11b] and [CMNT11,CNT12,CCK$^+$13] followed Gentry's blueprint to achieve a fully homomorphic scheme.

The first step of Gentry's blueprint is to construct a somewhat homomorphic encryption scheme (SWHE) capable of evaluating "low degree" polynomials homomorphically. Inherent to this construction is the property that ciphertexts are "noisy", and noises grow slightly with homomorphic additions and substantially with homomorphic multiplications. Thus ciphertexts need to be refreshed to maintain a low noise level and allow subsequent homomorphic operations. To obtain a FHE scheme, Gentry's key-idea, referred to as *bootstrapping*, states that a SWHE capable of evaluating its own decryption procedure (and an additional multiplication) can be transformed into a FHE scheme. Bootstrapping consists in evaluating the decryption circuit of the SWHE scheme using the decryption key bits in encrypted form, thus resulting in a different encryption of the same plaintext but with reduced noise. In practice, the scheme parameters are generally determined so that the refreshed ciphertexts can handle one additional homomorphic multiplication [GH11,CMNT11,CNT12,CCK$^+$13]. Unfortunately, the downside of these settings is that one needs to call the (very costly) bootstrapping procedure after each homomorphic multiplication.

Modulus Switching and Scale Invariance. To avoid bootstrapping a new noise management technique, called *modulus switching*, was introduced by Brakerski, Gentry and Vaikuntanathan [BGV12]. The authors obtained a *leveled* FHE scheme: i.e. a scheme in which the noise grows *linearly* with the multiplicative depth instead of exponentially as in somewhat homomorphic encryption. Therefore any circuit with polynomial depth can be evaluated. The technique consists in scaling down the noise by converting a ciphertext modulo q into a ciphertext modulo a smaller q'; the noise being reduced by roughly a factor q/q'. By carefully calibrating the ladder of moduli, the noise growth can then be made linear with the number of homomorphic multiplications. The technique was also adapted to the DGHV fully homomorphic encryption scheme over the integers [DGHV10] in [CNT12]. Unfortunately for a circuit with L layers of multiplication, the technique requires to store the equivalent of L public-keys, yielding a huge storage requirement.

At Crypto 2012, Brakerski introduced a new tensor product technique for LWE-based leveled FHE [Bra12] so that the *same* modulus is used throughout the evaluation process instead of a layer of moduli; the noise growth is still linear in the number of homomorphic multiplications. This was achieved by considering ciphertexts such that $\langle \boldsymbol{c}, \boldsymbol{s} \rangle = \lfloor q/2 \rfloor \cdot m + e \bmod q$, instead of $\langle \boldsymbol{c}, \boldsymbol{s} \rangle = m + 2e \bmod q$, as in Regev's initial scheme [Reg05].

Implementations of FHE Schemes. Independently at Crypto 2012, Gentry et al. benchmarked a LWE-based scheme by homomorphically evaluating an AES circuit [GHS12b], yielding to the first "real-world" circuit homomorphically evaluated by a FHE scheme. This implementation used the modulus switching technique of [BGV12] and additionally a batching technique [SV11,BGV12,GHS12a] that allows one to encrypt vectors of plaintexts in a single ciphertext, and to perform any permutation on the underlying plaintext vector while manipulating only the ciphertext. They obtained a timing of about 5 minutes *per AES block* homomorphically encrypted. Similar results were later obtained for the integer-based DGHV scheme [DGHV10], extending the batching technique and homomorphically evaluating AES on a desktop computer in about 12 minutes per block for 72 bits of security [CCK+13,CLT13].

Our Contributions. In this paper, we describe a variant of the DGHV scheme over the integers with the same scale-invariant property as in [Bra12]; i.e. our scheme does not use modulus switching and the noise grows linearly with the multiplicative depth. We obtain a DGHV variant with a single secret modulus p whose size is linear in the multiplicative depth (instead of exponential). Our technique is as follows.

In the original DGHV scheme, a ciphertext c of the bit message $m \in \{0, 1\}$ has the form

$$c = m + 2r + q \cdot p \ ,$$

where p is the secret key, q is a large random integer, and r is a small random integer (noise). The bit message is recovered by computing $m = (c \bmod p) \bmod 2$. Adding and multiplying ciphertexts over $\mathbb{Z}$ respectively adds and multiplies the plaintexts modulo 2 while keeping them hidden. Unfortunately, the noise grows exponentially with the number of homomorphic multiplications: if two ciphertexts c_1, c_2 have ρ-bit noise, the noise of $c_3 = c_1 \cdot c_2$ has $\approx 2\rho$ bits. Therefore to evaluate a circuit with L sequential layers of multiplications without bootstrapping, the bit-size η of the modulus p must satisfy $\eta > 2^L \rho$.

In our new scheme, similar to [Bra12], instead of encrypting the bit $m \in \{0, 1\}$ in the LSB of $[c \bmod p]$, we encrypt it in the MSB of $[c \bmod p]$; additionally we work modulo p^2 instead of modulo p. More precisely, the message m is now encrypted as

$$c = r + (m + 2r^*) \cdot \frac{p-1}{2} + q \cdot p^2 \ , \tag{1}$$

where the ciphertext now contains two noises r and r^*. We decrypt c by computing $m = (2c \bmod p) \bmod 2$. Clearly adding two ciphertexts over $\mathbb{Z}$ still adds the underlying bit messages m modulo 2. However, multiplication of two ciphertexts moves the bit message m from the MSB of $[c \bmod p]$ to the MSB of $[c \bmod p^2]$. Namely, a ciphertext c obtained as the multiplication of ciphertexts c_1 and c_2 for the respective bit messages m_1 and m_2 will have the form

$$c = 2 \cdot c_1 \cdot c_2 = r + (m_1 \cdot m_2) \cdot \frac{p^2 - 1}{2} + q \cdot p^2 \ , \tag{2}$$

where $r > p$ but still $r \ll p^2$. We then describe a procedure $\mathsf{Convert}$ that allows to publicly convert the result of a multiplication (i.e. a ciphertext as in Equation (2)) into a ciphertext reusable in subsequent homomorphic operations (i.e. a ciphertext as in Equation (1)), either keeping the same secret p (which requires, as usual, a circular security assumption) or using a different fresh p at each level (which requires a larger secret key). The bit length of the noise

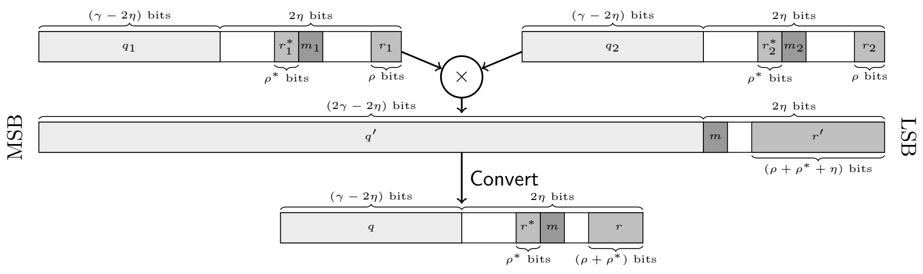

Fig. 1. Conversion of a ciphertext after a homomorphic multiplication

in the new ciphertext grows only by a constant additive factor with respect to the noise in c_1 and c_2 (see Figure 1 for an illustration). Therefore, our scheme is a variant of the DGHV scheme that is a leveled fully homomorphic encryption scheme. It can be turned into a pure FHE scheme using bootstrapping (cf. [DGHV10,CMNT11,CNT12,CCK+13]). We also show that our scheme is semantically secure, under the Approximate-GCD assumption.

We also adapt our scale-invariant technique to the batch setting in [CCK+13] and homomorphically evaluate an AES encryption as in [GHS12b,CCK+13]. Our scheme offers competitive performances as it can evaluate the full AES circuit in about 23 seconds (resp. 3 minutes) per AES block at the 72-bit (resp. 80-bit) security level on a mid-range workstation, that is one order of magnitude faster than [CCK+13].

Finally, we prove the equivalence between the (error-free) computational Approximate-GCD problem [DGHV10] and the (error-free) decisional Approximate-GCD problem introduced in [CCK+13,KLYC13]. From this equivalence, the additional noise added during encryption to drawn the noises coming from the public key elements is no longer required. This yields automatic improvements in the parameters of all the fully homomorphic encryption schemes over the integers.

2 The Somewhat Homomorphic DGHV Scheme

In this section we first recall the somewhat homomorphic encryption scheme over the integers of van Dijk, Gentry, Halevi and Vaikuntanathan (DGHV) in [DGHV10]. We denote by λ the security parameter, τ the number of elements in the public key, γ their bit-length, η the bit-length of the secret key p and ρ (resp. ρ') the bit-length of the noise in the public key (resp. in a fresh ciphertext).

For a real number x, we denote by $\lceil x \rceil$, $\lfloor x \rfloor$ and $\lceil x \rfloor$ the upper, lower or nearest integer part of x. For integers z, p we denote the reduction of z modulo p by $(z \bmod p)$ or $[z]_p$ with $-p/2 < [z]_p \leqslant p/2$. For a specific η-bit odd integer p, we use the following distribution over γ-bit integers:

$$\mathcal{D}_{\gamma,\rho}(p) = \left\{ \begin{array}{c} \text{Choose } q \leftarrow \mathbb{Z} \cap [0, 2^\gamma/p),\ r \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho) : \\ \text{Output } x = q \cdot p + r \end{array} \right\}.$$

$\mathsf{DGHV.KeyGen}(1^\lambda)$**.** Generate an η-bit random prime integer p. For $0 \leqslant i \leqslant \tau$, sample $x_i \leftarrow \mathcal{D}_{\gamma,\rho}(p)$. Relabel the x_i's so that x_0 is the largest. Restart unless x_0 is odd and $[x_0]_p$ is even. Let $\mathsf{pk} = (x_0, x_1, \ldots x_\tau)$ and $\mathsf{sk} = p$.

$\mathsf{DGHV.Encrypt}(\mathsf{pk}, m \in \{0,1\})$**.** Choose a random subset $S \subseteq \{1, 2, \ldots, \tau\}$ and a random integer r in $(-2^{\rho'}, 2^{\rho'})$, and output the ciphertext:

$$c = \left[m + 2r + 2 \sum_{i \in S} x_i \right]_{x_0} . \tag{3}$$

$\mathsf{DGHV.Evaluate}(\mathsf{pk}, C, c_1, \ldots, c_t)$**.** Given the circuit C with t input bits and t ciphertexts c_i, apply the addition and multiplication gates of C to the ciphertexts, performing all the additions and multiplications over the integers, and return the resulting integer.

$\mathsf{DGHV.Decrypt}(\mathsf{sk}, c)$**.** Output $m \leftarrow (c \bmod p) \bmod 2$.

This completes the description of the scheme. The scheme is clearly somewhat homomorphic, i.e. a limited number of homomorphic operations can be performed on ciphertexts. More precisely given two ciphertexts $c = q \cdot p + 2r + m$ and $c' = q' \cdot p + 2r' + m'$ where r and r' are ρ'-bit integers, the ciphertext $c + c'$ is an encryption of $m + m' \bmod 2$ with a $(\rho' + 1)$-bit noise and the ciphertext $c \cdot c'$ is an encryption of $m \cdot m'$ with noise bit-length $\simeq 2\rho'$. Therefore the scheme allows roughly η/ρ' successive multiplications on ciphertexts (since the noise must remain smaller than p for correct decryption).

As shown in [DGHV10] the scheme is semantically secure under the Approximate-GCD assumption.

Definition 1 (Approximate GCD). *The (ρ, η, γ)-Approximate-GCD problem consists, given a random η-bit odd integer p and given polynomially many samples from $\mathcal{D}_{\gamma,\rho}(p)$, in outputting p.*

3 Scale-Invariant DGHV Scheme

In this section we describe our variant of the DGHV scheme with the scale-invariant property. We first explain the two main ideas of our scheme, namely 1) moving the plaintext bit from the LSB to the MSB of $[c \bmod p]$ and working modulo p^2, and 2) converting the result of a ciphertext multiplication back to a ciphertext usable in subsequent homomorphic operations. We then provide the full description of our scheme.

3.1 Ciphertexts and Homomorphic Operations

As explained in introduction, instead of encrypting the plaintext $m \in \{0,1\}$ in the LSB of $[c \bmod p]$, m is now encrypted in the MSB of $[c \bmod p]$ as

$$c = r + (m + 2r^*) \cdot \frac{p-1}{2} + q \cdot p^2 , \tag{1}$$

where the ciphertext has now two noises r and r^* of respective bit-length ρ and ρ^*. We call such ciphertext a *Type-I ciphertext* and we say that c has noise length (ρ, ρ^*). To decrypt c, one computes $(2c \bmod p) \bmod 2 = m$.

Homomorphic additions are performed as additions over $\mathbb{Z}$: namely given two Type-I ciphertexts c_1 and c_2 of noise (ρ, ρ^*):

$$c_1 = r_1 + (m_1 + 2r_1^*) \cdot (p-1)/2 + q_1 \cdot p^2$$
$$c_2 = r_2 + (m_2 + 2r_2^*) \cdot (p-1)/2 + q_2 \cdot p^2$$

we get

$$c_1 + c_2 = r_3 + (m_1 + m_2 + 2r_3^*) \cdot \frac{p-1}{2} + q_3 \cdot p^2 ,$$

for some integers r_3, r_3^* and q_3, with $\log_2 |r_3| \leqslant \rho + 1$ and $\log_2 |r_3^*| \leqslant \rho^* + 1$.

Next, to homomorphically multiply the ciphertexts c_1 and c_2, one computes $c_3 = 2 \cdot c_1 \cdot c_2$ over $\mathbb{Z}$. This gives

$$\begin{aligned} c_3 = 2 \cdot c_1 \cdot c_2 = 2r_1r_2 + \big(r_1(m_2 + 2r_2^*) + r_2(m_1 + 2r_1^*)\big) \cdot (p-1) + \\ (m_1 + 2r_1^*) \cdot (m_2 + 2r_2^*) \cdot \frac{(p-1)^2}{2} + q_3' \cdot p^2 \\ = r_3' + (m_1 + 2r_1^*) \cdot (m_2 + 2r_2^*) \cdot \frac{(p-1)^2}{2} + q_3' \cdot p^2 \end{aligned}$$

for some integers q_3' and r_3', with $\log_2 |r_3'| \leqslant \eta + \rho + \rho^* + 3$, where η is the bit-size of p. We use $\eta \gg \rho, \rho^*$. Then, there exist integers r_3 and q_3 such that

$$c_3 = r_3 + m_3 \cdot \frac{p^2 - 1}{2} + q_3 \cdot p^2 , \tag{2}$$

where $m_3 = m_1 \cdot m_2$. We call an integer c verifying Equation (2) a *Type-II ciphertext*. The bit-length of noise r_3 satisfies $\log_2 |r_3| \leqslant \eta + \rho + \rho^* + 4$, assuming $\rho^* < \rho$. We refer to Figure 1 for a graphical representation of the homomorphic multiplication.

3.2 Conversion from Type-II Ciphertext to Type-I Ciphertext

We show that we can efficiently convert a Type-II ciphertext back to a Type-I ciphertext, using only the public-key. Our procedure Convert uses essentially the same technique as the modulus switching technique for DGHV in [CNT12]. Namely modulus switching in [CNT12] enables to convert a classical DGHV ciphertext modulo a prime p into a new ciphertext modulo a prime p', with noise scaled by a factor p'/p. Similarly, our Convert procedure converts a Type-II ciphertext modulo p^2 back to a ciphertext where the noise is modulo p (therefore the noise is scaled by a factor $p/p^2 = 1/p$), but still somehow encrypted modulo p^2.

More precisely, we start from a Type-II ciphertext:

$$c = r + \frac{p^2 - 1}{2} \cdot m + q \cdot p^2 \tag{4}$$

where $|r| \leqslant 2^{\rho'}$. Let κ be such that $|c| < 2^{\kappa}$. Let $\boldsymbol{z}$ be a vector of Θ rational numbers in $[0, 2^{\eta})$ with κ bits of precision after the binary point, and let $\boldsymbol{s}$ be a vector of Θ bits such that

$$\frac{2^{\eta}}{p^2} = \langle \boldsymbol{s}, \boldsymbol{z} \rangle + \varepsilon \mod 2^{\eta} , \tag{5}$$

where $|\varepsilon| \leqslant 2^{-\kappa}$. Here Θ is a parameter to be chosen later for security. We use the same $\mathsf{BitDecomp}$ and $\mathsf{PowersofTwo}$ procedures as in [BGV12].

– $\mathsf{BitDecomp}_{\eta}(\boldsymbol{v})$: For $\boldsymbol{v} \in \mathbb{Z}^n$, let $\boldsymbol{v}_i \in \{0,1\}^n$ be such that $\boldsymbol{v} \bmod 2^{\eta} = \sum_{i=0}^{\eta-1} \boldsymbol{v}_i \cdot 2^i$. Output the vector

$$(\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{\eta-1}) \in \{0,1\}^{n \cdot \eta} .$$

– $\mathsf{PowersofTwo}_{\eta}(\boldsymbol{w})$: For $\boldsymbol{w} \in \mathbb{Z}^n$, output the vector

$$(\boldsymbol{w}, 2 \cdot \boldsymbol{w}, \ldots, 2^{\eta-1} \cdot \boldsymbol{w}) \in \mathbb{Z}^{n \cdot \eta} .$$

Given the vector $\boldsymbol{s}$ from (5), we let $\boldsymbol{s'} = \mathsf{PowersofTwo}_{\eta}(\boldsymbol{s})$, and let

$$\boldsymbol{\sigma} = \boldsymbol{q} \cdot p^2 + \boldsymbol{r} + \left\lfloor \boldsymbol{s'} \cdot \frac{p}{2^{\eta+1}} \right\rceil \tag{6}$$

be an "encryption" of the vector $\boldsymbol{s'}$, where $\boldsymbol{q} \leftarrow (\mathbb{Z} \cap [0, 2^{\gamma}/p^2))^{\eta \cdot \Theta}$ and $\boldsymbol{r} \leftarrow (\mathbb{Z} \cap (-2^{\rho}, 2^{\rho}))^{\eta \cdot \Theta}$. We can now define the $\mathsf{Convert}$ algorithm:

$\mathsf{Convert}(\boldsymbol{z}, \boldsymbol{\sigma}, c)$**.** First compute $\boldsymbol{c} = (\lfloor c \cdot z_i \rceil \bmod 2^{\eta})_{1 \leqslant i \leqslant \Theta}$ and its decomposition $\boldsymbol{c'} = \mathsf{BitDecomp}_{\eta}(\boldsymbol{c})$. Finally, output

$$c' \leftarrow 2 \langle \boldsymbol{\sigma}, \boldsymbol{c'} \rangle .$$

The following Lemma shows that our procedure $\mathsf{Convert}$ enables one to transform a Type-II ciphertext back to a Type-I ciphertext. We provide the proof in the full version of the paper [CLT14].

Lemma 1. *Let ρ' be such that $\rho' \geqslant \eta + \rho + \log_2(\eta\Theta)$. The procedure* $\mathsf{Convert}$ *above converts a Type-II ciphertext with noise size ρ' into a Type-I ciphertext with noise $(\rho' - \eta + 5, \log_2 \Theta)$.*

Assume that initially the two ciphertexts c_1, c_2 are Type-I ciphertexts with noise $(\rho_1, \log_2 \Theta)$. After computing $c_3 = 2 \cdot c_1 \cdot c_2$ which has noise size at most $\rho' = \eta + \rho_1 + \log_2 \Theta + 4$ (see previous section) one can convert c_3 back into a Type-I ciphertext with noise (ρ_3, ρ_3^*) with $\rho_3 = \rho_1 + \log_2 \Theta + 9$ and $\rho_3^* = \log_2 \Theta$, from Lemma 1. Therefore the noise length in bits has only grown by an additive factor $\log_2 \Theta + 9$. Therefore the ciphertext noise grows only linearly with the number of homomorphic multiplications.

3.3 Description of the Public-Key Leveled Fully Homomorphic Scheme

We are now ready to describe our scale-invariant version of the DGHV encryption scheme. For a specific η-bit odd integer p and an integer q_0 in $[0, 2^\gamma/p^2)$, we define the set:

$$\mathcal{D}^\rho_{p,q_0} = \{q \cdot p^2 + r : q \in \mathbb{Z} \cap [0, q_0), r \in \mathbb{Z} \cap (-2^\rho, 2^\rho)\} .$$

$\mathsf{SIDGHV.KeyGen}(1^\lambda)$. Generate an odd η-bit integer p and a γ-bit integer $x_0 = q_0 \cdot p^2 + r_0$ with $r_0 \leftarrow (-2^\rho, 2^\rho) \cap \mathbb{Z}$ and $q_0 \leftarrow [0, 2^\gamma/p^2) \cap \mathbb{Z}$. Let $x_i \leftarrow \mathcal{D}^\rho_{p,q_0}$ for $1 \leqslant i \leqslant \tau$. Let also $y' \leftarrow \mathcal{D}^\rho_{p,q_0}$ and $y = y' + (p-1)/2$.
Let $\boldsymbol{z}$ be a vector of Θ numbers with $\kappa = 2\gamma + 2$ bits of precision after the binary point, and let $\boldsymbol{s}$ be a vector of Θ bits such that

$$\frac{2^\eta}{p^2} = \langle \boldsymbol{s}, \boldsymbol{z} \rangle + \varepsilon \bmod 2^\eta,$$

with $|\varepsilon| \leqslant 2^{-\kappa}$. Now, define

$$\boldsymbol{\sigma} = \boldsymbol{q} \cdot p^2 + \boldsymbol{r} + \left\lfloor \mathsf{PowersofTwo}_\eta(\boldsymbol{s}) \cdot \frac{p}{2^{\eta+1}} \right\rceil ,$$

where the components of $\boldsymbol{q}$ (resp. $\boldsymbol{r}$) are randomly chosen from $[0, q_0) \cap \mathbb{Z}$ (resp. $(-2^\rho, 2^\rho) \cap \mathbb{Z}$).
The secret-key is $\mathsf{sk} = \{p\}$ and the public-key is $\mathsf{pk} = \{x_0, x_1, \ldots, x_\tau, y, \boldsymbol{\sigma}, \boldsymbol{z}\}$.

$\mathsf{SIDGHV.Encrypt}(\mathsf{pk}, m \in \{0,1\})$. Choose a random subset $S \subset \{1, \ldots, \tau\}$ and output

$$c \leftarrow \left[m \cdot y + \sum_{i \in S} x_i \right]_{x_0} .$$

$\mathsf{SIDGHV.Add}(\mathsf{pk}, c_1, c_2)$. Output $c \leftarrow c_1 + c_2 \bmod x_0$.

$\mathsf{SIDGHV.Convert}(\mathsf{pk}, c)$. Output $c' \leftarrow 2 \cdot \big\langle \boldsymbol{\sigma}, \mathsf{BitDecomp}_\eta(\boldsymbol{c}) \big\rangle$ where $\boldsymbol{c} = \big(\lfloor c \cdot z_i \rceil \bmod 2^\eta\big)_{1 \leqslant i \leqslant \Theta}$.

$\mathsf{SIDGHV.Mult}(\mathsf{pk}, c_1, c_2)$. Output $c' \leftarrow \mathsf{SIDGHV.Convert}(\mathsf{pk}, 2 \cdot c_1 \cdot c_2) \bmod x_0$.

$\mathsf{SIDGHV.Decrypt}(\mathsf{sk}, c)$. Output $m \leftarrow \big((2c) \bmod p\big) \bmod 2$.

Remark 1. This describes a *leveled* fully homomorphic encryption scheme, because the noise growth is only linear in the number of levels. The scheme can be bootstrapped to obtain a (pure) fully homomorphic encryption scheme, as in [DGHV10,CCK+13],

3.4 Constraints on the Parameters

The parameters of the scheme must meet the following constraints (where λ is the security parameter):

- $\rho = \Omega(\lambda)$ to avoid brute force attack on the noise [CN12,CNT12],
- $\eta \geqslant \rho + \mathcal{O}(L \log \lambda)$ where L is the multiplicative depth of the circuit to be evaluated,
- $\gamma = \omega(\eta^2 \cdot \log \lambda)$ in order to thwart lattice-based attacks (see [DGHV10] and [CMNT11,CH12]),
- $\Theta^2 = \gamma \cdot \omega(\log \lambda)$ to avoid lattice attacks on the subset sum (see [CMNT11]),
- $\tau \geqslant \gamma + 2\lambda$ in order to apply the Leftover Hash Lemma (see Section 3.5).

To satisfy the above constraints one can take $\rho = 2\lambda$, $\eta = \tilde{\mathcal{O}}(L + \lambda)$, $\gamma = \tilde{\mathcal{O}}(L^2\lambda + \lambda^2)$, $\Theta = \tilde{\mathcal{O}}(L\lambda)$ and $\tau = \gamma + 2\lambda$.

3.5 Semantic Security

We show that the semantic security of our scheme can be based on the following variant of the decisional problem introduced in [KLYC13], called the Decisional-Approximate-GCD problem. Roughly speaking, it should be hard to distinguish integers from $\mathcal{D}^{\rho}_{p,q_0}$ from completely uniform integers modulo x_0, where:

$$\mathcal{D}^{\rho}_{p,q_0} = \{q \cdot p^2 + r : q \in \mathbb{Z} \cap [0, q_0), r \in \mathbb{Z} \cap (-2^{\rho}, 2^{\rho})\} .$$

Definition 2 ((ρ, η, γ)-Decisional-Approximate-GCD). *Let p be a random odd integer of η bits, q_0 an integer uniformly distributed in $[0, 2^{\gamma}/p^2)$, r_0 an integer uniformly distributed in $(-2^{\rho}, 2^{\rho})$. Given $x_0 = q_0 \cdot p^2 + r_0$, polynomially many samples from $\mathcal{D}^{\rho}_{p,q_0}$ and $y \leftarrow \mathcal{D}^{\rho}_{p,q_0} + (p-1)/2$, determine $b \in \{0,1\}$ from $c = x + b \cdot r \bmod x_0$ where $x \leftarrow \mathcal{D}^{\rho}_{p,q_0}$ and $r \leftarrow [0, x_0) \cap \mathbb{Z}$.*

The following theorem shows that our scheme is semantically secure under the Decisional-Approximate-GCD assumption; below we only consider a subset of our scheme without the procedure Convert, i.e. without the public parameters $\boldsymbol{z}$ and $\boldsymbol{\sigma}$. To prove the semantic security of the full scheme it suffices to include $\boldsymbol{z}$ and $\boldsymbol{\sigma}$ in the above decisional assumption.[5]

[5] Usually in FHE we first show the semantic security of a restricted scheme, and then a 'circular security' assumption is used to get the semantic security of the entire FHE; that is we assume that the encryption scheme remains secure even when the adversary is given encryptions of the individual bits of the private-key.

Here we first prove that the scheme is secure without the terms $\boldsymbol{z}$ and $\boldsymbol{\sigma}$. If the scheme is 'circular secure' (secure even with encryptions of the invariant switching, i.e. $\boldsymbol{z}$ and $\boldsymbol{\sigma}$) then it remains semantically secure. This circular security assumption can be avoided by using the classical modulus switching technique [CNT12] instead of our scale-invariance technique.

Theorem 1. *The above scale-invariant DGHV scheme without the parameters $\boldsymbol{z}$, $\boldsymbol{\sigma}$ is semantically secure under the (ρ, η, γ)-Decisional-Approximate-GCD assumption.*

To prove the theorem, we use a preliminary Lemma from [KLYC13] stating that the distribution of the public-key elements is indistinguishable from random elements in $[0, x_0)$ if the Decisional-Approximate-GCD problem is hard; the proof follows from a standard hybrid argument.

Lemma 2. *For the parameters (ρ, η, γ), let $\mathsf{pk} = (x_0, \{x_i\}_i, y)$ and $\mathsf{sk} = p$ be chosen as in the* KeyGen *procedure. Define $\mathsf{pk}' = (x_0, \{x_i'\}_i, y)$ for x_i' uniformly generated in $[0, x_0)$. Then* pk *and* pk′ *are indistinguishable under the Decisional-Approximate-GCD assumption.*

Proof (of Theorem 1). Under the attack scenario the attacker first receives the public key, and an encryption of a random bit $b \in \{0, 1\}$. The attacker outputs a guess b' and succeeds if $b' = b$. We use a sequence of games and denote by S_i the event that the attacker succeeds in $\mathbf{Game}_i$.

$\mathbf{Game}_0$: This is the attack scenario. We simulate the challenger by running KeyGen to obtain pk and sk.

$\mathbf{Game}_1$: We replace the x_i's in the public key by elements uniformly drawn in $[0, x_0)$. By Lemma 2, we have

$$|\Pr[S_1] - \Pr[S_0]| \leqslant \tau \cdot \varepsilon_{\text{dagcd}} \, .$$

$\mathbf{Game}_2$: By the Leftover Hash Lemma (Lemma 5 in Appendix A), $\sum_{i \in S} x_i \bmod x_0$ is ε-statistically indistinguishable from uniform modulo x_0, with $\varepsilon = 2^{(\gamma-\tau)/2}$. Therefore we can replace the challenge ciphertext by a uniform integer modulo x_0; this no longer gives any information on b and therefore $\Pr[S_2] = 1/2$. Moreover we have $|\Pr[S_2] - \Pr[S_1]| \leqslant \varepsilon$. This gap can be made negligible by satisfying the constraints on the parameters from Section 3.4, which concludes the proof. □

Remark 2. We show in Section 6 that the (Error-Free) Decisional-Approximate-GCD problem is equivalent to the computational (Error-Free) Approximate-GCD problem. Thus our scheme is automatically based on the *computational* Approximate-GCD problem as in previous works on the DGHV schemes [DGHV10,CMNT11,CNT12].

4 Generalization to Batch Scale-Invariant DGHV Scheme

We now describe a generalization of the previous scheme to the batch setting (as in RLWE-based schemes [BV11a,BV11b] and integer schemes [CCK$^+$13]). The goal is to pack ℓ plaintext bits $m_0, \ldots, m_{\ell-1}$ into a single ciphertext. Homomorphic addition and multiplication will then apply in parallel and component-wise on the m_i's.

Our batch generalization is similar to [CCK+13]. A ciphertext encrypting a vector $\boldsymbol{m} = (m_0, \ldots, m_{\ell-1})$ has the form:

$$c = \mathsf{CRT}_{q_0, p_0^2, \ldots, p_{\ell-1}^2} \left(q, \ldots, r_i + (2r_i^* + m_i) \cdot \frac{p_i - 1}{2}, \ldots \right) \tag{7}$$

for a tuple of $\ell+1$ coprime integers $q_0, p_0, \ldots, p_{\ell-1}$, where we denote by $\mathsf{CRT}_{b_i}(a_i)$ the unique integer u such that $0 \leqslant u < \prod_i b_i$ and $u \bmod b_i = a_i$ for all i. We call such ciphertext a *batch Type-I ciphertext*. Modulo each of the p_j's the ciphertext c behaves as in the SIDGHV scheme in Section 3. Accordingly, the addition of two ciphertexts yields a new ciphertext that decrypts to the componentwise sum modulo 2 of the original plaintexts.

To homomorphically multiply two ciphertexts c_1 and c_2, as previously one computes $c_3 = 2 \cdot c_1 \cdot c_2$ in $\mathbb{Z}$. As previously there exists small integers $r_{3,j}$ such that

$$c_3 \equiv r_{3,j} + m_j \cdot \frac{p_j^2 - 1}{2} \pmod{p_j} \quad \text{for } j = 0, \ldots, \ell - 1, \tag{8}$$

where each m_j is the product of the corresponding plain text components of c_1 and c_2. We call c_3 a *batch Type-II ciphertext*. Modulo each of the p_j's, the ciphertext c_3 behaves as a Type-II ciphertext given by Equation (2); therefore the message bit m_j is the MSB of $[c \bmod p_j^2]$ for all j. As in Section 3, there exists an efficient conversion procedure $\mathsf{Convert}$ to convert any Type-II ciphertext to a new Type-I ciphertext. As shown below the procedure $\mathsf{Convert}$ is actually the same as in Section 3, with adapted public parameters.

Namely let $\boldsymbol{z}$ be a vector of Θ rational numbers in $[0, 2^\eta)$ with κ bits of precision after the binary point (where $|c| < 2^\kappa$), and let $(\boldsymbol{s}_j)$ be a set of ℓ vectors of Θ bits such that, for all $j = 0, \ldots, \ell - 1$,

$$\frac{2^\eta}{p_j^2} = \langle \boldsymbol{s}_j, \boldsymbol{z} \rangle + \varepsilon_j \mod 2^\eta$$

where $|\varepsilon_j| \leqslant 2^{-\kappa}$. Let $\boldsymbol{s}'_j = \mathsf{PowersofTwo}_\eta(\boldsymbol{s}_j) \in \mathbb{Z}^{\eta\Theta}$. Define $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_{\eta\Theta})$ so that, for all $1 \leqslant i \leqslant \eta\Theta$:

$$\sigma_i = \mathsf{CRT}_{q_0, p_0^2, \ldots, p_{\ell-1}^2} \left(q_i, r_{0,i} + \left\lfloor s'_{0,i} \cdot \frac{p_0}{2^{\eta+1}} \right\rceil, \ldots, r_{\ell-1,i} + \left\lfloor s'_{\ell-1,i} \cdot \frac{p_{\ell-1}}{2^{\eta+1}} \right\rceil \right)$$

is an encryption of $(s'_{j,i})_{1 \leqslant j \leqslant \ell}$. For $\mathsf{Convert}$ we use the same algorithm as in Section 3:

$\mathsf{Convert}(\boldsymbol{z}, \boldsymbol{\sigma}, c)$**.** First compute $\boldsymbol{c} = (\lfloor c \cdot z_i \rceil \bmod 2^\eta)_{1 \leqslant i \leqslant \Theta}$ and then its decomposition $\boldsymbol{c'} = \mathsf{BitDecomp}_\eta(\boldsymbol{c})$. Finally, output

$$c' \leftarrow 2\langle \boldsymbol{\sigma}, \boldsymbol{c'} \rangle \bmod x_0 \, .$$

The proof of the following lemma follows directly from the proof of Lemma 1 applied modulo each of the p_j's.

Lemma 3. *The procedure* Convert *above converts a Type-II ciphertext with noise size ρ' into a Type-I ciphertext with noise $(\rho' - \eta + 5, \log_2 \Theta)$, for $\rho' - \eta \geqslant \rho + \log_2(\eta\Theta)$.*

In the full version of this paper [CLT14] we provide a full description of the resulting batch leveled fully homomorphic scheme. We also show that the batch scheme is semantically secure under a variant of the previous Decisional-Approximate-GCD assumption with error-free x_0.

5 Practical Implementation

In this section, we provide concrete parameters and timings for a homomorphic evaluation of AES with our batch scale-invariant DGHV scheme. For homomorphic AES evaluations we compare our timings with the RLWE-based leveled-FHE scheme in [GHS12b] and with the batch (bootstrapping-based) DGHV scheme in [CCK$^+$13,CLT13]. We use the following existing optimizations:

1. Subset-sum: as in [CMNT11] we use β-bit integers b_i instead of bits in the subset sum, to reduce the value of τ. Namely the condition becomes $\beta \cdot \tau \geqslant \gamma + 2\lambda$.
2. Public-key compression: the technique in [CNT12,CLT13] enables to compress the ciphertexts in the public-key from γ to roughly $\ell \cdot \eta$ bits.
3. Ciphertext expand [CNT12]: the technique consists in generating the z_i's with a special structure instead of pseudo-random. Let δ be a parameter to be specified later. One generates a random z with $\kappa + \delta \cdot \Theta \cdot \eta$ bits of precision after the binary point, and one defines the z_i's for $\ell + 1 \leqslant i \leqslant \Theta$ as
$$z_i = \left[z \cdot 2^{i \cdot \delta \cdot \eta}\right]_{2^\eta},$$
keeping only κ bits of precision after the binary point for each z_i as previously. We fix $z_1, \ldots, z_\ell$ so that the previous equalities hold. Then the ciphertext expansion can be computed as follows, for all $\ell + 1 \leqslant i \leqslant \Theta$:
$$c_i = \lfloor c \cdot z_i \rceil \bmod 2^\eta = \lfloor c \cdot z \cdot 2^{i \cdot \delta \cdot \eta} \rceil \bmod 2^\eta .$$
Therefore computing all the z_i's (except the first ℓ) is now essentially a single multiplication $c \cdot z$. A lattice attack against this optimization is described in [CNT12]; the authors show that the attack is thwarted by selecting δ such that $\delta \cdot \Theta \cdot \eta \geqslant 3\gamma$.

5.1 Optimization of Scalar Product

We describe an additional optimization for computing the scalar product $c' = 2\langle \boldsymbol{\sigma}, \boldsymbol{c'} \rangle$ computed in Convert, similar to the ciphertext expand optimization above. The vectors $\boldsymbol{\sigma}$ and $\boldsymbol{c'}$ have $\eta\Theta$ elements. We first divide the vectors $\boldsymbol{\sigma}$ and $\boldsymbol{c'}$ into subvectors of Θ elements, and we compute the scalar products of the subvectors separately. In the following for simplicity we keep the same notations and now assume that $\boldsymbol{\sigma}$ and $\boldsymbol{c'}$ have Θ elements each.

We generate the vector $\boldsymbol{\sigma} \in \mathbb{Z}^{\Theta}$ such that:

$$\sigma_i = \left\lfloor \sigma \cdot 2^{i \cdot \delta \cdot \eta} \right\rceil + v_i$$

for small public corrections $|v_i| \leqslant 2^{\eta \cdot \ell}$ for all $1 \leqslant i \leqslant \Theta$, where the large public random σ has $\delta\eta\Theta$ bits of precision after the binary point, and $\gamma + \delta\eta\Theta$ bits in total. Then

$$\begin{aligned} c' = 2\langle \boldsymbol{\sigma}, \boldsymbol{c'} \rangle &= 2 \sum_{i=1}^{n} \left\lfloor \sigma \cdot 2^{i \cdot \delta \cdot \eta} \right\rceil \cdot c'_i + 2\langle \boldsymbol{v}, \boldsymbol{c'} \rangle = 2 \sum_{i=1}^{n} \left(\sigma \cdot 2^{i\delta\eta} + u_i \right) c'_i + 2\langle \boldsymbol{v}, \boldsymbol{c'} \rangle \\ &= 2\sigma \cdot \left(\sum_{i=1}^{n} c'_i \cdot 2^{i\delta\eta} \right) + 2\langle \boldsymbol{v}, \boldsymbol{c'} \rangle + u = \left\lfloor 2\sigma \cdot \left(\sum_{i=1}^{n} c'_i \cdot 2^{i\delta\eta} \right) \right\rceil + 2\langle \boldsymbol{v}, \boldsymbol{c'} \rangle + u' \ , \end{aligned}$$

where $|u_i| \leqslant 1/2$, $|u| \leqslant \Theta$, and $u' \in \mathbb{Z}$ is such that $|u'| \leqslant \Theta + 1$. Then the scalar product becomes essentially one multiplication and another scalar product but with much smaller entries v_i's instead of σ_i's.

Therefore with vectors $\boldsymbol{\sigma}$ and $\boldsymbol{c'}$ with $\eta\Theta$ elements each instead of Θ, the scalar product $2\langle \boldsymbol{\sigma}, \boldsymbol{c'} \rangle$ becomes essentially η multiplications and another scalar product but with much smaller entries v_i's instead of σ_i's. Note that the size of c' is now $\gamma + \Theta\delta\eta$ bits instead of γ; therefore one must increase κ by twice the same additive factor (to support multiplications of two such converted ciphertexts).

Finally we use the following straightforward optimization: instead of using BitDecomp and PowersofTwo with bits, we use words of size ω bits instead. This decreases the size of the vector $\boldsymbol{\sigma}$ by a factor ω, at the cost of increasing the resulting noise by roughly ω bits. In particular the scalar product $2\langle \boldsymbol{\sigma}, \boldsymbol{c'} \rangle$ then requires essentially $\lceil \eta/\omega \rceil$ multiplications and another scalar product but with smaller entries v_i's instead of σ_i's. In our code we used $\omega = 64$.

5.2 Concrete Parameters and AES Evaluation

In Table 1 we derive concrete parameters as in [CNT12,CCK$^+$13], taking into account the known attacks on the Approximate-GCD problem (see [DGHV10] and [CMNT11,CNT12,CN12,CH12]).

AES evaluation has become a standard evaluation circuit for fully homomorphic encryption [GHS12b,CCK$^+$13]. The main difference between [GHS12b] and [CCK$^+$13] (apart from the underlying FHE scheme) is that bootstrapping was used in the later while in the former the parameters could be made large enough so that no bootstrapping was required to evaluate the full-fledged AES circuit (thanks to the linear growth of the noise). In our scheme we also chose large enough parameters so that the entire AES evaluation could be performed without bootstrapping.

In practice we have evaluated the AES circuit using the state-wise bitslicing variant described in [CLT13] and we obtained the results in Table 1. In this

Table 1. Benchmarking of a C++ implementation of our scale-invariant batch DGHV scheme with a compressed public key on an Intel Xeon E5-2690 at 2.9 GHz on the state-wise AES implementation, using GMP [Gt13].

Instance	λ	ℓ	ρ	η	$\gamma \times 10^{-6}$	τ, Θ	pk size	KeyGen	Encrypt	Decrypt	Mult	Convert
Toy	42	9	42	971	0.27	135	3.2 MB	0.5s	0.0s	0.0s	0.0s	0.1s
Small	52	35	52	976	1.1	525	45 MB	11s	0.2s	0.0s	0.0s	0.3s
Medium	62	140	62	981	4.2	2100	704 MB	5min	3s	0.2s	0.0s	2.8s
Large	72	569	72	986	15.8	8535	11 GB	2h 50min	45s	3.3s	0.1s	33s
Extra	80	1875	86	993	35.9	28125	100 GB	213h	5min	24s	0.3s	277s

Instance	λ	ℓ = # of enc. in parallel	AddRoundKey	SubBytes	ShiftRows	MixColumns	Total Time	**Time/AES block**
Toy	42	9	0.0s	1.5s	0.0s	0.0s	15.1s	**1.7s**
Small	52	35	0.1s	9.9s	0.0s	0.0s	1min 40s	**2.9s**
Medium	62	140	0.3s	80.5s	0.0s	0.1s	13min 29s	**5.8s**
Large	72	569	2.1s	21min	0.0s	0.6s	3h 35min	**23s**
Extra	80	1875	6.9s	10h 9min	0.1s	1.6s	102h	**195s**

variant, the state is represented as an array of 128 ciphertexts, each ciphertext representing one bit of the state of ℓ different AES blocks encrypted in parallel. In [CCK+13,CLT13], the authors obtained a time per AES block of 12 min 46 s on a 4-core machine at 3.4 GHz whereas we obtained 23 s on a 16-core machine at 2.9 GHz for the same security level (72 bits of security); which is *one order of magnitude faster*. For 80 bits of security, timings are competitive with [GHS12b] (3 min vs. 5 min).

6 Equivalence between the (Error-Free) Decisional and Computational Approximate-GCD Problems

In this section, we show the equivalence between the (error-free) decisional and computational Approximate-GCD problems. As a consequence, it follows directly that the additional noises in the fully homomorphic encryption schemes over the integers [DGHV10,CMNT11,CNT12,CLT13] can be removed (as in [CCK+13, Section 3]), simplifying both the schemes and the security proofs. In the following for simplicity we only consider integers $r \in [0, 2^\rho)$ instead of $(-2^\rho, 2^\rho)$. One can always go from one distribution to another by an appropriate centering. Therefore, for a η-bit integer p and $q_0 \in [0, 2^\gamma/p)$, we consider the following distribution over γ-bit integers:

$$\mathcal{D}_\rho(p, q_0) = \{\textsf{Choose } q \leftarrow [0, q_0), r \leftarrow \mathbb{Z} \cap [0, 2^\rho) : \textsf{Output } y = q \cdot p + r\}.$$

Let us recall the definition of the computational and decisional Error-Free Approximate-GCD problems.

Definition 3 (Error-Free (Computational) Approximate-GCD). *The (ρ, η, γ)-error-free Approximate-GCD problem is: For a random η-bit prime p, given a γ-bit 2^{λ^2}-rough integer $x_0 = q_0 \cdot p$ where q_0 is a random integer in $[0, 2^\gamma/p)$, and polynomially many samples from $\mathcal{D}_\rho(p, q_0)$, output p.*

Algorithm 1 Learn-LSB(z, pk)

```
Input: $z = qp + r \in [0, 2^\gamma)$ with $|r| \leqslant 2^\rho$, and $x_0 = q_0 \cdot p$.
Output: The least significant bit of $q$
  Generate $x_1, \ldots, x_\tau \leftarrow \mathcal{D}_\rho(p, q_0)$
  for $j = 1$ to $\mathsf{poly}(\lambda/\epsilon)$ do
    Choose randomly and uniformly a noise $r_j \leftarrow [0, 2^{\rho'})$, a bit $\delta \leftarrow \{0,1\}$ and a random subset $S_j \subset \{1, \ldots, \tau\}$
    Set $y_j = z + \delta + 2r_j + 2\sum_{i \in S_j} x_i \bmod x_0$
    Call $\mathcal{A}$ to get a prediction of $(r \bmod 2) \oplus \delta$: $a_j \leftarrow \mathcal{A}(y_j)$
    Set $b_j \leftarrow a_j \oplus \mathsf{parity}(z) \oplus \delta$
  end for
  Output the majority vote among the $b_j$'s
```

Definition 4 (Error-Free Decisional Approximate-GCD). *The (ρ, η, γ)-error-free Decisional-Approximate-GCD problem is: For a random η-bit prime p, given a γ-bit 2^{λ^2}-rough integer $x_0 = q_0 \cdot p$ and polynomially many samples from $\mathcal{D}_\rho(p, q_0)$, determine $b \in \{0,1\}$ from $z = x + r \cdot b \mod x_0$ where $x \leftarrow \mathcal{D}_\rho(p, q_0)$ and $r \leftarrow \mathbb{Z} \cap [0, x_0)$.*

We also consider the following decisional problem.

Definition 5 (Error-Free LSB Approximate-GCD Problem). *The (ρ, η, γ)-error-free LSB Approximate-GCD problem is: For a random η-bit prime p, given a γ-bit 2^{λ^2}-rough integer $x_0 = q_0 \cdot p$ and polynomially many samples from $\mathcal{D}_\rho(p, q_0)$, determine $b \in \{0,1\}$ from $z = q \cdot p + 2r + b \cdot c$ where $q \leftarrow [0, q_0)$, $r \leftarrow \mathbb{Z} \cap [0, 2^{\rho-1})$ and $c \leftarrow \{0,1\}$.*

One can show that the problems from Definitions 3 and 5 are equivalent. Indeed, we can construct a high-accuracy LSB predictor subroutine (cf. Algorithm 1 below) using an adversary $\mathcal{A}$ having a non-negligible advantage ϵ against the (ρ', η, γ)-Error-Free LSB Approximate-GCD problem (with $\rho' > \log_2(\tau+1) + \rho + \lambda$)[6], and by using it in Step 2 of the security proof of [DGHV10], we automatically get the equivalence.

Let us show that Definitions 4 and 5 are equivalent. We consider the sequence of distributions for $\rho \leqslant i \leqslant \eta + \lambda$:

$$\mathcal{D}'_\rho(p, q_0, i) = \left\{ \begin{array}{l} \mathsf{Choose}\ q \leftarrow [0, q_0), r \leftarrow \mathbb{Z} \cap [0, 2^i) : \\ \mathsf{Output}\ y = q \cdot p + 2^{\lambda+\eta-i} \cdot r \bmod x_0 \end{array} \right\} .$$

Note that in the distribution $\mathcal{D}'_\rho(p, q_0, i)$ above the size of the random r is i-bit instead of ρ-bit. For $i = \rho$, the distribution of y is the same as the distribution $\mathcal{D}_\rho(p, q_0)$, up to a factor $2^{\lambda+\eta-\rho}$ modulo x_0. One can show that for $i = \eta + \lambda$, the distribution $\mathcal{D}'_\rho(p, q_0, i)$ is $2^{-\lambda}$-statistically close to uniform modulo x_0. Therefore by a standard hybrid argument, if a distinguisher solves the Error-Free

[6] The additional noise is use to drawn the noise due to the public key elements and z.

Decisional-Approximate-GCD problem with some non-negligible advantage, then he must be able to distinguish between two successive distributions $\mathcal{D}'_\rho(p, q_0, i)$ and $\mathcal{D}'_\rho(p, q_0, i+1)$ for some i.

Let us consider the challenge from the Error-Free LSB Approximate-GCD problem:

$$z = q \cdot p + 2r + b \cdot c$$

where $r \leftarrow \mathbb{Z} \cap [0, 2^{\rho-1})$ and $c \leftarrow \{0, 1\}$. We let:

$$y = 2^{\lambda+\eta-i-1} \cdot (2^\rho \cdot u + z) \bmod x_0$$

where $u \leftarrow \mathbb{Z} \cap [0, 2^{i+1-\rho})$. This gives:

$$\begin{aligned} y &= q' \cdot p + 2^{\lambda+\eta-i-1} \cdot (2^\rho \cdot u + 2r + b \cdot c) \bmod x_0 \\ &= q' \cdot p + 2^{\lambda+\eta-i-1} \cdot (2r' + b \cdot c) \end{aligned}$$

for some $q' \in \mathbb{Z}$, where $r' \leftarrow \mathbb{Z} \cap [0, 2^i)$.

If $b = 0$ then we get $y = q' \cdot p + 2^{\lambda+\eta-i} \cdot r'$ which corresponds to the distribution $\mathcal{D}'_\rho(p, q_0, i)$. If $b = 1$ then we get $y = q' \cdot p + 2^{\lambda+\eta-i-1} \cdot r''$ where $r'' \leftarrow \mathbb{Z} \cap [0, 2^{i+1})$, which corresponds to the distribution $\mathcal{D}'_\rho(p, q_0, i+1)$. Therefore we can use the previous distinguisher to solve the Error-Free LSB Approximate-GCD problem.

References

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, 2012.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886. Springer, 2012.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS 2011*, pages 97–106. IEEE Computer Society, 2011.

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

[CCK+13] Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 315–335. Springer, 2013.

[CH12] Henry Cohn and Nadia Heninger. Approximate common divisors via lattices. In *ANTS X*, 2012.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Batch fully homomorphic encryption over the integers. Cryptology ePrint Archive, Report 2013/036, 2013. `http://eprint.iacr.org/`.

[CLT14] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Scale-invariant fully homomorphic encryption over the integers. Full version of this paper. Cryptology ePrint Archive, Report 2014/032, 2014. `http://eprint.iacr.org/`.

[CMNT11] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 487–504. Springer, 2011.

[CN12] Yuanmi Chen and Phong Nguyen. Faster algorithms for approximate common divisors: Breaking fully-homomorphic-encryption challenges over the integers. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 502–519. Springer, 2012.

[CNT12] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 446–464. Springer, 2012.

[DGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010.

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

[GH11] Craig Gentry and Shai Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In Kenneth Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 129–148. Springer, 2011.

[GHS12a] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 465–482. Springer, 2012.

[GHS12b] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867. Springer, 2012.

[Gt13] Torbjörn Granlund and the GMP development team. *GNU MP: The GNU Multiple Precision Arithmetic Library*, 5.1.3 edition, 2013. `http://gmplib.org/`.

[HILL99] Johan Håstad, Russel Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28:12–24, 1999.

[KLYC13] Jinsu Kim, Moon Sung Lee, Aaram Yun, and Jung Hee Cheon. CRT-based fully homomorphic encryption over the integers. Cryptology ePrint Archive, Report 2013/057, 2013. `http://eprint.iacr.org/`.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *STOC 2005*, pages 84–93. ACM, 2005.

[SV10] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Phong Nguyen and David

Pointcheval, editors, *Public Key Cryptography – PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443. Springer, 2010.

[SV11] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *IACR Cryptology ePrint Archive*, 2011:133, 2011.

A Leftover Hash Lemma

We recall the classical Leftover Hash Lemma (LHL), following [DGHV10]. A family $\mathcal{H}$ of hash functions from X to Y, both finite sets, is said to be pairwise-independent if for all distinct $x, x' \in X$, $\Pr_{h \leftarrow H}[h(x) = h(x')] = 1/|Y|$. A distribution D is ε-uniform if its statistical distance from the uniform distribution is at most ε, where the statistical distance $\Delta(D_1, D_2)$ between two distributions D_1, D_2 over a finite domain X is given by $\Delta(D_1, D_2) = \frac{1}{2}\sum_{x \in X} |D_1(x) - D_2(x)|$.

Lemma 4 (Leftover Hash Lemma [HILL99]). *Let $\mathcal{H}$ be a family of pairwise hash functions from X to Y. Suppose that $h \leftarrow \mathcal{H}$ and $x \leftarrow X$ are chosen uniformly and independently. Then, $(h, h(x))$ is $\frac{1}{2}\sqrt{|Y|/|X|}$-uniform over $\mathcal{H} \times X$.*

From the LHL one can deduce the following Lemma for finite sums modulo an integer M, as proved in [DGHV10]:

Lemma 5. *Set $x_1, \ldots, x_m \leftarrow \mathbb{Z}_M$ uniformly and independently, set $s_1, \ldots, s_m \leftarrow \{0,1\}$, and set $y = \sum_{i=1}^{m} s_i \cdot x_i \bmod M$. Then $(x_1, \ldots, x_m, y)$ is $1/2\sqrt{M/2^m}$-uniform over $\mathbb{Z}_M^{m+1}$.*

Proof. We consider the following hash function family $\mathcal{H}$ from $\{0,1\}^m$ to $\mathbb{Z}_M$. Each member $h \in \mathcal{H}$ is parameterized by the elements $(x_1, \ldots, x_m) \in \mathbb{Z}_M^m$. Given $\boldsymbol{s} \in \{0,1\}^m$, we define $h(\boldsymbol{s}) = \sum_{i=1}^{m} s_i \cdot x_i \in \mathbb{Z}_M$. The hash function family is clearly pairwise independent. Therefore by Lemma 4, $(h, h(x))$ is $1/2\sqrt{M/2^m}$-uniform over $\mathbb{Z}_M^{m+1}$. □