Some security bounds for the DGHV scheme

Franca Marinelli (f.marinelli@studenti.unitn.it)
Department of Mathematics, University of Trento, Italy

Riccardo Aragona (riccardo.aragona@unitn.it)
Department of Mathematics, University of Trento, Italy

Chiara Marcolla (chiara.marcolla@unitn.it)
Department of Mathematics, University of Trento, Italy

Massimiliano Sala (maxsalacodes@gmail.com)
Department of Mathematics, University of Trento, Italy

Abstract

The correctness in decrypting a ciphertext after some operations in the DGVH scheme depends heavily on the dimension of the secret key. In this paper we compute two bounds on the size of the secret key for the DGHV scheme to decrypt correctly a ciphertext after a fixed number of additions and a fixed number of multiplication. Moreover we improve the original bound on the dimension of the secret key for a general circuit.

Keywords: Public-key cryptography, Fully Homomorphic Encryption, Somewhat Homomorphic Encryption, DGVH scheme.

1 Introduction

Fully Homomorphic Encryption (FHE) allows to perform computation of arbitrary functions on encrypted data without being able to decrypt. The first construction of an FHE scheme was described by Gentry in his PhD thesis [Gen09] in 2009. The first Gentry's FHE scheme [Gen09,Gen10] proceeds in three steps. First, one constructs a *Somewhat Homomorphic Encryption (SHE)* scheme, namely which is able to decrypt correctly only a limited number of operations. The second step is to express the encryption function by a low-degree polynomial (*squash*). Then, one applies the *bootstrapping* to reduce the noise and to compact the ciphertext.

Nowadays three main families of FHE schemes are:

1. Gentry's scheme [Gen09,Gen10], based on hard problems on ideal lattices, and implemented in [SV10,GH11].
2. van Dijk, Gentry, Halevi and Vaikuntanathan's (DGHV) scheme over the integers [DGHV10], based on the approximate GCD problem, and implemented in [CMNT11]. A *batch* version of this scheme has been proposed in [CCK$^+$13].
3. Brakerski and Vaikuntanathan's (BV) scheme based on the Learning with Errors (LWE) problem [BV11a] or Ring Learning with Errors (RLWE) problem [BV11b]. Other implementations are in [NLV11,BGV12,GHS12].

For a survey articles see [Sil13,Vai11].

In this paper we focus on the SHE of the DGHV scheme [DGHV10], whose description we recall in Section 2. As already said, in SHE schemes if the size of noise remains below a certain threshold, then one can decrypt correctly the ciphertext. This threshold is dependent on the dimension of the secret key. In Section 3, first we compute two bounds on the size of the secret key which permit respectively to decrypt correctly a ciphertext after performing a certain number of homomorphic either additions or multiplications (Lemma 3.2). Then, in Lemma 3.3 we improve slightly the bound on the secret key for a general circuit, claimed in Lemma 3 in [DGHV10]. Finally, in Subsection 3.3 we show explicitly that in Evaluate algorithm we cannot reduce the ciphertext modulo the public key element x_0 as observed in [DGHV10].

2 Preliminary

2.1 Homomorphic Encryption

Let λ be the security parameter. An *homomorphic public key encryption scheme* $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Evaluate})$ is a quadruple of probabilistic polynomial-time (PPT) algorithms as follows.

- $\mathsf{KeyGen}(\lambda) = (\mathsf{sk}, \mathsf{pk})$,
  it takes as input a security parameter λ and outputs a pair of keys $(\mathsf{sk}, \mathsf{pk})$, where sk is called the secret key and pk is called the public key.
- $\mathsf{Encrypt}(\mathsf{pk}, m) = c$, where $m \in \mathbb{F}_2$,
  it takes as inputs pk and a message m, that is a bit, and outputs a ciphertext c.
- $\mathsf{Decrypt}(\mathsf{sk}, c) = m^*$,
  it takes as input sk and a ciphertext c and outputs a bit m^*.
- $\mathsf{Evaluate}(\mathsf{pk}, C, (c_1, \ldots, c_t)) = c_f$,

it takes as input a public key pk, a circuit C and a t-upla of ciphertexts $\mathbf{c} = (c_1, \dots, c_t)$, where $c_i = \mathsf{Encrypt}(\mathsf{pk}, m_i)$ and outputs a ciphertext c_f.

A fully homomorphic scheme is a scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Evaluate})$ having the following property: if $\mathsf{Decrypt}(\mathsf{sk}, c_i) = m_i$, then

$$\mathsf{Decrypt}(\mathsf{sk}, c_1 + c_2) = m_1 + m_2 \qquad \text{and} \qquad \mathsf{Decrypt}(\mathsf{sk}, c_1 \cdot c_2) = m_1 \cdot m_2$$

More generally, whenever we have finitely many additions and multiplications, that is, a t-input circuit C, then

$$\mathsf{Decrypt}(\mathsf{sk}, \mathsf{Evaluate}(\mathsf{pk}, C, c_1, \dots, c_t)) = C(m_1, \dots, m_t)$$

Another requirement of FHE, called *ciphertext compactness*, is that the size of ciphertexts is bounded and independent of the circuit C. To define formally FHE, we need the following definitions given in [DGHV10].

Definition 2.1. *The scheme $\mathcal{E}$ is **correct** for a given t-input circuit C if, for any key-pair $(\mathsf{sk}, \mathsf{pk})$ output by $\mathsf{KeyGen}(\lambda)$, any t plaintexts $m_1, \dots, m_t \in \mathbb{F}_2$, and any t ciphertexts $c_i = \mathsf{Encrypt}(\mathsf{pk}, m_i)$ for $i = 1, \dots, t$, it is the case that:*

$$\mathsf{Decrypt}(\mathsf{sk}, \mathsf{Evaluate}(\mathsf{pk}, C, (c_1, \dots, c_t)) = C(m_1, \dots, m_t).$$

Definition 2.2. *The scheme $\mathcal{E}$ is **homomorphic** for a class $\mathcal{C}$ of circuits if it is correct for all circuits $\mathcal{C} \in C$, whereas $\mathcal{E}$ is **fully homomorphic** if it is correct for all boolean circuits.*

In this paper, we deal with Somewhat Homomorphic Encryption. Informally, we say that $\mathcal{E}$ is a SHE scheme if it has only some homomorphic properties but it is not fully since

- it can perform a *limited number of operations*,
- the *ciphertexts compactness* requirement might be violated.

In the next subsection we see in details the DGHV scheme defined in [DGHV10].

2.2 The Somewhat Homomorphic DGHV Scheme

Throughout the paper we denote a random choice of an element x in a set X by $x \xleftarrow{\$} X$. Let λ be the security parameter. The DGHV public-key scheme is the homomorphic encryption scheme

$$\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Evaluate}),$$

which uses the five parameters (all depending from λ):

η : the bit-length of the secret key sk,

γ : the bit-length of the integers in the public key pk,

ρ : the bit-length of the noise in KeyGen,

ρ' : the bit-length of the noise in $\mathsf{Encrypt}$,

τ : the number of integers in the public key.

In [DGHV10] the authors proved that these parameters must be set as follows:

- $\rho = \omega(\log \lambda)$, to protect against brute-force attacks on the noise;
- $\eta \geq \rho \cdot \Theta(\lambda \log^2 \lambda)$ to support homomorphism for deep enough circuits to evaluate the "squashed decryption circuit" (see Sections 3.2 and 6.2 in [DGHV10]);
- $\gamma = \omega(\eta^2 \log \lambda)$ to thwart various lattice-based attacks on the underlying approximate-gcd problem (see Section 5 in [DGHV10]);
- $\tau \geq \gamma + \omega(\log \lambda)$ (see Lemma 4.3 in [DGHV10]);
- $\rho' = \rho + \omega(\log \lambda)$, used as secondary noise parameter.

Hence, a convenient set of parameters is $\rho = \lambda$, $\rho' = 2\lambda, \eta = O(\lambda^2), \gamma = O(\lambda^5)$ and $\tau = \gamma + \lambda$.
For a specific odd η-bit positive integer p, we use the following distribution over γ-bit integers:

$$\mathcal{D}_{\gamma,\rho}(p) = \{x = pq + r : q \xleftarrow{\$} \mathbb{Z} \cap [0, 2^\gamma/p), r \xleftarrow{\$} \mathbb{Z} \cap (-2^\rho, 2^\rho)\}.$$

The algorithms of $\mathcal{E}$ are defined as follows:

$\mathsf{KeyGen}(\lambda) = (\mathsf{pk}, \mathsf{sk})$.
The secret key sk is a random odd η-bit positive integer

$$\mathsf{sk} := p \in (2\mathbb{Z} + 1) \cap [2^{\eta-1}, 2^\eta).$$

Instead, for the public key, KeyGen works as follows:

(a) for each $i = 0, ..., \tau$, chooses randomly $x_i \in \mathcal{D}_{\gamma,\rho}(p)$,
(b) relabels x_i so that $x_0 = pq_0 + r_0$ is the largest,
(c) if $x_0 \mod 2 \equiv 1$ and $r_0 \mod 2 \equiv 0$, i.e. x_0 is odd and r_0 is even, then the public key is the set of numbers chosen in (a):

$$\mathsf{pk} = \{x_0, x_1, ..., x_\tau\}.$$

Otherwise it restarts from (a).

$\mathsf{Encrypt}(\mathsf{pk}, m) = c$.
It chooses:

- a random subset $S \subseteq \{1, ..., \tau\}$,
- a random r' in $\mathbb{Z} \cap (-2^{\rho'}, 2^{\rho'})$

and computes a ciphertext

$$c = (m + 2r' + 2\sum_{i\in S} x_i) \mod x_0.$$

$\mathsf{Decrypt}(\mathsf{sk}, c) = m^*$.
The output m^* is computes in this way:

$$m^* = (c \mod p) \mod 2.$$

$\mathsf{Evaluate}(\mathsf{pk}, C, c_1, ..., c_t) = c_f$.
Given the (binary) circuit C with t inputs, and t ciphertexts c_i, we apply the (integer) addition and multiplication gates of C to the ciphertexts, performing all the operations over the integers, and return the resulting integer. Finally the output of $\mathsf{Evaluate}$ is a ciphertext c_f, that is an integer.

3 Bound on the decryption

For every $x \in \mathbb{R}$, let $\lfloor x \rfloor = \max\{k \in \mathbb{Z} : k \leq x\}$ be the floor of x and let $\lceil x \rceil = \min\{k \in \mathbb{Z} : k \geq x\}$ be the ceiling of x. We denote the nearest integer of x by $\lfloor x \rceil$, in other words

$$\lfloor x \rceil = \begin{cases} \lfloor x \rfloor & \text{if } x - \lfloor x \rfloor < \frac{1}{2} \\ \lceil x \rceil & \text{if } x - \lfloor x \rfloor \geq \frac{1}{2}. \end{cases}$$

Throughout the section for every y and $m \in \mathbb{Z}$ we consider the reduction of y modulo m in $(-\frac{m}{2}, \frac{m}{2}]$.

3.1 Bound on the decryption of fresh ciphertexts

Lemma 3.1. *Let* $(\mathsf{pk}, \mathsf{sk})$ *be the output by* $\mathsf{KeyGen}(\lambda)$ *and let* c *be the output by* $\mathsf{Encrypt}(\mathsf{pk}, m)$*. If* $\eta > \log_2(2^{\rho'} + \tau 2^{\rho+1}) + 2$*, then* $\mathsf{Decrypt}(\mathsf{sk}, c) = m$*, that is, it is able to decrypt correctly* c*.*

Proof. By definition $c = (m + 2r' + 2\sum_{i\in S} x_i) \mod x_0$ is a fresh ciphertext. In particular we have that:

$$\begin{aligned} c &= \Big(m + 2r' + 2\sum_{i\in S} x_i\Big) \mod x_0 \\ &= m + 2r' + 2\sum_{i\in S} x_i - kx_0 \text{ (for some } k \in \mathbb{Z}) \\ &= m + 2r' + 2\sum_{i\in S}(pq_i + r_i) - k(pq_0 + r_0) \end{aligned}$$

$$c = p\Big(2\sum_{i\in S} q_i - kq_0\Big) + m + 2r' + 2\sum_{i\in S} r_i - kr_0.$$

When we decrypt the fresh ciphertext c, we compute:

$$(c \mod p) \mod 2 = \Big(\big(m + 2r' + 2\sum_{i\in S} r_i - kr_0\big) \mod p\Big) \mod 2.$$

Since r_0 is even, if $m + 2r' + 2\sum_{i\in S} r_i - kr_0$ is in $[-\frac{p}{2}, \frac{p}{2})$ then

$$\Big(\big(m+2r'+2\sum_{i\in S} r_i-kr_0\big) \mod p\Big) \mod 2 = \Big(m+2r'+2\sum_{i\in S} r_i-kr_0\Big) \mod 2 = m.$$

So we want that

$$-\frac{p}{2} \leq m + 2r' + 2\sum_{i\in S} r_i - kr_0 < \frac{p}{2}. \quad (1)$$

Now $m+2r'+2\sum_{i\in S} x_i \equiv c \mod x_0$, i.e. $m+2r'+2\sum_{i\in S} x_i = c+kx_0$. Since we consider the reduction modulo x_0 in $[-\frac{x_0}{2}, \frac{x_0}{2})$, then

$$k = \frac{m + 2r' + 2\sum_{i\in S} x_i}{x_0} - \frac{c}{x_0} = \left\lfloor \frac{m + 2r' + 2\sum_{i\in S} x_i}{x_0} \right\rceil. \quad (2)$$

Since $m + 2r' \ll x_0$, then we can consider $\frac{m+2r'}{x_0} < \frac{1}{2}$. In the worst case of (2), we have that $\sum_{i\in S} x_i = \tau x_0$, so we obtain

$$k = \left\lfloor \frac{m + 2r' + 2\sum_{i\in S} x_i}{x_0} \right\rceil = \left\lfloor \frac{m + 2r'}{x_0} + 2\tau \right\rceil = 2\tau$$

Whereas, in the worst case of (1), replacing k with 2τ, we have

$$\frac{p}{2} > m + 2r' + 2\sum_{i\in S} r_i - 2\tau r_0.$$

Considering the number of bits of p, r_i and r', since $r_0 \in (-2^\rho, 2^\rho)$, for the worst case we obtain

$$2^{\eta-1} > 2 \cdot 2^{\rho'} + 2\tau \cdot 2^\rho + 2\tau \cdot 2^\rho = 2 \cdot 2^{\rho'} + 4\tau \cdot 2^\rho = 2(2^{\rho'} + \tau 2^{\rho+1}),$$

that is, $2^\eta > 4(2^{\rho'} + \tau 2^{\rho+1})$. Then we obtain the bound

$$\eta > \log_2(2^{\rho'} + \tau 2^{\rho+1}) + 2. \quad (3)$$

□

3.2 Bound on the decryption of ciphertexts after operations

In this section we present two different bounds on the bit-length of sk such that DGHV-scheme is homomorphic with respect to either a circuit with only v addictions or a circuit with only s multiplications.

Lemma 3.2. *Let* (pk, sk) *be the output of* $\mathsf{KeyGen}(\lambda)$ *and let* c_a *be the output of* $\mathsf{Evaluate}(\mathsf{pk}, C, c_1, ..., c_v)$ *in the addition case, where for* $i = 1, \dots, v$, $\mathsf{Encrypt}(\mathsf{pk}, m_i) = c_i$. *Let* c_m *be the output of* $\mathsf{Evaluate}(\mathsf{pk}, C, c_1, ..., c_s)$ *in the multiplication case, where* $\mathsf{Encrypt}(\mathsf{pk}, m_i) = c_i$, *for* $i = 1, \dots, s$ *.*

1. *If* $\eta > \log_2(2^{\rho'} + \tau 2^{\rho+1}) + \log_2 v + 2$, *then* $\mathsf{Decrypt}(\mathsf{sk}, c_a)$ *is able to decrypt correctly* c_a, *that is,* $\mathsf{Decrypt}(\mathsf{sk}, c_a) = C(m_1, \dots, m_v) = m_1 + \dots + m_v$.
2. *If* $\eta > s\left[\log_2(2^{\rho'} + \tau 2^{\rho+1}) + 1\right] + 1$, *then* $\mathsf{Decrypt}(\mathsf{sk}, c_m)$ *is able to decrypt correctly* c_m, *that is,* $\mathsf{Decrypt}(\mathsf{sk}, c_m) = C(m_1, \dots, m_s) = m_1 \cdot \ldots \cdot m_s$.

Proof. We consider two distinct cases: the sum of v ciphertexts and the product of s ciphertexts. For both we examine the worst case, that is, the sum (or the product) of ciphertext having the same error.

1. We suppose that $\mathsf{Evaluate}$ takes as input v times the same ciphertext c. So, we have

$$\underbrace{c + ... + c}_{v\ times} = vc = v\left(m + 2r' + 2\textstyle\sum_{i\in S} x_i - kx_0\right),$$

where k is such as in (2).

As in the proof of Lemma 3.1, we want that

$$-\frac{p}{2} \leq v(m + 2r' + 2\sum_{i\in S} r_i - kr_0) < \frac{p}{2}.$$

So we obtain $2^\eta > 4v(2^{\rho'} + \tau 2^{\rho+1})$, that is,

$$\eta > \log_2(2^{\rho'} + \tau 2^{\rho+1}) + \log_2 v + 2 = \log_2(v(2^{\rho'} + \tau 2^{\rho+1})) + 2.$$

Note that the inequality below is $\eta > B + \log_2 v$, where B is the value on the right side of bound (3).

2. We suppose that $\mathsf{Evaluate}$ takes as input s times the same ciphertext c.

$$\underbrace{c \cdot ... \cdot c}_{s\ times} = c^s = \left(m + 2r' + 2\textstyle\sum_{i\in S} x_i - kx_0\right)^s.$$

where k is such as in (2).

As before, we want that $-\frac{p}{2} \leq \left(m + 2r' + 2\sum_{i\in S} x_i - kr_0\right)^s < \frac{p}{2}$. Thus, we obtain

$$2^\eta > 2^{s+1}(2^{\rho'} + \tau 2^{\rho+1})^s,$$

that is,

$$\eta > s\log_2(2^{\rho'} + \tau 2^{\rho+1}) + s + 1 = s\left[\log_2(2^{\rho'} + \tau 2^{\rho+1}) + 1\right] + 1. \quad (4)$$

Note that the inequality below is $\eta > s \cdot (B-1)+1$, where B is the value on the right side of bound (3).

□

In general we have the following lemma:

Lemma 3.3. *Let C be a binary circuit with t inputs, and let C' be the associated integer circuit (where the gates are replaced with integer operations). Let $f(x_1,\ldots,x_t)$ be the multivariate polynomial computed by C' and let d be its degree. If*

$$\eta \geq d\left[\log_2(2^{\rho'} + \tau 2^{\rho+1}) + 1\right] + 1 + \log|\mathbf{f}|,$$

where $|\mathbf{f}|$ is the sum of absolute values of the coefficients of f, then $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{Evaluate}(\mathsf{pk}, C, c_1, ..., c_t)) = C(m_1,\ldots,m_t)$.

Proof. Suppose that Evaluate takes as input s times the same ciphertext c, that is, $f(c) = a_0 + a_1c + \ldots + a_dc^d$. As before, we want that $|f(c)| \leq p/2$. Since

$$|a_0 + a_1c + \ldots + a_dc^d| \leq |a_0 + a_1 + \ldots + a_d| \cdot |c^d| = |\mathbf{f}| \cdot |c^d|,$$

so, if the scheme is correct for $|f(c)| < p/2$, then it is correct also for $|\mathbf{f}||c^d| < p/2$. By 2. of Lemma 3.2, we obtain:

$$\eta \geq d\log_2(2^{\rho'} + \tau 2^{\rho+1}) + d + 1 + \log|\mathbf{f}|.$$

□

Note that for large λ, we have $\log_2(2^{\rho'} + \tau 2^{\rho+1}) \approx \log_2(2^{\rho'})$ and so we obtain

$$\eta \geq d(\rho'+1) + 1 + \log|\mathbf{f}|.$$

If we consider $|f(c)| < p/8$, we obtain $\eta \geq d(\rho'+1) + 4 + \log|\mathbf{f}|$, slightly improving of Lemma 3 in [DGHV10] which claims $\eta \geq d(\rho'+2) + 4 + \log|\mathbf{f}|$. In particular, if d is large then the improvement is significant.

3.3 Encrypt *vs* Evaluate

As we saw before, a key property of FHE is the compactness of the ciphertext [Gen09,Gen10,Vai11,DGHV10]. We recall that a ciphertext c_f is compact if its size, after homomorphic evaluation, does not depend on the number of inputs t and it is also independent of the circuit C. This means that we want that the ciphertext c_f has the same size of the output c of Encrypt. Note that it does not happen in DGHV scheme because in the Encrypt algorithm the ciphertext is reduced modulo x_0, whereas in the Evaluate algorithm this reduction is not performed and so the ciphertext grows. After just one multiplication the ciphertext becomes much larger than x_0 and this implies that $\eta > \gamma$. We recall that γ has to be equal to $\omega(\eta^2 \log \lambda)$, and so bigger than η.

Although the reduction modulo x_0 would help in the Evaluate algorithm, in [DGHV10] the authors claim that this reduction is not possible. We now prove their claim. We consider c_f equals to c^2 modulo x_0:

$$
\begin{aligned}
c_f = c^2 \mod x_0 &= \Big(m + 2r' + 2\sum_{i\in S} x_i - kx_0\Big)^2 \mod x_0 \\
&= \Big(m + 2r' + 2\sum_{i\in S} x_i\Big)^2 \mod x_0 \\
&= \Big(m + 2r' + 2\sum_{i\in S} x_i\Big)^2 - k'x_0, \text{ for some } k' \in \mathbb{Z}. \qquad (5)
\end{aligned}
$$

So we have

$$
k' = \left\lfloor \frac{\big(m + 2r' + 2\sum_{i\in S} x_i\big)^2}{x_0} - \frac{c^2}{x_0} \right\rceil = \left\lfloor \frac{\big(m + 2r' + 2\sum_{i\in S} x_i\big)^2}{x_0} \right\rceil,
$$

that is, in the worst case,

$$
k' = \left\lfloor \frac{(m + 2r' + 2\tau x_0)^2}{x_0} \right\rceil = \left\lfloor \frac{(m + 2r')^2}{x_0} + \frac{4\tau^2 x_0^2}{x_0} + \frac{4\tau x_0(m + 2r')}{x_0} \right\rceil.
$$

Hence, since we can consider $\frac{m+2r'}{x_0} < \frac{1}{2}$, we obtain

$$
k' = 4\tau^2 x_0 + 4\tau(m + 2r'). \qquad (6)
$$

By (5)

$$
\begin{aligned}
c_f \mod p &= \Big(\big(m + 2r' + 2p\sum_{i\in S} q_i + 2\sum_{i\in S} r_i\big)^2 - k'pq_0 - k'r_0\Big) \mod p \\
&= \Big(\big(m + 2r' + 2\sum_{i\in S} r_i\big)^2 - k'r_0\Big) \mod p
\end{aligned}
$$

Therefore, as done in the proof of Lemma 3.1 and replacing k' with $4\tau^2 x_0 + 4\tau(m + 2r')$, to decrypt correctly c_f we want that

$$
-\frac{p}{2} \le \Big(m + 2r' + 2\sum_{i\in S} r_i\Big)^2 - (4\tau^2 x_0 + 4\tau(m + 2r'))r_0 < \frac{p}{2} \qquad (7)
$$

Considering the number of bits of p, r_i and r', since (7) holds f or every $r_0 \in (-2^\rho, 2^\rho)$, in the worst case we can observe that

$$
\begin{aligned}
2^{\eta-1} &> \left(2 \cdot 2^{\rho'} + 2\tau \cdot 2^{\rho}\right)^2 + 2^{\rho}\left(2^2\tau^2 \cdot 2^{\gamma} + 2^3\tau \cdot 2^{\rho'}\right) \\
&> 4\left(\tau^2 \cdot 2^{\rho}(2^{\gamma} + 2^{\rho}) + 4\tau \cdot 2^{\rho+\rho'} + 2^{2\rho'}\right).
\end{aligned}
$$

Hence

$$
\begin{aligned}
\eta &> 3 + \log_2\left(\tau^2 \cdot 2^{\rho}(2^{\gamma} + 2^{\rho}) + 4\tau \cdot 2^{\rho+\rho'} + 2^{2\rho'}\right) \\
&> 3 + \log_2\left(\tau^2 \cdot 2^{\rho+\gamma}\right) = 3 + 2\log_2 \tau + \rho + \gamma > \gamma.
\end{aligned}
$$

Acknowledgements

This research has been supported by TELSY S.p.A.

These results are contained in the first author's MSc thesis and she would like to thank the other authors, especially her supervisor (the last author).

References

[BGV12] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, *(Leveled) fully homomorphic encryption without bootstrapping*, Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ACM, 2012, pp. 309–325.

[BV11a] Z. Brakerski and V. Vaikuntanathan, *Efficient fully homomorphic encryption from (standard) LWE*, 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science—FOCS 2011, IEEE Computer Soc., Los Alamitos, CA, 2011, pp. 97–106.

[BV11b] Z. Brakerski and V. Vaikuntanathan, *Fully homomorphic encryption from ring-LWE and security for key dependent messages*, Advances in cryptology—CRYPTO 2011, Lecture Notes in Comput. Sci., vol. 6841, Springer, Heidelberg, 2011, pp. 505–524.

[CCK+13] J. H. Cheon, J.-S. Coron, J. Kim, M. S. Lee, T. Lepoint, M. Tibouchi, and A. Yun, *Batch fully homomorphic encryption over the integers*, Advances in Cryptology–EUROCRYPT 2013, Springer, 2013, pp. 315–335.

[CMNT11] J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi, *Fully homomorphic encryption over the integers with shorter public keys*, Advances in cryptology—CRYPTO 2011, Lecture Notes in Comput. Sci., vol. 6841, Springer, Heidelberg, 2011, pp. 487–504.

[DGHV10] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, *Fully homomorphic encryption over the integers*, Advances in cryptology—EUROCRYPT 2010, Lecture Notes in Comput. Sci., vol. 6110, Springer, Berlin, 2010, pp. 24–43.

[Gen09] C. Gentry, *A fully homomorphic encryption scheme*, Ph.D. thesis, Stanford University, 2009.

[Gen10] ———, *Computing arbitrary functions of encrypted data*, Commun. ACM **53** (2010), no. 3, 97–105.

[GH11] C. Gentry and S. Halevi, *Implementing Gentry's fully-homomorphic encryption scheme*, Advances in cryptology—EUROCRYPT 2011, Lecture Notes in Comput. Sci., vol. 6632, Springer, Heidelberg, 2011, pp. 129–148.

[GHS12] C. Gentry, S. Halevi, and N. P. Smart, *Fully homomorphic encryption with polylog overhead*, Advances in cryptology—EUROCRYPT 2012, Lecture Notes in Comput. Sci., vol. 7237, Springer, Heidelberg, 2012, pp. 465–482.

[NLV11] M. Naehrig, K. Lauter, and V. Vaikuntanathan, *Can homomorphic encryption be practical?*, Proceedings of the 3rd ACM workshop on Cloud computing security workshop, ACM, 2011, pp. 113–124.

[Sil13] A. Silverberg, *Fully Homomorphic Encryption for Mathematicians.*, IACR Cryptology ePrint Archive **2013** (2013), 250.

[SV10] N. P. Smart and F. Vercauteren, *Fully homomorphic encryption with relatively small key and ciphertext sizes*, Public key cryptography—PKC 2010, Lecture Notes in Comput. Sci., vol. 6056, Springer, Berlin, 2010, pp. 420–443.

[Vai11] V. Vaikuntanathan, *Computing blindfolded: new developments in fully homomorphic encryption*, 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science—FOCS 2011, IEEE Computer Soc., Los Alamitos, CA, 2011, pp. 5–16.