Downlink (T)FHE ciphertexts compression*

Antonina Bondarchuk[1], Olive Chakraborty[1], Geoffroy Couteau[2], and Renaud Sirdey[1]

[1] Université Paris-Saclay, CEA, List, Palaiseau, France,
name.surname@cea.fr
[2] Université Paris Cité, CNRS, IRIF, Paris, France
couteau@irif.fr

Abstract. This paper focuses on the issue of reducing the bandwidth requirement for FHE ciphertext transmission. While this issue has been extensively studied from the *uplink* viewpoint (transmission of encrypted *inputs* towards a FHE calculation), where several approaches exist to essentially cancel FHE ciphertext expansion, the *downlink* case (transmission of encrypted *results* towards an end-user) has been the object of much less attention. In this paper, we address this latter issue with a particular focus on the TFHE scheme, for which we revisit a number of folklore methods, including several approaches for switching to more compact linearly homomorphic schemes, reducing the precision of T(R)LWE coefficients (while maintaining acceptable probabilities of decryption errors), and others. We also investigate how to use these methods in combination, depending on the number of encrypted results to transmit. We further perform extensive experiments demonstrating that the downlink TFHE ciphertext expansion factor can be practically reduced to values *below 10*, depending on the setup, with little additional computational burden.

1 Introduction

Since its inception more than ten years ago, Fully Homomorphic Encryption has been the subject of a lot of research toward more efficiency and better practicality, with two main issues to be dealt with: the high computational cost of homomorphic operators and the large ciphertext expansion induced by FHE schemes. This paper focuses on the latter of these two issues, which leads to the following two problems, which are very different in nature depending on whether:

- Encrypted *inputs*, i.e. *freshly* encrypted ciphertexts, are transmitted towards a FHE calculation (which we refer to as the *uplink* case).
- Encrypted *results*, i.e. *evaluated* ciphertexts, obtained following some FHE calculation are transmitted towards an end-user for decryption (which we refer to as the *downlink* case).

* This work was supported by the France 2030 ANR Projects ANR-22-PECY-003 SecureCompute.

Reducing the expansion factor for the uplink case has received a lot of attention over the last ten years or so. Indeed, FHE ciphertext expansion can be almost cancelled in this case by a technique usually referred to as transciphering, which simply consists in transmitting data encrypted by means of a symmetric scheme (say, AES) and homomorphically turning them into FHE-encrypted data by running the symmetric scheme decryption algorithm over the FHE scheme. To do so, one has to pay the FHE expansion factor only to transmit a FHE encryption of the symmetric scheme key at setup time. As a result, many works focused on the issue of designing symmetric schemes amenable to practical homomorphic execution [2, 3, 18, 29, 33, 36, 37] or on optimizing the homomorphic execution of more standard ones [7, 8, 47]. Other approaches can also be applied to reduce that expansion factor. For instance, as all practical FHE schemes are based on (R)LWE, *in the symmetric setting* (where both encryption and decryption use the secret vector or polynomial $\mathbf{s}$), it is well known [1, 44] that one can simply synchronize the sender and the receiver on a PRF to avoid sending the a term in the (R)LWE pairs. This results in an expansion factor of $\log_2 q / \log_2 t$, where q and t respectively denote the ciphertext and plaintext moduli of the scheme. For typical TFHE parameters, this approach leads to an expansion factor of "only" 8, almost for free.

Unfortunately, techniques such as the above are not applicable to the downlink case. Indeed, the dream of being able to convert FHE-encrypted results back to AES form is an ill-posed problem for several reasons, the first of which being that, as transciphering requires homomorphically executing the decryption function of the source scheme under the target scheme, the technique applies only towards a homomorphic scheme, which is of course not the case with AES. Likewise, the above synchronization technique does not apply to the downlink case, as the a term in evaluated (R)LWE pairs cannot be a priori chosen. As such, compressing FHE calculation results for downlink transmission requires completely different approaches. To the best of our knowledge, the study of this issue has been initiated in [15], which was the first paper to suggest switching from an FHE scheme to a more compact linearly homomorphic scheme. For example, considering a LWE ciphertext $(\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, the idea simply consists in executing the dot product $b - \langle \mathbf{a}, \mathbf{s} \rangle$ under the LHE (given some form of encryption of $\mathbf{s}$ under the latter). Then one LHE ciphertext is transferred rather than $n+1$ elements in $\mathbb{Z}_q$, achieving some compression as soon as the size of an LHE ciphertext is smaller than $(n+1)\log_2 q$. Depending on the LHE at hand, several such dot products may be packed in a single LHE ciphertext in order to further enhance compression. Using this technique, they [15] further show that for GSW (with a binary plaintext domain and very specific parameters) and the Dåmgard-Jurik scheme as the LHE, it is possible to *asymptotically* achieve "rate 1" FHE (although this result is of theoretical interest as it relies on the fact that, for that latter scheme, the ratio between the plaintext modulus, N^k, and the ciphertext modulus, N^{k+1}, tends to 1 as k goes to infinity).

In essence, our paper builds on these ideas in a more practically minded fashion by focusing mainly on the non-asymptotic regime and considering sev-

eral possible candidates for the LHE, depending on their plaintext/ciphertext size ratios, the amount of "partially decrypted" messages that can be packed in their plaintexts, and the conditions under which they admit an efficient decryption (as some LHE require solving a discrete log during decryption). We do so with an explicit focus on TFHE [25]. It stands out in the FHE landscape as an intrinsically LWE-based scheme compared to the other ones, which are based on RLWE. Because of all the above degrees of freedom, some LHE are more appropriate than others depending on the number of (T)FHE ciphertexts that need to be transmitted. We also investigate how this approach can be combined with other (lossy) compression techniques based on modulus switching, a simple folklore method already used in several works [20, 34, 35, 48] for which we give rigorous bounds for the probability of erroneous decryption.

1.1 Summary of Contributions

The contributions of this paper can be summarized as follows:

- We address the issue of compressing *evaluated* (T)FHE ciphertexts to reduce the communication burden of transferring *results* from FHE computations prior to their decryption. Contrary to prior works, we do so in the non-asymptotic regime.
- We consider a simple, previously known (lossy) compression technique for TLWE and TRLWE ciphertexts, which consists in reducing the precision of their coefficients. Our contribution is then to carefully analyze the resulting noise increase and show how to choose the precision loss in order to comply with a preset probability of incorrect decryption. Then, we further demonstrate that this technique is a powerful tool when combined with other compression techniques.
- We propose a new "compressed" variant of the linearly homomorphic BCP03 cryptosystem with ciphertext size reduced to $\log \mu + |m|$, where μ is the modulus of the scheme and $|m|$ denotes the bitlength of the encrypted message m. This variant, which we refer to as *compressed Paillier-ElGamal* (CPG for short) in the sequel, may also be of independent interest.
- Building on the (known) idea of executing the linear part of the decryption function for a TLWE ciphertext over a more compact LHE scheme, we investigate several candidates for the LHE (including the above) in combination with other techniques from the state-of-the-art or the present paper to achieve a high compression rate of *evaluated* TFHE ciphertexts.
- We report extensive experimental results revealing the most appropriate regime for each technique (depending on parameters for TFHE as well as the number of FHE computation results that have to be transmitted).
- To the best of our knowledge, this paper is the first to demonstrate that expansion factors below 10 are practically achievable when transmitting results of FHE calculations, with limited additional computational burden.

1.2 Paper organization

This paper is organized as follows: Sect. 2 surveys existing techniques to reduce FHE ciphertext transmission overhead. Sect. 3 reviews the basics of TFHE (needed for understanding the paper) and gives the necessary details on the LHE that we use in this paper. Then, in Sect. 4, we present the compression techniques for evaluated TFHE ciphertexts. In Sect. 5, we study several combinations of techniques in order to achieve high compression rates of TFHE evaluated ciphertexts. We then report our experimental results in Sect. 6 and conclude the paper in Sect. 7.

2 Related Work

In this section, we review existing techniques to reduce FHE ciphertext transmission overhead and analyze whether they are applicable for downlink ciphertext compression *under the TFHE scheme* (which is the focus of this paper). Tab. 1 summarizes the comparison of the various approaches.

Paper	Uplink	Downlink	TFHE	BGV	BFV	CKKS	GSW
[21]	✓	✓	✓	✓	✓	✓	–
[6]	✓	–	–	✓	✓	✓	–
[5]	–	✓	✓	✓	✓	✓	–
[20]	✓	✓	✓	○	○	○	–
[34]	–	✓	○	✓	✓	○	–
[48]	–	✓	○	✓	✓	○	–
[15]	–	✓	○	○	○	○	✓
[35]	–	✓	✓	✓	✓	✓	–
Our	–	✓	✓	○	○	○	–

Table 1: Related work comparison (✓- yes, – - no, ○ - can be naturally applied).

In [21], the authors provide a toolkit to transform (R)LWE-based ciphertexts by providing methods to perform a key-switching operation between LWE ciphertexts, to transform an LWE ciphertext into an RLWE-based ciphertext, and to merge multiple LWE ciphertexts into a single RLWE ciphertext. Although the proposed keyswitch and packing methods can be applied to the TFHE scheme [25], they do not achieve any ciphertext compression (the goal of [21] is to reduce the storage requirement for key-switching keys and the computational cost of the procedure). The authors claim that they achieve lightweight and low-latency communication from the client to the server (thus, on the uplink), but do not address the downlink case.

Paper [6] is focused on LWE to RLWE ring packing after uplink ciphertext transmission. The objective is to efficiently obtain a CRT-encoded RLWE ciphertext to support component-wise multiplications and not to achieve any compression. Although the authors focus on the CKKS scheme [23], the proposed techniques are also applicable to BGV [12] and BFV [10, 31]. However, the approach cannot be easily applied for TFHE, as TRLWE samples use coefficient encoding. Moreover, their approach involves switching to larger ciphertext

moduli (rather than smaller), which is both difficult to perform in the LWE setting of TFHE and does not achieve compression.

In [5], the authors focus on verifiable computation over HE data. As a side contribution, they also address the issue of minimizing the HE ciphertexts overhead for a verifier. Their main idea is to avoid transmitting RLWE ciphertexts with unused slots by means of a repacking procedure that also switches to a smaller plaintext modulus (hence also a smaller ciphertext modulus), at the expense of dropping the homomorphic properties before downlink transmission. The proposed approach is generic and can be adapted to TFHE, BGV, BFV, and CKKS.

In [20], the authors present a constant bandwidth TFHE-based ORAM scheme in a single server model. On the downlink, they use a ModSwitch operation [12] for TRLWE samples to reduce the communication cost. For the uplink scenario, [6,20,21] use the PRF-based synchronization technique, proposed in [1,44], to avoid sending (R)LWE $\mathbf{a}$ coefficients. As already discussed, this technique is not applicable for the downlink, as in this case $\mathbf{a}$ cannot be a priori known.

Papers [34, 48] present HE-based protocols for neural networks. In both papers, to reduce FHE communication overhead, the authors use a technique similar to the one from [20]. Indeed, they achieve compression on the downlink by sending only some of the most significant bits in the components of the LWE ciphertexts. However, [20,34,48] provide only a very crude noise analysis and do not assess the impact of the technique on the probability of erroneous decryption.

Another approach attempting to reduce the downlink FHE communication burden consists in switching from a FHE scheme with linear decryption to a LHE scheme with a smaller expansion factor. This approach was first proposed in [15]. As the present paper, that paper is focused on techniques to compress post-evaluation FHE ciphertexts (the downlink case) and does not discuss compression techniques for the uplink case. In that setting, it introduces a general approach to build a rate-1 FHE by switching from GSW-style schemes [4,14,32,39] to the Dåmgard-Jurik cryptosystem or to a "shrinked" RLWE scheme (ciphertext shrinking is a technique introduced in that paper that has the nice property of not impacting the decryption error probability). The authors claim that a rate-1 expansion factor is achievable asymptotically (for very specific GSW parameters); as for the Dåmgard-Jurik cryptosystem, the rate $\frac{\log(\mu^y)}{\log(\mu^{y+1})} = 1 - \frac{1}{y+1}$ approaches 1 as y grows. In this case, however, the asymptotic growth of y also increases the computational cost of the latter scheme.

In [35], the authors put in an application for the idea from [15] of switching to the Paillier/Dåmgard-Jurik scheme. The authors also briefly mention the existence of different scheme switching/compression techniques (i.e., scheme switching as in [9], key/modulus switching, dimension reduction [13, 40], transciphering, etc.), but do not compare their resulting expansion factors. They also use the idea from [15] to pack several LWE samples into a single LHE ciphertext and, interestingly, further improve this packing by first switching to a smaller ciphertext modulus. However, they do not formally assess the induced increase in decryption error probability beyond "just" stating that the smaller modulus

q' should satisfy $q' \geq 2(n+1)t$ and investigate this technique only in the levelled BFV/BGV setting by then switching to the smaller modulus in the hierarchy. The authors experimentally conclude that this combined approach outperforms the other above LWE to RLWE packing approaches.

With respect to positioning, we revisit ideas from both [15] and [35] but adapt them to the specificities of the TFHE scheme, for which we provide a rigorous analysis of the induced decryption error probability, eventually leading to better compression ratios (4 to 5 times better than in [35]). We also consider a more exhaustive set of LHE depending on the number K of TFHE ciphertexts to transmit (including a new variant of the BPC03 scheme that allows us to achieve best-in-class compression in the regime where K is a few tens).

3 Preliminaries

3.1 General notation

In the upcoming sections, we denote vectors by bold letters, so a vector $\mathbf{x}$ of n elements is $\mathbf{x} = (x_0, \ldots, x_{n-1})$. The inner product of two vectors $\mathbf{x}$ and $\mathbf{y}$ is $\langle \mathbf{x}, \mathbf{y} \rangle$. $x \xleftarrow{\$} \mathbb{D}$ denotes sampling uniformly x from $\mathbb{D}$, and $x \xleftarrow{\mathcal{N}(0,\sigma^2)} \mathbb{D}$ denotes sampling x from $\mathbb{D}$ following a Gaussian distribution of mean 0 and variance σ^2.

3.2 TFHE

TFHE cryptosystem [25] was proposed by Chillotti et al. in 2016 and is notably implemented in the TFHE library [24]. It is intrinsically a LWE scheme working over the $[0, 1)$ torus, which we denote by $\mathbb{T}$. TFHE relies on three types of ciphertexts: TLWE, TRLWE, and TRGSW. In this paper, we focus only on the issue of compressing TLWE and TRLWE ciphertexts, as only those have to be transmitted when performing homomorphic calculations. TRGSW ciphertexts are only used temporarily within the bootstrapping procedure and never transmitted (except, offline, for transferring the bootstrapping key).

- TLWE ciphertext: a pair $(\tilde{\mathbf{a}}, \tilde{b})$ is a valid TLWE encryption of $m \in \mathbb{Z}_t$ (for plaintext modulus t), with $\tilde{\mathbf{a}} \xleftarrow{\$} \mathbb{T}^n$ and $\tilde{b} \in \mathbb{T}$ if it verifies $\tilde{b} = \langle \tilde{\mathbf{a}}, \mathbf{s} \rangle + \frac{m}{t} + \tilde{e}$, where $\mathbf{s} \xleftarrow{\$} \mathbb{B}^n$ is a TLWE secret key, and $\tilde{e} \xleftarrow{\mathcal{N}(0,\sigma^2)} \mathbb{T}$ is a noise term.
- TRLWE ciphertext: a pair $(\tilde{\mathbf{a}}, \tilde{\mathbf{b}})$ is a valid TRLWE encryption of $\mathbf{m} \in \mathbb{Z}_t[X]/(X^N+1)$, with $\tilde{\mathbf{a}} \xleftarrow{\$} \mathbb{T}_N[X]$ and $\tilde{\mathbf{b}} \in \mathbb{T}_N[X]$ if it verifies $\tilde{\mathbf{b}} = \tilde{\mathbf{a}} \cdot \mathbf{s} + \frac{\mathbf{m}}{t} + \tilde{\mathbf{e}}$, where $\mathbf{s} \xleftarrow{\$} \mathbb{B}_N[X]$ is a TRLWE secret key, and $\tilde{\mathbf{e}} \xleftarrow{\mathcal{N}(0,\sigma^2)} \mathbb{T}_N[X]$ is a noise polynomial. Here polynomials are represented by vectors of their coefficients.

Please note that, as this paper focuses on the input/output of FHE calculations, only a high-level understanding of the inner workings of the TFHE scheme and, in particular, its bootstrapping procedure is required to understand this work. We refer the reader to [25] for further details.

It should further be emphasized that TFHE is fully homomorphic only over TLWE ciphertexts. Up to N TLWE ciphertexts can be packed to/unpacked

from a single TRLWE ciphertext using standard techniques introduced in [11, 25, 43]. This may be useful for improving transmission efficiency (as we shall later discuss) or to perform batched homomorphic additions. As an LWE-based scheme, the TFHE decryption function is decomposed into a first linear part

$$\langle(\tilde{b}; -\tilde{\mathbf{a}}), (1; \mathbf{s})\rangle = \tilde{\varphi} = \frac{m}{t} + \tilde{\mathbf{e}} \tag{1}$$

followed by a scale (up) and round operation

$$\lceil t\tilde{\varphi} \rfloor = m. \tag{2}$$

In TFHE terminology, $\tilde{\varphi}$ is called the *phase* or the *partial decryption* of the ciphertext $(\tilde{\mathbf{a}}; \tilde{b})$. One key observation from [15] is that, given some encryption of $\mathbf{s}$ (with the appropriate form) over another LHE, (1) can be executed over the latter to get an encryption of φ under that LHE. How and when this is practically useful is investigated in the sequel. Equivalently, the scheme is always defined relative to a discretization of the torus by steps of $\frac{1}{q}$, where q is a power of two (typically either 2^{32} or 2^{64}). As such, TLWE or TRLWE ciphertexts are equivalently represented as LWE or RLWE ciphertexts over $\mathbb{Z}_q$ or $\mathbb{Z}_q[X]/(X^N + 1)$. When this representation is used, we will use the notations $(\mathbf{a}, b)$ and $(\mathbf{a}, \mathbf{b})$ instead of $(\tilde{\mathbf{a}}, \tilde{b})$ and $(\tilde{\mathbf{a}}, \tilde{\mathbf{b}})$ (with e.g., $\mathbf{a} = \lceil q\tilde{\mathbf{a}} \rfloor$ and similarly for the others). Note also that the phase computation (1) over $(\mathbf{a}, b)$ returns $\varphi = \Delta m + e$ with $\Delta = \frac{q}{t}$, and decryption is finalized by outputting

$$\left\lceil \frac{\varphi}{\Delta} \right\rfloor \tag{3}$$

instead of Eq. (2). We will use the illustrative parameters in Tab. 2 as a running example throughout the paper. This will allow us to provide illustrative numbers before the reader reaches Sect. 6 on experimental results, where several sets of TFHE parameters are investigated.

λ	n	q	t	N	σ_0	σ_{BS}
128	550	2^{32}	2	1024	$q \cdot 2.82 \times 10^{-4}$	$q \cdot 1.69 \times 10^{-2}$

Table 2: Illustrative TFHE parameters used as a running example.

3.3 Linear Homomorphic Encryption schemes

Tab. 3 provides the high-level characteristics of the LHE schemes that we consider in this paper (namely Paillier [42], Dåmgard-Jurik [30], Elliptic Curve ElGamal [38], and BCP03 [17]). As hinted in the previous Sect. we will consider using these schemes to evaluate the linear part of the TFHE decryption function, therefore converting one (or possibly more) ciphertext of size $(n + 1) \log_2 q$ into a single ciphertext of the LHE scheme. For completion, we briefly recall their definitions in Appendix Sect. A.

Cryptosystem	Plaintext domain	Ciphertext domain	Plaintext size (bits)	Ciphertext size (bits)	Expansion factor
Paillier	$\mathbb{Z}_\mu$	$\mathbb{Z}_{\mu^2}$	$\log_2 \mu$	$2\log_2 \mu$	2
Dåmgard-Jurik	$\mathbb{Z}_{\mu^y}$	$\mathbb{Z}_{\mu^{y+1}}$	$y \log_2 \mu$	$(y+1)\log_2 \mu$	$1+\frac{1}{y}$
EC ElGamal	$\mathbb{F}_\omega$	$\mathbb{F}_\omega^2$	$\mathfrak{p}$	$2\log_2 \omega$	$\frac{2\log_2 \omega}{\mathfrak{p}}$
BCP03	$\mathbb{Z}_\mu$	$\mathbb{Z}_{\mu^2}^2$	$\log_2 \mu$	$4\log_2 \mu$	4
CPG (Sect. 4.2)	$\mathbb{Z}_\mu$	$\mathbb{Z}_\mu \times \mathbb{Z}_{\mu^2}$	$\log_2 \mu$	$\log_2 \mu + \log_2 U$	$1+\frac{\log_2 U}{\log_2 \mu}$

Table 3: Summary of the main characteristics of the LHE schemes presented in Sect. A.1.

4 Compression building-blocks for evaluated TFHE ciphertexts

4.1 ℓ-truncation

Now we turn our attention to the basic (folklore) compression technique, which consists in dropping least significant bits in LWE or RLWE pairs coefficients (we refer to it as ℓ-truncation rather than modulus switching to emphasize that we floor rather than round). The main question is whether the additional noise that it finally induces may be small enough to allow significant compression without prohibitively increasing the probability for decryption errors to occur. The goal is then to carefully choose the number of discarded bits in order to comply with a preset probability of incorrect decryption, e.g., $\epsilon = 2^{-k}$ for $k = 40$, 64, or 128.

On one hand, TLWE ℓ-truncation can be combined with the LHE switching technique of Sect. 5 to increase the number of TLWE partial decryptions that can be packed into a single LHE ciphertext. On the other hand, its straightforward extension to TRLWE (Sect. C.2) is also useful as a stand-alone method: we can apply it after TLWE to TRLWE packing to reduce the size of the coefficient of a TRLWE sample before transmission[3].

We study how ℓ-truncation may be applied to a TLWE ciphertext. Let $\mathbf{c} = (a_0, \ldots, a_{n-1}, b = a_n)$ denote a TLWE encryption of m. Given $\ell < \lceil \log_2(q) \rceil$, we define the following three operations:

- $\mathsf{PartialDec}(\mathbf{c}, \mathbf{s})$: return $a_n - \langle \mathbf{a}, \mathbf{s} \rangle = \Delta m + e$, with $\Delta = \frac{q}{t}$.
- $\mathsf{Trunc}(\mathbf{c}, \ell)$: set $a'_i = \left\lfloor \frac{a_i}{2^\ell} \right\rfloor$ for $i \in \{0, ..., n\}$ and return $\mathbf{c}' = (a'_0, \ldots, a'_n)$.
- $\mathsf{Rescale}(\mathbf{c}', \ell)$: set $a''_i = 2^\ell a'_i$ for $i \in \{0, ..., n\}$ and return $\mathbf{c}'' = (a''_0, \ldots, a''_n)$.

[3] It is worth mentioning that the RLWE ℓ-truncation technique can be applied to BGV [12], BFV [10,31], and CKKS [23] RLWE ciphertexts. As these FHE cryptosystems use composite ciphertext modulus, it is natural to think of reducing it via rescaling before applying ℓ-truncation and sending the final result to the client. We leave this idea as a perspective for future explorations.

From these definitions, it follows that when $\mathbf{c}$ is a TLWE encryption of m with noise e (i.e., $\mathsf{PartialDec}(\mathbf{c}, \mathbf{s}) = \Delta m + e$), then $\mathbf{c}''$ is an encryption of m with noise

$$e'' = e - \sum_{i=0}^{n-1} e_i'' s_i + e_n'' \tag{4}$$

where $e_i'' = -(a_i \mod 2^\ell)$. Now, given a preset probability of decryption error $\epsilon = 2^{-k}$ and a noise variance σ for e, we would like to be able to choose ℓ such that an ℓ-truncated ciphertext decrypts correctly with probability at least $1 - \epsilon$, i.e., following Eq. (4) $\Pr\left(|e''| < \frac{\Delta}{2}\right) \geq 1 - \epsilon$. We then have the following proposition, which provides us with a first lower bound for the choice of ℓ.

Proposition 1. *Let $\mathbf{c}$ denote a TLWE encryption of m subject to a centered Gaussian noise e with variance σ^2, and let $\mathbf{c}' = \mathsf{Trunc}(\mathbf{c}, \ell_0)$ with*

$$\ell_0 \leq \left\lfloor \log_2 \left(\frac{1}{n+1} \left(\frac{\Delta}{2} - \sigma \sqrt{2(k+1)\log 2} \right) + 1 \right) \right\rfloor. \tag{5}$$

Then, $\left\lceil \frac{1}{\Delta} \mathsf{PartialDec}(\mathsf{Rescale}(\mathbf{c}', \ell_0), \mathbf{s})) \right\rfloor = m$ with probability at least $1 - 2^{-k}$.

Proof. Let us start by bounding the probability that $\mathbf{c}'' = \mathsf{Rescale}(\mathbf{c}', \ell)$ incorrectly decrypts, i.e., following (4), that

$$\begin{aligned} \Pr\left(|e''| \geq \frac{\Delta}{2}\right) &= \Pr\left(\left|e - \sum_{i=0}^{n-1} e_i'' s_i + e_n''\right| \geq \frac{\Delta}{2}\right), \\ &\leq \Pr\left(|e| \geq \frac{\Delta}{2} - (n+1)(2^\ell - 1)\right), \end{aligned}$$

as, by definition, $|e_i''| \leq 2^\ell - 1$. Assuming e follows a (centered) Gaussian distribution of variance σ^2, the Chernoff bound[4] tells us that $\Pr\left(|e''| \geq \frac{\Delta}{2}\right) \leq 2e^{-\frac{\left(\frac{\Delta}{2} - (n+1)(2^\ell - 1)\right)^2}{2\sigma^2}}$. Which, letting $2e^{-\frac{\left(\frac{\Delta}{2} - (n+1)(2^\ell - 1)\right)^2}{2\sigma^2}} = 2^{-k}$, leads to,

$$2^\ell = \frac{1}{n+1} \left(\frac{\Delta}{2} - \sigma \sqrt{2(k+1)\log 2} \right) + 1,$$

hence the claim. □

Applying the ℓ-truncation technique, we focus on the probability of decryption error rather than the variance or infinite norm of the noise. It allows us to ensure correct decryption after truncation for a preset probability of decryption error. To do so, we based our noise analysis on the Chernoff bound rather than

[4] Recall that for a (centered) Gaussian deviates, the two-sided Chernoff bound is such that $P(|X| \geq \alpha) \leq 2e^{-\frac{\alpha^2}{2\sigma^2}}$.

the erf function, as the former is more manageable for very low probabilities (e.g., 2^{-40}) and quite tight[5].

In general, we use Prop. 1 for evaluated ciphertexts, taking σ^2 as the post-bootstrapping variance σ^2_{BS}. To give an intuition, with our running TFHE parameter example (Tab. 2), for $\epsilon = 2^{-40}$, Prop. 1 leads to $\ell_0 = 19$, meaning a ciphertext size reduction of around 60%. Although it does not give us a closed-form formula for ℓ, we give a slightly more precise proposition in Sect. C.1.

4.2 Compressed Paillier-ElGamal (CPG)

In this section, we recall a standard variant of the BCP encryption scheme, which is commonly called Paillier-ElGamal [41, 45] and features smaller ciphertext compared to the traditional BCP encryption scheme ($3\log\mu$ bits versus $4\log\mu$). Then, we show that this variants additionally supports an efficient *compression* procedure, that allows to further reduce its ciphertext size down to $\log\mu + |m|$, where $|m|$ denotes the bitlength of the encrypted message m. This compression procedure is incompatible with the homomorphic features of the scheme, but can be used once all homomorphic operations have been computed, before sending the final ciphertext to the owner of the secret key.

A variant of BCP with shorter ciphertexts. The following variant of BCP is well-known and has roots in [27, 28]. It builds upon the fact that

- KeyGen: Let $\mu = pq$ be an RSA modulus. Choose a random $\alpha \in \mathbb{Z}^*_{\mu^2}$, a random value $d \in [1, \text{ord}(\mathbb{G})]$. Set $g = \alpha^2 \bmod \mu$ and $h = g^{\mu \cdot d} \bmod \mu^2$. Return a public key $pk = (\mu, g, h)$ and a secret key $sk = d$.
- Enc: For a given message $m \in \mathbb{Z}_\mu$, a random pad $r \xleftarrow{\$} \mathbb{Z}_{\mu^2}$ return a ciphertext $\mathbf{c} = (c_0, c_1)$ such that $c_0 = g^r \bmod \mu,\ c_1 = h^r(1+\mu)^m \bmod \mu^2$.[6]
- Dec: Compute $c = c_1(c_0)^{-\mu \cdot d} \bmod \mu^2$ and return $m = \frac{c-1}{\mu}$.

We make a few remarks on the above scheme. Compared with BCP, h is now computed as a μ-th power. This implies that $c_1 = h^r(1+\mu)^m = (g^{rd})^\mu(1+\mu)^m \bmod \mu^2$ (without the component c_0) is actually a valid Paillier encryption of m. Second, the component c_0 is now given modulo μ, reducing the ciphertext size by 25%. This is without loss of security, as one can easily check that $[g^r \bmod \mu]^\mu = g^{\mu r} \bmod \mu^2$.

Eventually, security-wise, the scheme can be proven IND-CPA secure under the DCR assumption; hence, it achieves identical security guarantees compared to Paillier. This follows from a sequence of straightforward game hops. We believe this proof to be essentially folklore. However, to our knowledge it has not been explicitly described anywhere, and we include it for convenience:

[5] A bound for ℓ_0, obtained using the erf, differs from (5) by one term ($\sqrt{k\log 2}$ vs $\text{erf}^{-1}(1-2^{-k})$), which is more challenging to compute. Furthermore, $\text{erf}^{-1}(1-2^{-k})$ is not defined for $k > 54$, while we want to determine ℓ_0 for $k = 64, 128$ as well.

[6] Note that $(1+\mu)^m \equiv 1+\mu m \mod \mu^2$ (from Binomial theorem applied to $(1+\mu)^m$).

- **Hybrid 1:** replace $h = g^{\mu d} \bmod \mu^2$ with a uniformly random $h \xleftarrow{\$} \mathbb{Z}^*_{\mu^2}$. Under the DCR assumption, this hybrid is indistinguishable from the real key generation algorithm.
- **Hybrid 2:** replace α with a random μ-th power $\alpha \leftarrow \beta^\mu \bmod \mu^2$ for $\beta \xleftarrow{\$} \mathbb{Z}_{\mu^2}$, and $g \leftarrow \alpha^2 \bmod \mu^2$. Under the DCR assumption, this hybrid is indistinguishable from the previous one.

Then, observe that in Hybrid 2, the message m is statistically hidden given a ciphertext (c_0, c_1). Indeed, the uniform distribution over $\mathbb{Z}_{\mu^2}$ is statistically close to the uniform distribution over $\mathbb{Z}_{\mu\cdot\phi(\mu)}$, because $\mu - \phi(\mu) = p + q - 1$ is of the order of $\sqrt{\mu}$. By the chinese remainder theorem, as μ is coprime to $\phi(\mu)$, $\mathbb{Z}_{\mu\cdot\phi(\mu)}$ is isomorphic to $\mathbb{Z}_\mu \times \mathbb{Z}_{\phi(\mu)}$. Write $h = (1+n)^a g^b \bmod \mu^2$ for some (a, b) (all elements admit such a decomposition), where $a \neq 0$ and is coprime to μ with overwhelming probability. Then, observe that since g generates a subgroup of order $\phi(\mu)/4$, the ciphertext c_0 leaks only the value $r_0 = [r \bmod \phi(\mu)/4]$. In contrast, $c_1 = h^r(1+\mu)^m = (1+\mu)^{ar_1 + m \bmod \mu} g^{br_0 \bmod \phi(\mu)/4} \bmod \mu^2$, where $r_1 = [r \bmod \mu]$ is statistically indistinguishable from random given r_0 (by coprimality of μ and $\phi(\mu)$). Hence, the value $ar_1 + m \bmod \mu$ statistically hides m. This concludes the proof. In the following, in line with previous works, we call the above scheme Paillier-ElGamal.

Distributed discrete logarithm. The scheme above enjoys shorter ciphertexts than BCP, but still larger than Paillier ($3\log\mu$ versus $2\log\mu$). In this section, we recall the *distributed discrete logarithm* procedure introduced in [41,45]. At a high level, this procedure allows two parties, given divisive shares of $(1+\mu)^m \bmod \mu^2$ over $\mathbb{Z}^*_{\mu^2}$, to non-interactively derive *substractive shares* of m over $\mathbb{Z}_\mu$. We outline the procedure DDLog_μ below.

Input. An element $u \in \mathbb{Z}^*_{\mu^2}$.
Output. A value $v \in \mathbb{Z}_\mu$.
Procedure. Write $u = u_0 + \mu \cdot u_1$, where $u_0, u_1 \in \mathbb{Z}_\mu$ denote the base-μ decomposition of u. Return $v = u_1/u_0 \bmod \mu$.

We now explain why this procedure has the intended behavior. Let u, u' denote two divisive shares over $\mathbb{Z}^*_{\mu^2}$ of $(1+\mu)^m \bmod \mu^2$; that is, $u'/u = (1+\mu)^m = 1 + \mu m \bmod \mu^2$. Writing $u = u_0 + \mu \cdot u_1$ and $u' = u'_0 + \mu \cdot u'_1$, we obtain $u'_0 + \mu \cdot u'_1 = (u_0 + \mu \cdot u_1) \cdot (1 + \mu \cdot m) \bmod \mu^2$. The above equation yields $u_0 = u'_0 \bmod \mu$ and $u'_1 = u_1 + u_0 m \bmod \mu$. Therefore, $m = u'_1/u'_0 - u_1/u_0 \bmod \mu$: u'_1/u'_0 and u_1/u_0 form subtractive shares of m over $\mathbb{Z}_\mu$, as intended.

Compressing ciphertexts via DDLog_μ. The distributed discrete logarithm procedure implies a simple and efficient compression mechanism for Paillier-ElGamal. The key observation is that given $c_0 = g^r \bmod \mu$, the holder of the secret key d can locally compute $u = c_0^{\mu\cdot d} = h^r \bmod \mu^2$. Then, u and c_1 form divisive shares of $c_1/u = (1+\mu m) \bmod \mu^2$. This immediately yields the following compression mechanism:

- $\mathsf{Compress}(c_0, c_1)$: run $v' \leftarrow \mathsf{DDLog}_\mu(c_1)$. Output (c_0, v').

– $\mathsf{Dec}'(c_0, v')$: compute $u \leftarrow c_0^{\mu \cdot d} \bmod \mu^2$ and $v \leftarrow \mathsf{DDLog}_\mu(u)$. Output $m = v' - v \bmod \mu$.

The resulting compressed ciphertext size is $2 \log \mu$, down from $3 \log \mu$, matching the size of a standard Paillier ciphertext. However, if m is known to be smaller than a bound $B < \mu/2^\lambda$ (where λ denotes a statistical security parameter), we can do better. The main observation (which is not new; the same observation was used in [41,45]) is that with overwhelming probability, v', v form subtractive shares of m *over the integers*. This stems from the following facts:

– Over the randomness of r, the value $v' = \mathsf{DDLog}_\mu(c_1) = \mathsf{DDLog}(h^r \cdot (1 + \mu m) \bmod \mu^2)$ is uniformly distributed over $\mathbb{Z}_\mu$.
– Then, if $m \leq B$, the probability that $v' - m$ causes a wraparound modulo μ is at most $B/\mu \leq 1/2^\lambda$, hence $v' - v = m$ over the integers.

This observation allows to further reduce the compressed ciphertext size by reducing v' modulo B:

– $\mathsf{Compress}(c_0, c_1)$: run $v' \leftarrow \mathsf{DDLog}_\mu(c_1)$ and set $v'' \leftarrow [v' \bmod B]$. Output (c_0, v'').
– $\mathsf{Dec}'(c_0, v')$: compute $u \leftarrow c_0^{\mu \cdot d} \bmod \mu^2$ and $v \leftarrow \mathsf{DDLog}_\mu(u)$. Output $m = v'' - v \bmod B$.

With this last optimization, the ciphertext size went down to $\log \mu + \log B$ bits. When B is small (e.g., $B \approx 2^{40}$ as in our application), this yields an almost twofold size improvement over a standard Paillier encryption. We note that a similar procedure has been previously described in the context of ElGamal encryption [16]. The Paillier-ElGamal variant, which we outline here, has the advantage of being extremely efficient, as compression amounts only to an inversion and a product modulo μ followed by a modular reduction.

5 Switching to LHE

In this section, we investigate several approaches to switch from TLWE homomorphic ciphertexts to (more compact) linearly homomorphic ones by executing the linear part of the (T)LWE decryption function, $b - \langle \mathbf{a}, \mathbf{s} \rangle$ (recall Eq. (1)), under the target linearly homomorphic scheme. As a result of this operation, we obtain encryptions of *partial decryptions* of TFHE ciphertexts under the target LHE. We consider several candidate LHE cryptosystems such as Paillier, Dåmgard-Jurik, EC ElGamal, and our compressed variant of BCP03 (Sect. 4.2). When the properties of the LHE allow it, we also consider packing the partial decryptions of *several* TFHE ciphertexts in a single LHE ciphertext in order to achieve better transmission efficiency when several evaluated TFHE ciphertexts have to be transmitted. To reduce the number of bits needed to encode a partial decryption (and hence be able to pack more partial decryptions per LHE ciphertext), we also investigate the use of this technique in conjunction with the ℓ-truncation technique, which we introduced in Sect. 4.1.

Overall, the most appropriate choice for the LHE depends on several factors, such as its plaintext/ciphertext ratio size, how many partial decryptions can be packed in its plaintexts, and the conditions under which it can decrypt efficiently (as some LHE require solving a discrete log in their decryption function). Because of all these degrees of freedom, some LHE are more appropriate than others for the purpose of transmitting a given number of evaluated TFHE ciphertexts.

5.1 A generic switching algorithm

Let $\mathcal{E}_H$ denote an instance of TFHE and $\mathcal{E}_L$ a target LHE. Let μ denote the plaintext modulus of $\mathcal{E}_L$ and v denote the number of bits necessary to represent a partial decryption (because μ is generally much greater than q, the partial decryptions are *not* computed modulo q, and we have to account for the carries occurring in their computation). Then up to

$$M = \left\lfloor \frac{\lfloor \log_2 \mu \rfloor}{v} \right\rfloor \tag{6}$$

partial decryptions can be packed into a single LHE ciphertext. Switching then works as follows. Let $\mathbf{s} \in \mathbb{B}^n$ denote $\mathcal{E}_H$'s secret key, and let $c_s^{(i)} = \mathcal{E}_L.\mathsf{Enc}(s_i)$ for $i \in \{0, ..., n-1\}$ denote encryptions of its coefficients under $\mathcal{E}_L$. We further denotes by $(\mathbf{a}^{(j)}, b^{(j)})$, $j \in \{0, ..., M-1\}$, the M TLWE pairs that we wish to convert. The conversion algorithm then starts with ciphertext $c = \mathcal{E}_L.\mathsf{Enc}(0)$. For $i = 0$ to $n-1$, we then perform, $c := c \oplus \left(-\sum_{j=0}^{M-1} 2^{jv} a_i^{(j)}\right) \odot c_s^{(i)}$, where $\oplus$ and $\odot$ respectively denote the addition and the multiplication-by-a-constant operator of $\mathcal{E}_L$ (remark that $-\sum_{j=0}^{M-1} 2^{jv} a_i^{(j)}$ lives in the clear domain of $\mathcal{E}_L$). Lastly, switching is finalized by doing[7]

$$c := c \oplus \mathcal{E}_L.\mathsf{Enc}\left(\sum_{j=0}^{M-1} 2^{jv} b^{(j)}\right). \tag{7}$$

Alg. 1 summarizes the above. As such, the algorithm terminates with an encryption of $\sum_{j=0}^{M-1} 2^{jv}\left(b^{(j)} - \langle \mathbf{a}^{(j)}, \mathbf{s}\rangle\right)$ but *without the modulo q which is implicit in Eq.* (1). After LHE decryption, one may recover the j-th partial decryption by doing

$$\varphi^{(j)} = \left(\left\lfloor \frac{\mathcal{E}_L.\mathsf{Dec}(c)}{2^{jv}} \right\rfloor \mod 2^v\right) \mod q, \tag{8}$$

and decryption is finalized using Eq. (3). Lastly, (8) has to be slightly modified as follows when ℓ-truncation is applied,

$$\varphi^{(j)} = 2^\ell \left(\left\lfloor \frac{\mathcal{E}_L.\mathsf{Dec}(c)}{2^{jv'}} \right\rfloor \mod 2^{v'}\right) \mod q,$$

where $v' < v$ is used instead of v in Alg. 1.

[7] When the target LHE provides an addition-by-a-constant operator, the latter can be used in Eq. (7) instead of invoking the encryption function of the scheme.

Algorithm 1 TLWEtoLHE

Input: Encryptions of $\mathcal{E}_H$'s secret key coefficients under $\mathcal{E}_L$, $c_s^{(i)} = \mathcal{E}_L.\mathsf{Enc}(s_i)$, M TLWE pairs $(\mathbf{a}^{(j)}, b^{(j)})$.
Output: $c \in \mathcal{E}_L.\mathcal{C}$ such that c is an encryption of $\sum_{j=0}^{M-1} 2^{jv} \left(b^j - \langle \mathbf{a}^j, \mathbf{s} \rangle\right)$.
1: $c = \mathcal{E}_L.\mathsf{Enc}(0)$
2: **for** $i = 0$, $i < n$, $i{+}{+}$ **do**,
3: $\quad c := c \oplus \left(-\sum_{j=0}^{M-1} 2^{jv} a_i^{(j)}\right) \odot c_s^{(i)}$,
4: **end for**
5: **return** $c := c \oplus \mathcal{E}_L.\mathsf{Enc}\left(\sum_{j=0}^{M-1} 2^{jv} b^{(j)}\right)$.

Considering our running TFHE parameters example of Tab. 2, with $q = 2^{32}$, we need $v = 42$ bits[8] to be able to represent a partial decryption prior to its reduction modulo q. This number goes down to 23 bits if we apply ℓ-truncation as proposed in Sect. 4.1 and drop 19 least significant bits in the coefficients of the LWE pairs prior to switching them (leading to a probability of erroneous decryption of 2^{-40}). We will use these numbers for illustration purposes in the next subsections.

5.2 Switching to Paillier

Recall Sect. 3.3, Paillier scheme has respective plaintext and ciphertext domains $\mathbb{Z}_\mu$ and $\mathbb{Z}_{\mu^2}$, where μ is an RSA modulus. Following the previous Sect. we can pack up to $M = \left\lfloor \frac{\lfloor \log_2 \mu \rfloor}{v} \right\rfloor$ partial decryptions in a single Paillier ciphertext. Let K denote a number of partial decryptions that need to be transmitted, and let $r_1 = K \bmod M$ and $r_2 = \lfloor K/M \rfloor$; then we have to pack these partial decryptions M-by-M (from Alg. 1) using r_2 Paillier ciphertexts when $r_1 = 0$ or $r_2 + 1$ such ciphertexts otherwise. When $r_1 = 0$, the expansion factor is then

$$\frac{\lceil 2 \log_2 \mu \rceil}{M \log_2 t} \tag{9}$$

(this is also the asymptotic expansion factor when $K \to \infty$), and, otherwise, the expansion factor is given by $\frac{(r_2+1)\lceil 2 \log_2 \mu \rceil}{K \log_2 t}$.

Considering an RSA modulus of 2048 bits (the usual recommendation to achieve 128 bits security) and our running example of TFHE parameters (Tab. 2), we can pack around $2048/42 \approx 48$ TLWE partial decryptions per Paillier ciphertext. For $K \leq 48$ we then get an expansion factor of $4096/K$, i.e. 4096 for $K = 1$ and around 85 for $K = 48$ (which is also the asymptotic expansion factor). If we apply ℓ-truncation, then we can pack around $2048/23 \approx 89$ partial decryptions in a single Paillier ciphertext. For $K \leq 89$, we obtain an expansion factor between 4096 ($K = 1$) and 46 ($K = 89$), this latter also being the

[8] As we have to sum $n + 1$ numbers (uniformly distributed) in $\{0, ..., q\}$, in the worst case, we get $(n+1)q$, which requires $\lceil \log_2(n+1) + \log_2 q \rceil$ bits. With n between 512 and 1024 and $q = 2^{32}$, 42 bits are conservatively required.

asymptotic expansion factor. *Keep in mind that these latter numbers must be compared with the raw expansion factor of 32800 induced by TLWE ciphertexts: for $K = 89$ (with ℓ-truncation) the expansion thus becomes 713 times smaller.* We explore more TFHE parameters in Sect. 6.

5.3 Switching to Dåmgard-Jurik

As an alternative to Paillier (and as initially considered in [15]), we may consider using the Damgård-Jurik scheme (recall the definition in Sect. 3.3), which generalizes the former scheme with $\mathbb{Z}_{\mu^y}$ and $\mathbb{Z}_{\mu^{y+1}}$ ($y > 1$) respectively as plaintext and ciphertext domains. Because the plaintext modulus is larger than for Paillier, it is possible to pack more TLWE partial decryptions into a single Dåmgard-Jurik ciphertext. Indeed, following Sect. 5.1, we can now pack up to $M = \left\lfloor \frac{\lfloor y \log_2 \mu \rfloor}{v} \right\rfloor$ partial decryptions in a single Dåmgard-Jurik ciphertext. As in the previous Sect., let K denote the number of partial decryptions that need to be transmitted, and let $r_1 = K \bmod M$ as well as $r_2 = \lfloor K/M \rfloor$, then we have to pack these partial decryptions M-by-M (Alg. 1) using r_2 Dåmgard-Jurik ciphertexts, when $r_1 = 0$, or $r_2 + 1$ such ciphertexts otherwise. When $r_1 = 0$, the expansion factor is then

$$\frac{\lceil (y+1) \log_2 \mu \rceil}{M \log_2 t} \tag{10}$$

(*for a fixed y*, this is also the asymptotic expansion factor when $K \to \infty$) and, otherwise, the expansion factor is given by $\frac{(r_2+1)\lceil (y+1)\log_2 \mu \rceil}{K \log_2 t}$. Interestingly, Dåmgard-Jurik asymptotically achieves the lowest expansion factor when both K and y increase to ∞. Indeed, (10) can be approximated by $\frac{v(y+1)\log_2 \mu}{y \log_2 \mu \log_2 t} = \frac{v(y+1)}{y \log_2 t}$ leading to $\lim_{y\to\infty} \frac{v(y+1)}{y \log_2 t} = \frac{v}{\log_2 t}$, which is the optimal rate achievable with the LHE switching technique.

Returning to our favorite running TFHE parameter example and considering, as for Paillier, a 2048-bit RSA modulus with $y = 2$, we can pack up to $M = 4096/42 \approx 97$ TLWE partial decryptions in a single Dåmgard-Jurik ciphertext. Then for $K = 97$ we illustratively obtain an expansion factor of around 63, this is also the asymptotic expansion factor (for fixed $y = 2$) which can then be compared with the value 85 obtained for Paillier. Lastly, putting ℓ-truncation into the picture leads to $M = 4096/23 \approx 178$ and an asymptotic expansion factor (again for fixed $y = 2$) of around 34. We compare the different methods and explore more TFHE parameters in Sect. 6.

5.4 Switching to compressed Paillier-ElGamal

We now consider packing TLWE partial decryptions within ciphertexts of the compressed Paillier-ElGamal (CPG) scheme that we introduced in Sect. 4.2. As for Paillier (and Dåmgard-Jurik), this scheme has a plaintext modulus μ. However, the size of a ciphertext with an l-bits payload ($l \leq \log_2(\mu)$) is only $l + \lceil \log_2 \mu \rceil$. As a consequence, this scheme is best used when small numbers of

TLWE partial decryptions have to be transmitted. As in the Paillier case, we can pack up to $M = \left\lfloor \frac{\lfloor \log_2 \mu \rfloor}{v} \right\rfloor$ partial decryptions in a single CPG ciphertext, and the two schemes achieve the same asymptotic expansion factor. However, when we wish to pack only $K \leq M$ partial decryptions in a CPG ciphertext, the resulting expansion factor is $\frac{Kv+\lceil \log_2 \mu \rceil}{K \log_2 t}$ compared to the Paillier case, which, recall Eq. (9), gives $\frac{2 \log_2 \mu}{K \log_2 t}$.

Returning again to our running TFHE parameters example of Tab. 2 (and reusing the numbers from the end of Sect. 5.2), we can pack up to 48 (w/o ℓ-truncation) or 89 (with ℓ-truncation) partial decryptions in a single CPG ciphertext. Without ℓ-truncation, for $K = 5$, 10, and 40 we then obtain respective expansion factors of 451, 247, and 93 (versus 819, 410, and 102 for Paillier). With ℓ-truncation into the picture, again for $K = 5$, 10, and 40, we respectively end up with 432, 228 and 74 (also versus 819, 410, and 102 for Paillier). More comparisons are provided in Sect. 6 on various parameter sets.

5.5 Switching to EC ElGamal

For completeness, we also briefly consider packing partial decryptions in EC ElGamal ciphertexts (recall the definition in Sect. 3.3). This scheme is the most compact that we consider in this work, but this compacity comes with the price of having to solve a discrete logarithm in the scheme decryption function. As such, we can only use it to encrypt small payloads (say, with a length of around or slightly above 40 bits). As a consequence, it is not possible to pack many TLWE partial decryptions in an EC ElGamal ciphertext, and this reduces its applicability to settings when no more than one or two evaluated TLWE ciphertexts have to be transmitted (using ℓ-truncation to decrease the number of bits needed to represent their partial decryptions). Still, in such cases, it is competitive with other approaches (transferring only one TLWE ciphertext always leads to the worst expansion factor, as the size of the LHE ciphertext is not amortized). Indeed, when a single TLWE ciphertext $(\mathbf{a}, b)$ has to be transmitted, switching to EC ElGamal by executing $b - \langle \mathbf{a}, \mathbf{s} \rangle$ over that cryptosystem will lead to a ciphertext of size around $\log_2 \omega$ bits, and an expansion factor will be $\frac{\lceil \log_2 \omega \rceil}{K \log_2(t)}$.

For our favorite running TFHE parameters example, with $t = 2$ and $K = 1$ we thus get 512 (which is the smallest expansion factor we obtain when transferring a single TLWE partial decryption, all the others LHE being in the thousands in that case). If we apply ℓ-truncation, we can either accelerate the decryption function (only a discrete log with an upper bound of 23 bits then needs to be solved) or attempt to pack two partial decryptions (needing 46 bits in total) in a single ciphertext. In that latter case, the expansion factor gets down to 256.

6 Experimental results and comparisons

In our experimental analysis, we consider two TFHE parameter sets. The first set is identical to the running example we have used so far for illustrative purposes (Tab. 2) and our second parameter set is for $t = 16$ meaning that TFHE

ciphertexts now have a 4-bit payload, rather than only 1 bit. We give the full parameter sets in Sect. D.

6.1 Relationship between ℓ and k

Recall Eq. (5) on page 9, which tells the number of bits ℓ that we can drop in the coefficients of a TLWE pair in function of a target probability of decryption error 2^{-k}. Since ℓ is influenced by $\log_2 k$ and because a flooring occurs in the formula (as only an integer number of bits can be dropped), we can expect that a given choice for ℓ covers a wide range of decryption error probabilities.

We illustrate this in Tab. 4 for T(R)LWE ℓ-truncation. The table also shows that in both cases, ℓ-truncation does not prevent achieving a $\mathrm{negl}(\lambda)$ probability of decryption error, although of course fewer bits have to be dropped than for larger probabilities of error. Note that the probability of bootstrapping error has to be set consistently with the probability of erroneous decryption induced by ℓ-truncation. For example, since our parameter set for $t = 2$ induces a probability of bootstrapping error of 2^{-154}, Tab. 4 tells us that we can drop up to 18 bits per coefficient and still achieve an *overall* probability of decryption error less than 2^{-128}, i.e. FHE correctness with overwhelming probability[9]. For comparison, in Sect. E we also provide a similar table (Tab. 8c) for ℓ values found using Prop. 2.

t	ϵ		
	2^{-40}	2^{-64}	2^{-128}
2	19.863057	19.408270	17.486391
16	17.217345	17.151246	17.010226
2	19.747898	19.203378	15.716669
16	17.217326	17.151221	17.010187

Table 4: Applying Prop. 1 for $t = 2, 16$ and several values for the probability of decryption error ϵ, the upper part represents a maximum value for ℓ when ℓ-truncating an evaluated (i.e., bootstrapped) TLWE ciphertext, and the bottom part – when ℓ-truncating a degree-N TRLWE ciphertext in which N n-dimensional evaluated TLWE ciphertexts have been packed.

6.2 Expansion factors

We now turn our attention to the expansion factor metric, which, recall, is defined as the ratio between the ciphertext size and corresponding plaintext size. The expansion factor formulas for the T(R)LWE-based compression techniques and the LHE-based ones can be found in the respective sections describing these methods and in Sect. E. In Tab. 5 we then provide a comparison between the expansion factor obtained by the different methods for $t = 16$ as a function of K, the number of evaluated TLWE ciphertexts that have to be transmitted.

[9] This remark is important, as ensuring correctness is a natural countermeasure against the recent CPA^D attacks against TFHE and other schemes [19, 22].

K	1	50	150	250	500	∞
TLWE	6008	6008	6008	6008	6008	6008
TLWE ℓ-truncation	2816.2	2816.2	2816.2	2816.2	2816.2	2816.2
Shrinking	16393	328.8	110.2	66.5	33.7	9
TLWEtoTRLWE	16392	335.6	117.2	73.5	40.7	16
TLWEtoTRLWE + ℓ-truncation	7683.75	157.35	55	34.5	19.11	**7.5**
Paillier (w. packing)	1024	40.9	27.3	24.5	22.5	21.3
ℓ-truncation + Paillier (w. packing)	1024	20.4	13.6	16.3	14.3	12.6
Dåmgard-Jurik (w. packing)	1536	30.7	20.4	18.4	18.4	15.8
ℓ-truncation + D.-J. (w. packing)	1536	30.7	**10.2**	**12.2**	**12.2**	9.4
CPG (w. packing)	522.5	31	24.1	22.7	21.7	21.1
ℓ-truncation + CPG (w. packing)	518.25	**16.5**	13	14.4	13.4	12.5
EC ElGamal	**128**	–	–	–	–	–

Table 5: Example expansion factors for each of the methods considered in this paper ($t = 16$).

Overall, again for $t = 16$, when $K = 1$ or 2, EC ElGamal achieves the smallest expansion factors (128 and 64, respectively). In the range $2 < K \leq 81$, the lowest expansion factor is achieved by performing ℓ-truncation and then switching to CPG (achieving, for example, an expansion factor of around 16.5 for $K = 50$). Up to $K \leq 1000$, Dåmgard-Jurik (with $y = 2$) gives the best compression, achieving, for example, an expansion factor of around 12 for $K = 250$ and 500. For $K \geq 1000$, TRLWE packing followed by ℓ-truncation becomes the best option and leads to an expansion factor of around 7. This is summarized[10] in Tab. 6. As discussed in Sect. 5.3, if we forget practicality for a moment and let the y parameter of Dåmgard-Jurik increase and apply ℓ-truncation with $\ell = 17$ (Tab. 4), we will eventually achieve the smallest possible expansion factor of $\frac{32-17}{4} = 3.75$. As already emphasized, *all these expansion factors must be compared to the raw expansion factor of* 6008 *induced by TLWE ciphertexts for* $t = 16$.

6.3 Computational costs

This section further investigates the computational cost of the methods described in this paper. For example, packing 1024 550-TLWE ciphertexts ($t = 2$) into a single TRLWE ciphertext takes 0.4 secs when implemented by means of TFHE-Lib. Then, switching 89 20-truncated partial decryptions ($t = 2$) to a Paillier ciphertext with a 2048-bit RSA modulus takes 3.41 secs[11] on a single-core, and

[10] We also give two illustrative plots (Figs. 1 and 2) of the expansion factors change in Sect. E.

[11] Note that it is faster to "decrypt-then-pack", which takes 2.83 secs (since computing the dot-product involves smaller numbers, hence leading to smaller exponent

K	Most compressive method	Timing		
$1 \leq K \leq 2$	Switch. to EC ElGamal	0.02	0.01	0.001
$2 < K \leq 81$	ℓ-truncation + switch. to CPG (w.pack.)	6.93	5.66	0.86
$81 < K \leq 163$	ℓ-truncation + switch. to D.-J. (w.pack.)	13.87	11.32	1.73
$163 < K \leq 243$	ℓ-truncation + switch. to CPG (w.pack.)	20.79	16.89	2.58
$243 < K \leq 1141$	ℓ-truncation + switch. to D.-J. (w.pack.)	97.09	79.24	12.11
$1141 < K \leq 1228$	TLWEtoTRLWE + ℓ-truncation	0.4		
$1228 < K \leq 1304$	ℓ-truncation + switch. to D.-J. (w.pack.)	110.96	90.56	13.84
$K > 1304$	TLWEtoTRLWE + ℓ-truncation	0.4		

Table 6: Most appropriate compression methods in function of K, the number of evaluated TLWE ciphertexts that have to be transmitted. The timings are given in seconds for the maximum value of K on the intervals: the first column corresponds to "pack-then-decrypt" switching, the second to "decrypt-then-pack," and the third to parallelized "decrypt-then-pack".

this number gets down to 0.42 secs with mild parallelization on an average laptop. For Dåmgard-Jurik, we obtain (estimated) timings of 10.24 secs to switch 178 20-truncated partial ($t = 2$) with a 2048-bit RSA modulus. Lastly, the runtimes for CPG are very close to those of Paillier. Additional timings (for $t = 16$) are reported in Tab. 6.

All these numbers illustrate that our methods run quite fast in practice, especially when comparing to the cost and latency of a typical homomorphic computation which is generally a matter of minutes, if not much more.

7 Conclusion

In this paper, we have proposed and experimentally studied a versatile and practical toolbox to address the issue of compressing evaluated (T)FHE ciphertexts, i.e., encrypted results obtained following the FHE evaluation of some useful function, to minimize their downlink transmission footprint towards decryption. To the best of our knowledge, while this issue is very important to FHE practice, it has so far been largely overshadowed in the literature by the issue of compressing input FHE ciphertexts, i.e., encrypted inputs towards the FHE evaluation of some useful function, to minimize their uplink transmission footprint from encryption. Still, the two issues are very different in nature, and their solutions require different corpora of techniques and tools. As key takeaways, we have revealed the regimes in which the techniques we have studied are best applicable, leading to the following concrete recommendations:

– Switching to EC ElGamal is the most compact option for transmitting a single evaluated TFHE ciphertext.

corresponding mult-by-const operations under Paillier) than to "pack-then-decrypt", which accounts for 3.41 secs.

- Switching to our new compressed variant of BCP03 (with several partial decryptions packed in each ciphertext) is the most communication-efficient option for transmitting up to around 100 evaluated TFHE ciphertexts.
- Above this value, we recommend switching to Damgard-Jurik (also with several partial decryptions packed in each ciphertext) for compressing larger numbers of evaluated TFHE ciphertexts, although this approach may eventually become too computationally costly, and, in that case, TRLWE packing will provide relatively similar compression factors with a much lower computational cost.

Additionally, all these approaches can be combined with a simple precision reduction technique, which, we have shown, still allows to keep a manageable probability of erroneous decryption. This technique allows to pack more partial decryptions in each LHE ciphertext and thus further enhance compression. Compared to the previous works, which have essentially focused on asymptotic rates from a more theoretical viewpoint, all these approaches are practically applicable.

As a concluding remark, let us emphasize that the techniques developed in this paper are applicable and beneficial only to LWE-based schemes such as TFHE. This is so for several reasons: first the LHE that we are considering in this paper have a plaintext domain that is too small to absorb the large N typically used for RLWE schemes such as BFV or BGV (which then achieve relatively low expansion factors of $2\log_2 q/\log_2 t$ for both uplink and downlink[12] and by default fall in the large K regime in the terminology of Sect. 6). This is also true for the ℓ-truncation technique. Indeed, as it significantly increases the ciphertext noise, it can be applied only to schemes with an efficient bootstrapping procedure (like TFHE), which then allows it to be applied to evaluated ciphertexts with a sufficient noise margin. Trying to apply this technique in the SHE setting for BFV or BGV would then require larger parameters and would most likely cancel the benefits of using ℓ-truncation in the first place. As a perspective, developing compression techniques practically applicable to partially-filled evaluated RLWE ciphertexts is an interesting follow-up research question.

References

1. Ajtai, M.: Representing hard lattices with $o(n \log n)$ bits. In: STOC. pp. 94–103 (2005)
2. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. In: EUROCRYPT. pp. 430–454 (2015)
3. Albrecht, M.R., et al.: Feistel structures for mpc, and more. In: ESORICS. pp. 151–171 (2019)
4. Alperin-Sheriff, J., Peikert, C.: Faster bootstrapping with polynomial error. In: CRYPTO. pp. 297–314 (2014)

[12] Provided of course that these ciphertexts have no or only few unused slots.

5. Aranha, D.F., Costache, A., Guimarães, A., Soria-Vazquez, E.: HELIOPOLIS: Verifiable computation over homomorphically encrypted data from interactive oracle proofs is practical. In: ASIACRYPT. pp. 302–334 (2024)
6. Bae, Y., et al.: HERMES: Efficient ring packing using MLWE ciphertexts and application to transciphering. In: CRYPTO. pp. 37–69 (2023)
7. Bendoukha, A., Boudguiga, A., Sirdey, R.: Revisiting stream-cipher-based homomorphic transciphering in the TFHE era. In: FPS. pp. 19–33 (2021)
8. Bendoukha, A., Clet, P., Boudguiga, A., Sirdey, R.: Optimized stream-cipher-based transciphering by means of functional-bootstrapping. In: DBSec. pp. 91–109 (2023)
9. Boura, C., et al.: CHIMERA: Combining ring-LWE-based fully homomorphic encryption schemes. In: Journal of Mathematical Cryptology. pp. 316–338 (2020)
10. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: CRYPTO. pp. 868–886 (2012)
11. Brakerski, Z., Gentry, C., Halevi, S.: Packed ciphertexts in LWE-based homomorphic encryption. In: PKC. pp. 1–13 (2013)
12. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: ITCS. pp. 309–325 (2012)
13. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: FOCS. pp. 97–106 (2011)
14. Brakerski, Z., Vaikuntanathan, V.: Lattice-based FHE as secure as PKE. In: ITCS. pp. 1–12 (2014)
15. Brakerski, Z., et al.: Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In: TCC. pp. 407–437 (2019)
16. Brakerski, Z., et al.: Constant ciphertext-rate non-committing encryption from standard assumptions. In: TCC. pp. 58–87 (2020)
17. Bresson, E., et al.: A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In: ASIACRYPT. pp. 37–54 (2003)
18. Canteaut, A., et al.: Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. In: Journal of Cryptology. pp. 885–916 (2018)
19. Checri, M., Sirdey, R., Boudguiga, A., Bultel, J.P.: On the practical cpad security of "exact" and threshold FHE schemes. In: CRYPTO. pp. 3–33 (2024)
20. Chen, H., Chillotti, I., Ren, L.: Onion ring ORAM: efficient constant bandwidth oblivious RAM from (leveled) TFHE. In: CCS. pp. 345–360 (2019)
21. Chen, H., Dai, W., Kim, M., Song, Y.: Efficient homomorphic conversion between (ring) LWE ciphertexts. In: ACNS. pp. 460–479 (2021)
22. Cheon, J.H., Choe, H., Passelègue, A., Stehlé, D., Suvanto, E.: Attacks against the IND-CPAD security of exact FHE schemes. In: CCS. pp. 2505–2519 (2024)
23. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: ASIACRYPT. pp. 409–437 (2017)
24. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: Fast fully homomorphic encryption library (August 2016), https://tfhe.github.io/tfhe/
25. Chillotti, I., et al.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In: ASIACRYPT. pp. 3–33 (2016)
26. Clet, P.E., Boudguiga, A., Sirdey, R., Zuber, M.: Combo: A novel functional bootstrapping method for efficient evaluation of nonlinear functions in the encrypted domain. In: AFRICACRYPT. pp. 317–343 (2023)
27. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: EUROCRYPT. pp. 45–64 (2002)
28. Damgård, I., Groth, J., Salomonsen, G.: The theory and implementation of an electronic voting system. In: Secure Electronic Voting. pp. 77–98 (2003)

29. Dobraunig, C., et al.: Pasta: A case for hybrid homomorphic encryption. In: IACR Trans. Cryptogr. Hardw. Embed. Syst. pp. 30–73 (2023)
30. Dåmgard, I., Jurik, M., Nielsen, J.B.: A generalization of paillier's public-key system with applications to electronic voting. In: Int. J. Inf. Sec. pp. 371–385 (2010)
31. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. In: IACR Cryptol. ePrint Arch., Paper 2012/144 (2012)
32. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: CRYPTO. pp. 75–92 (2013)
33. Hoffmann, C., Méaux, P., Ricosset, T.: Transciphering, using filip and TFHE for an efficient delegation of computation. In: INDOCRYPT. pp. 39–61 (2020)
34. Huang, Z., Lu, W., Hong, C., Ding, J.: Cheetah: Lean and fast secure two-party deep neural network inference. In: USENIX. pp. 809–826 (2022)
35. Mahdavi, R.A., Diaa, A., Kerschbaum, F.: He is all you need: Compressing fhe ciphertexts using additive he (2024), https://arxiv.org/abs/2303.09043
36. Méaux, P., Journault, A., Standaert, F., Carlet, C.: Towards stream ciphers for efficient FHE with low-noise ciphertexts. In: EUROCRYPT. pp. 311–343 (2016)
37. Méaux, P., et al.: Improved filter permutators for efficient FHE: better instances and implementations. In: INDOCRYPT. pp. 68–91 (2019)
38. Menezes, A.: Elliptic curve public key cryptosystems. Springer Science and Business Media, Volume 234 (1993)
39. Mukherjee, P., Wichs, D.: Two round multiparty computation via multi-key FHE. In: EUROCRYPT. pp. 735–763 (2016)
40. Naehrig, M., Lauter, K.E., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: CCSW. pp. 113–124 (2011)
41. Orlandi, C., Scholl, P., Yakoubov, S.: The rise of paillier: Homomorphic secret sharing and public-key silent OT. In: EUROCRYPT. pp. 678–708 (2021)
42. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: EUROCRYPT. pp. 223–238 (1999)
43. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: CRYPTO. pp. 554–571 (2008)
44. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC. pp. 84–93 (2005)
45. Roy, L., Singh, J.: Large message homomorphic secret sharing from DCR and applications. In: CRYPTO. pp. 687–717 (2021)
46. Trama, D., Clet, P.E., Boudguiga, A., Sirdey, R.: Designing a general-purpose 8-bit (T)FHE processor abstraction. In: Tr. Cr. Hardw. Emb. Syst. pp. 535–578 (2025)
47. Trama, D., Clet, P., Boudguiga, A., Sirdey, R.: A homomorphic AES evaluation in less than 30 seconds by means of TFHE. In: WAHC. pp. 79–90 (2023)
48. Xu, T., Li, M., Wang, R.: Hequant: Marrying homomorphic encryption and quantization for communication-efficient private inference. In: CoRR (2024)

A Linear Homomorphic Encryption schemes

Here we briefly recall Paillier [42], Dåmgard-Jurik [30], Elliptic Curve ElGamal [38], and BCP03 [17] cryptosystems, which are used in the sequel.

A.1 Paillier.

Paillier cryptosystem [42] is a public key encryption scheme invented by Pascal Paillier in 1999. It is a partial homomorphic scheme that notably allows performing additions and multiplications-by-a-constant directly over its ciphertexts.

- KeyGen: Choose two large prime numbers p and q randomly and independently such that $\gcd(pq, (p-1)(q-1)) = 1$. Compute $\mu = pq$, $\lambda = \varphi(\mu)$, and set $g = \mu + 1$, where $g \in \mathbb{Z}^*_{\mu^2}$, $\eta = \varphi(\mu)^{-1} \bmod \mu$. Return a public key $pk = (\mu, g)$ and a secret key $sk = (\lambda, \eta)$.
- Enc: Let m, $0 \leq m < \mu$, be a message to encrypt. Select a random $r : 0 < r < \mu$ and $\gcd(r, \mu) = 1$. Return a ciphertext $c \in \mathbb{Z}^*_{\mu^2}$ as $c = g^m \cdot r^\mu \bmod \mu^2$.
- Dec: Return $m = L(c^\lambda \bmod \mu^2) \cdot \eta \bmod \mu$, where $L(x) = \frac{x-1}{\mu}$.

In summary, for Paillier scheme, a plaintext domain is $\mathbb{Z}_\mu$ and a ciphertext domain is $\mathbb{Z}_{\mu^2}$. A typical parameter choice to achieve 128-bit security or slightly above consists in taking a modulus μ on 2048 bits. A Paillier ciphertext is therefore of size 4096 bits with a 2048-bit payload (leading to an expansion factor of 2).

A.2 Dåmgard-Jurik.

Dåmgard-Jurik cryptosystem [30] is a generalization of Paillier cryptosystem. It uses a plaintext modulus μ^y and a ciphertext modulus μ^{y+1}, where μ is an RSA modulus and $y \in \mathbb{Z}^+$. Paillier cryptosystem is the special case with $y = 1$.

- KeyGen: Choose an admissible RSA modulus $\mu = pq$ and compute $\lambda = \operatorname{lcm}((p-1)(q-1))$. Return a public key $pk = (\mu)$ and a secret key $sk = (\lambda)$.
- Enc: Let an integer $m > 0$ be a message to encrypt. Choose $y : m < \mu^y$ and select a random $r \in \mathbb{Z}^*_\mu$. Return a ciphertext $c = r^{\mu^y}(1+\mu)^m \bmod \mu^{y+1}$.
- Dec: Compute $c^\lambda \bmod \mu^{y+1} = (1+\mu)^{m\lambda} \bmod \mu^{y+1}$. Compute $v = m\lambda \bmod n^y$ (apply an algorithm from Theorem 1 [30]). Return $m = v\lambda^{-1} \bmod n^y$.

In summary, for Dåmgard-Jurik scheme, a plaintext domain is $\mathbb{Z}_{\mu^y}$ and a ciphertext domain is $\mathbb{Z}_{\mu^{y+1}}$. A typical parameter choice to achieve 128-bit security or slightly above consists in taking a modulus μ on 2048 bits. A Dåmgard-Jurik ciphertext is therefore of size $2048(y+1)$ bits with a $2048y$ bits payload. Note that a choice of y has no impact on security. Interestingly, an expansion factor for Dåmgard-Jurik scheme is asymptotically such that $\lim_{y\to\infty} \frac{2048(y+1)}{2048y} = 1$.

A.3 Elliptic Curve ElGamal.

Elliptic Curve ElGamal [38] is a public key additive homomorphic encryption scheme. It is a generalization of ElGamal public key cryptosystem over an elliptic curve.

- KeyGen: Choose a large prime ω and an elliptic curve $E(\mathbb{F}_\omega)$. Choose a point $P \in E$, an integer $a < \text{ord}(P)$ and compute $Q = aP$. Return a public key $pk = (E, P, Q)$ and a secret key $sk = (a)$.
- Enc: Let m, $0 \leq m < \omega$ be a message to encrypt. Express m as a point $X \in E$: $X = mP$. Choose random r and return the ciphertext $\mathbf{c} = (c_0, c_1)$ as $c_0 = rP,\ c_1 = X + rQ$.
- Dec: Compute $X = c_1 - ac_0$. Solve $X = mP$ and return m.

In summary, to set up the scheme, we choose an elliptic curve $y^2 = x^3 + ax + b \bmod \omega$ then a plaintext domain is $\mathbb{F}_\omega$ and a ciphertext domain is $\mathbb{F}_\omega^2$. A typical parameter choice to achieve 128-bit security or slightly above consists in taking an elliptic curve Curve25519. An EC-ElGamal ciphertext is therefore of size 510 bits with a 255-bit payload. However, let us also emphasize that the decryption function requires solving an elliptic curve discrete logarithm problem over Curve25519, which is practical only when a small upper bound is known for the decrypted message. This means that only $\mathfrak{p} \leq 50$ bits are truly usable from the plaintext domain.

A.4 BCP03.

BCP03 [17] is another public key additive homomorphic cryptosystem that is an ElGamal-style variant of Paillier scheme. The scheme is composed of three algorithms defined below.

- KeyGen: Let $\mu = pq$ be an RSA modulus. Choose a random $\alpha \in \mathbb{Z}^*_{\mu^2}$, a random value $d \in [1, \text{ord}(\mathbb{G})]$. Set $g = \alpha^2 \bmod \mu^2$ and $h = g^d \bmod \mu^2$. Return a public key $pk = (\mu, g, h)$ and a secret key $sk = (d)$.
- Enc: For a given message $m \in \mathbb{Z}_\mu$, a random pad $r \xleftarrow{\$} \mathbb{Z}_{\mu^2}$ return a ciphertext $\mathbf{c} = (c_0, c_1)$ such that $c_0 = g^r \bmod \mu^2,\ c_1 = h^r(1+\mu)^m \bmod \mu^2$.
- Dec: Compute $c = c_1(c_0)^{-d} \bmod \mu^2$ and return $m = \frac{c-1}{\mu}$.

In summary, for a *vanilla* variant of this scheme, a plaintext domain is $\mathbb{Z}_\mu$ and a ciphertext domain is $\mathbb{Z}^2_{\mu^2}$. A typical parameter choice to achieve 128-bit security or slightly above consists in taking a modulus μ of around 2048 bits. A BCP03 ciphertext is therefore of size 8192 bits with a 2048-bit payload (leading to an expansion factor of 4). As such, this scheme might not appear competitive with the other LHE presented in this section; we, however, propose in Sect. 5.4 a more advanced variant, which we will refer to as *compressed Paillier-ElGamal* (CPG), in which c_0 is compressed onto 2048 bits and c_1 onto $\log_2 U$ bits (where U is an upper bound on the encrypted message). As such, this new variant will be much more competitive.

B Existing *downlink* compression techniques for FHE

B.1 TLWE packing to one TRLWE

In this Sect. we discuss a *well-known* idea of packing (up to) $N > n$ TLWE samples into a single TRLWE sample in order to amortize the $\mathbf{a}$ vectors of TLWE

samples. Doing that, we would need to transmit just one $\mathbf{a}$ vector for N TLWE samples rather than one $\mathbf{a}$ vector for each of N TLWE samples. This TLWE samples packing technique is called TFHE Public Functional Key Switching, which was introduced in [25]. We use TFHE Public Functional Key Switching with a function being an identity function, which allows to pack N TLWE samples by means of the $\mathbb{Z}-$module isomorphism into a single TRLWE sample whereby N TLWE messages $m_0, \ldots, m_{N-1} \in \mathbb{T} \mapsto m(X) = \sum_{i=0}^{N-1} m_i X^i \in \mathbb{T}_N[X]$.

Noise Analysis: Because additional noise increases a probability of erroneous decryption, it is important to characterize a noise variance σ^2_{pack} in a TRLWE sample obtained from TLWEtoTRLWE packing of N TLWE samples with *independant* noises of variance bounded by σ^2_{BS}. From [25], the error variance σ^2_{pack} satisfies:

$$\sigma^2_{\text{pack}} \leq R^2\sigma^2_{\text{BS}} + nt_{\text{d}}N\sigma^2_{\text{TRLWE}} + \frac{n}{12}B_{\text{KS}}^{-2t_{\text{d}}}, \tag{11}$$

where $R = 1$ and σ^2_{TRLWE} is the variance[13] of the error in a keyswitch key (KS).

If $K < N$ TLWE samples have to be transmitted, it is possible to pack just $K < N$ TLWE samples into a single TRLWE sample assuming a *not-full packing* to avoid transmitting a part of the resulting TRLWE sample coefficients. It means that we "fill" the first K slots of TRLWE with TLWEs and keep $N-K$ slots empty. To transmit a not-fully packed TRLWE sample on the downlink, we transmit $\mathbf{a}$ and the first K coefficients of $\mathbf{b}$: $b_0, \ldots, b_{K-1}$. To decrypt this TRLWE sample, we perform a decryption of the first K slots of TRLWE.

Now, let's compute an expansion factor for TLWEtoTRLWE packing in the downlink case. Let a value $r_2 = \lfloor K/N \rfloor$ define how many fully packed TRLWE samples we get after packing K TLWE samples, and a value $r_1 = K \bmod N$ define how many TLWE samples are left to pack in a last not-fully packed TRLWE sample. It means that we need to transmit r_2 fully packed TRLWE samples and one not-fully packed TRLWE sample with r_1 (> 0) packed coefficients; then the size of the resulting ciphertexts is $2r_2N\log_2 q + (N + r_1)\log_2 q$. At the same time, the resulting ciphertexts encrypt $r_2N + r_1$ plaintext messages of size $\log_2 t$; thus, for $r_1 > 0$ the expansion factor is $\frac{(2r_2N+N+r_1)\log_2 q}{(r_2N+r_1)\log_2 t}$, and for $r_1 = 0$: $\frac{2\log_2 q}{\log_2 t}$. Then, for example, for $t = 2$ and $K = N = 1024$ the expansion factor is 64.

B.2 Shrinking

The so-called shrinking technique, first proposed in [15], allows for compressing ciphertexts from a FHE scheme that supports linear decrypt-and-multiply, like the GSW scheme [32]. It can also be applied directly to compress BFV-style ciphertexts as well as TRLWE ones. In a nutshell, given a TRLWE sample $(\mathbf{a}, \mathbf{b})$, shrinking consists of computing two values, a 'helper' $r \in \mathbb{Z}_q$ and a value $w \in \mathbb{Z}_t$

[13] We use the identity function f as a public R-Lipschitz morphism; thus, $R = 1$; and note that the keyswitch key KS is a fresh TRLWE sample; hence, the variance of the error in KS is σ^2_{TRLWE}.

such that the decryption of the original ciphertext (which is not necessarily correct) can be recovered *exactly*[14] from r and w (as well as, of course, the knowledge of the secret key). Shrinking is interesting because a single helper value 'r' can be used to cover the N LWE samples assembled in a RLWE ciphertext.

In a nutshell, shrinking works as follows. Let $\mathbf{c} = (\mathbf{a}, \mathbf{b})$ be a TRLWE sample we need to compress, and let $\mathbf{s}$ be a TRLWE secret key. To shrink the TRLWE sample, we parse $\mathbf{b} = (b_0, \ldots, b_{N-1}) \in \mathbb{Z}_q^N$ and compute the union of intervals $U \subseteq \mathbb{Z}_q$, where $B = \sigma_{\text{TLWE}}$ is a TLWE noise bound.

$U = \bigcup_{i=0}^{N-1} \left(\left[\frac{\Delta}{2} - b_i - B, \frac{\Delta}{2} - b_i + B \right] \cup \left[-\frac{\Delta}{2} - b_i - B, -\frac{\Delta}{2} - b_i + B \right] \right)$, for $\Delta = q/t$. Then we pick any $r \in \mathbb{Z}_q \backslash U$ and for $i = [0, N-1]$ compute $w_i = \lceil b_i + r \rfloor_t$, where $\lceil x \rfloor_t = \left\lfloor x \cdot \frac{t}{q} \right\rceil \mod t$ is a rounding function. The resulting shrunk sample is $\tilde{\mathbf{c}} = (r, \mathbf{a}, w_0, \ldots, w_{N-1})$. This therefore leads to an overhead of

$$\frac{(N+1)\log_2 q + N \log_2 t}{N \log_2 t} \tag{12}$$

i.e., for $N = 1024$ ($t = 2$) it is ≈ 33 or for $N = 2048$ ($t = 16$) ≈ 9. If only $K < N$ LWE samples are to be transmitted (we assume not-full packing for TRLWE), then, trivially from (12), the expansion factor is $\frac{(N+1)\log_2 q + K \log_2 t}{K \log_2 t}$.

To decrypt the shrunk TRLWE sample on the downlink, we do the following: we compute $\mathbf{v} = \mathbf{s} \cdot \mathbf{a}$ and parse $\mathbf{v} = (v_0, \ldots, v_{N-1})$. For $i \in [0, N-1]$ we then compute $m'_i = (w_i - \lceil v_i \rfloor_t) \mod t$ and output $m' = (m'_0, \ldots, m'_{N-1})$. The equivalence between this decryption function and the original one then follows from Lemma 1 of [15]. Let us emphasize, however, that when used for compressing several TLWE ciphertexts, Shrinking affects the probability of erroneous decryption, as we can apply Shrinking only to TRLWE ciphertexts. As discussed in the previous Sect. packing several TLWE increases the noise deviation.

C Further details on ℓ-truncation

C.1 Refined TLWE ℓ-truncation

Although it does not give us a closed-form formula for ℓ, the following proposition is a bit more precise than Prop. 1.

Proposition 2. *Let* $\mathbf{c}$ *denote a TLWE encryption of* m *subject to a centered Gaussian noise* e *with variance* σ^2*, and let* $\mathbf{c}' = \mathsf{Trunc}(\mathbf{c}, \ell)$*, then*

$$\left\lceil \tfrac{1}{\Delta} \mathsf{PartialDec}(\mathsf{Rescale}(\mathbf{c}', \ell_0), \mathbf{s}) \right\rfloor = m \text{ with probability at least}$$

$$1 - 2\exp\left(- \frac{\left(\frac{\Delta}{2} - \frac{1}{2}(n-1)(2^\ell - 1) \right)^2}{2\left(\sigma^2 + \frac{1}{12}(n+1)(2^{2\ell} - 1) \right)} \right).$$

[14] This is important for BFV-style schemes, which tend to induce large noise variances in evaluated ciphertexts.

Proof. Recall Eq. (4), we have $e'' = e + \sum_{i=0}^{n-1} \underbrace{(a_i \mod 2^\ell)}_{-e''_i} s_i - \underbrace{(a_n \mod 2^\ell)}_{-e''_n}$.

Under the assumption that q is a power of 2, $a_i \bmod 2^\ell$ ($i \in \{0, ..., n-1\}$) is uniformly distributed over $\{0, ..., 2^\ell - 1\}$ as the associated a_i's are uniformly distributed in $\mathbb{Z}_q$. Additionally, $a_n \bmod 2^\ell$ is also uniformly distributed over $\{0, ..., 2^\ell - 1\}$ as, following the LWE assumption, $b = a_n$ is indistinguishable from a uniform deviate over $\mathbb{Z}_q$. Let $n' = \sum_i s_i$; from the CLT, we can thus assume that e'' follows a Gaussian distribution with expectation $E[e''] = \frac{1}{2}(n' - 1)(2^\ell - 1) \leq \frac{1}{2}(n-1)(2^\ell - 1)$, and variance[15] $V[e''] = \sigma^2 + \frac{1}{12}(n'+1)(2^{2\ell} - 1) \leq \sigma^2 + \frac{1}{12}(n+1)(2^{2\ell} - 1)$. The claim follows from applying the Chernoff bound[16] to $\Pr\left(|e''| \geq \frac{\Delta}{2}\right)$. □

Using the latter Prop. in conjunction with Prop. 1 usually allows to slightly increase the number of bits that may be dropped off. For example, as discussed just above, with our running TFHE parameter set example, Prop. 1 tells that for $\epsilon = 2^{-40}$, $l_0 = 19$. Prop. 2, however, tells us that the probability of decryption error is bounded by $2^{-116.06}$ for that value (part of that gap is explained by the ceiling that occurs in (5), as we can only drop an integer number of bits). Then, if we choose $\ell = 20$, the bound drops to $2^{-81.78}$, still above our 2^{-40} target. As this is the cutoff value, we can finally settle on $\ell = 20$ meaning a ciphertext size reduction of 62.5%. Overall, the expansion factor goes from 8816 down to 3306, i.e., is reduced by a factor of 2.66.

C.2 TRLWE ℓ-truncation

ℓ-truncation can equally be applied to TRLWE ciphertext by dropping ℓ least bits on all the coefficients of the $\mathbf{b}$ polynomial. Because this is very similar to the TLWE case, we do not provide further details. For TRLWE, we also have the analogous of Prop. 1 with Eq. (5) replaced by (recall from Sect. B.1 that, by default, we pack N n-dimensional TLWE samples in a single degree-N TRLWE)

$$\ell_0 \leq \left\lfloor \log_2 \left(\frac{1}{n+1} \left(\frac{\Delta}{2} - \sqrt{-2\sigma^2 \log(1 - \sqrt[N]{1 - 2^{-(k+1)}})} \right) + 1 \right) \right\rfloor .$$

which provides the guarantees that $\Pr\left(||\mathbf{e}''||_\infty \geq \frac{\Delta}{2}\right) \leq 2^{-k}$. However, let us emphasize that the above equation bounds the probability that an ℓ_0-truncated TRLWE ciphertext decrypts incorrectly, meaning that *a decryption error occurs in at least one slot of the message polynomial*. Since the present work is focusing on TFHE and is therefore TLWE-centric, we are only using TRLWE ciphertexts as a mean to more efficiently transmit TLWE ciphertexts, and, in fine, it is the decryption error probability of these TLWE ciphertexts that is important to us.

[15] Recall that the variance of the discrete uniform distribution over $\{a, ..., b\}$ is $\frac{1}{12}((b - a + 1)^2 - 1)$, leading to $\frac{1}{12}(2^{2l} - 1)$ when $a = 0$ and $b = 2^\ell - 1$.

[16] For a Gaussian deviate of expectation $\mu \geq 0$ and variance σ^2, it holds that $P(|X| \geq \alpha) \leq 2e^{-\frac{(\alpha-\mu)^2}{2\sigma^2}}$. Furthermore, for $\mu' \geq \mu$ and $\sigma' \geq \sigma$, $e^{-\frac{(\alpha-\mu)^2}{2\sigma^2}} \leq e^{-\frac{(\alpha-\mu')^2}{2\sigma'^2}}$.

So despite the fact that we may use TRLWE ciphertexts for transmission, we stick to the tools provided in the previous section in the sequel. However, in using Prop. 1 and 2 to choose the number of bits that can be dropped from coefficients of a TLWE pair embedded in a TRLWE ciphertext, we have to take into account the extra variance induced by packing following Eq. (11).

Then, with our running TFHE parameter set example, Prop. 1 tells that for $\epsilon = 2^{-40}$, $l_0 = 19$. Prop. 2, however, tells us that the probability of decryption error is bounded by $2^{-100.55}$ for that value. Then, if we choose $\ell = 20$, the bound drops to $2^{-70.86}$, still above our 2^{-40} target. As this is the cutoff value, we can finally settle on $\ell = 20$ (as in the previous Sect., meaning that TRLWE packing does not affect how much we can truncate in the present setting). Assuming one fully packed TRLWE, we get a ciphertext size reduction of 56.25%. Overall, the expansion factor goes from 64 (Sect. B.1) down to 24, i.e., is reduced by a factor of 2.66.

D TFHE parameters

In our experimental analysis, we consider the two TFHE parameter sets given in Tab. 7. The first set is identical to the running example we have used so far for illustrative purposes. This first parameter set is consistent with the "standard TFHE gate bootstrapping" approach where TFHE is configured for performing operations over binary plaintexts (i.e., $t = 2$). This first parameter set achieves an error probability for bootstrapping of 2^{-154}. Our second parameter set is for $t = 16$, meaning that TFHE ciphertexts now have a 4-bit payload. This set is the most interesting because, as we shall see, the increased payload length will consistently lead to the smallest expansion factors in our experiments. Furthermore, several recent works, notably [46, 47], hint that $t = 16$ may achieve an optimal tradeoff between the bootstrapping time (which increases with t) and the number of operations (which decreases with t) required when executing (useful) algorithms over TFHE. However, for $t = 16$, the bootstrapping error probability is increased to 2^{-46} and cannot be significantly lowered unless one is willing to increase q to 2^{64} (and also mechanically increase n). We do not consider this latter option, as it would result in a large performance hit for the FHE calculations themselves. Our two parameter sets achieve 128-bit security according to the lattice-estimator and have been obtained following the methodology in [26]. Note that, since we have used our parameter set for the case where $t = 2$ as a running example throughout the paper, the present section focuses essentially on the case where $t = 16$.

t	q	n	N	l	t_{d}	B_{g}	B_{KS}	σ_{TLWE}	σ_{TRLWE}
2	2^{32}	550	1024	2	1	256	1024	$q \cdot 2.82 \times 10^{-4}$	$q \cdot 5.6 \times 10^{-8}$
16	2^{32}	750	2048	2	2	1024	1024	$q \cdot 7.7 \times 10^{-6}$	$q \cdot 9.6 \times 10^{-11}$

t	σ_{BR}	σ_{KS}	σ_{BS}	σ_{pack}
2	$q \cdot 1.12 \times 10^{-2}$	$q \cdot 1.27 \times 10^{-2}$	$q \cdot 1.69 \times 10^{-2}$	$q \cdot 1.81 \times 10^{-2}$
16	$q \cdot 3.62 \times 10^{-4}$	$q \cdot 4.92 \times 10^{-4}$	$q \cdot 6.1082 \times 10^{-4}$	$q \cdot 6.1087 \times 10^{-4}$

Table 7: Our TFHE parameter sets are for $t = 2$ (binary payloads) and 16 (4-bit payloads). The full set of parameters is provided for completion and reproducibility of our results, but we do not detail the meaning of them all. As of the second of the above tables, it provides the post-bootstrapping noise standard deviation σ_{BS} (which characterizes the noise present in evaluated ciphertexts) and the post-packing standard deviation (obtained after packing N, n-dimensional evaluated TLWE ciphertexts, hence with a noise deviation of σ_{BS}, in a *single* degree-N TRLWE ciphertext). The other two deviations are post-Blind Rotation (BR) and post-KeySwitch (KS), but we did not need to detail these operations in this paper.

E Additional tables and plots

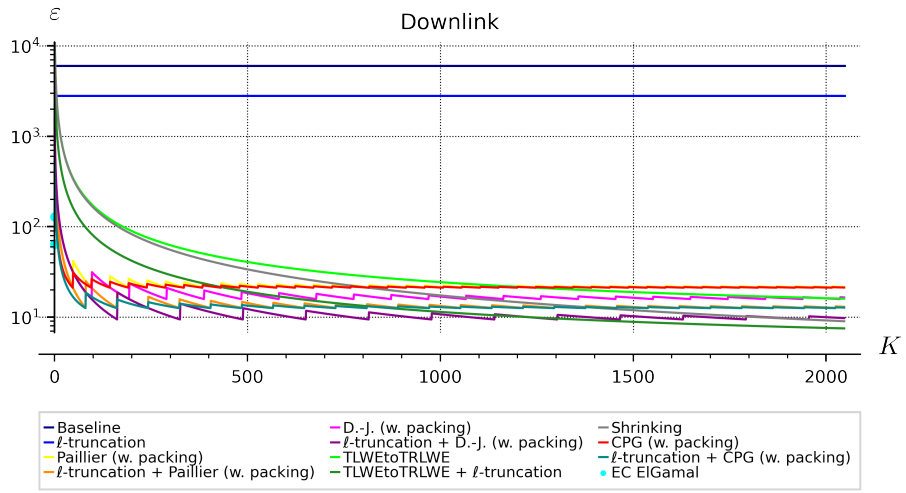

Fig. 1: Comparison of expansion factors for the downlink ciphertext compression methods studied in this paper and ℓ found using Prop. 1 ($t = 16$)

Method	Exp. factor
TLWE	$\frac{(n+1)\log_2 q}{\log_2 t}$
TLWE ℓ-truncation	$\frac{(n+1)(\log_2 q-\ell)}{\log_2 t}$
Shrinking	$\frac{(N+1)\log_2 q+K\log_2 t}{K\log_2 t}$
TLWEtoTRLWE	$\frac{(2r_2N+N+r_1)\log_2 q}{K\log_2 t}$
TLWEtoTRLWE + ℓ-truncation	$\frac{(2r_2N+N+r_1)(\log_2 q-\ell)}{K\log_2 t}$

(a) Expansion factor formula for the different methods based on TLWE/TRLWE when K evaluated TLWE ciphertexts have to be transmitted on the downlink. For RLWE-based methods, $r_1 = K \bmod N$ and $r_2 = \lfloor K/N \rfloor$ when the K n-dimensional TLWE are packed N-by-N in degree-N TRLWE ciphertexts.

Compression method	M	Exp. factor
Paillier (w. packing)	$\frac{\lfloor \log_2 \mu \rfloor}{v}$	$\frac{(r_2+1)\lceil 2\log_2 \mu \rceil}{K\log_2 t}$
ℓ-truncation + Paillier (w. packing)	$\frac{\lfloor \log_2 \mu \rfloor}{v-\ell}$	
Dåmgard-Jurik (w. packing)	$\frac{\lfloor y\log_2 \mu \rfloor}{v}$	$\frac{(r_2+1)\lceil (y+1)\log_2 \mu \rceil}{K\log_2 t}$
ℓ-truncation + D.-J. (w. packing)	$\frac{\lfloor y\log_2 \mu \rfloor}{v-\ell}$	
CPG (w. packing)	$\frac{\lfloor \log_2 \mu \rfloor}{v}$	$\frac{Kv+(r_2+1)\lceil \log_2 \mu \rceil}{K\log_2 t}$
ℓ-truncation + CPG (w. packing)	$\frac{\lfloor \log_2 \mu \rfloor}{v-\ell}$	$\frac{K(v-\ell)+(r_2+1)\lceil \log_2 \mu \rceil}{K\log_2 t}$
EC ElGamal	2	$\frac{\lceil \log_2 \omega \rceil}{K\log_2 t}$

(b) Expansion factor formula for the different LHE-based methods when K evaluated TLWE ciphertexts have to be transmitted on the downlink. Above, M is the number of TLWE partial decryptions that can be packed in a single LHE ciphertext (and v the number of bits required to represent one such partial decryption). Let $r_1 = K \bmod N$ and $r_2 = \lfloor K/N \rfloor$ when the K TLWE partial decryptions are packed M-by-M in LHE ciphertexts.

t	ϵ		
	2^{-40}	2^{-64}	2^{-128}
2	20	20	18
16	18	18	17
2	20	20	16
16	18	18	17

(c) Applying Prop. 2 for $t = 2, 16$ and several values for the probability of decryption error ϵ, the upper part represents a maximum value for ℓ when ℓ-truncating an evaluated (i.e., bootstrapped) TLWE ciphertext, and the bottom part – when ℓ-truncating a degree-N TRLWE ciphertext in which N n-dimensional evaluated TLWE ciphertexts have been packed.

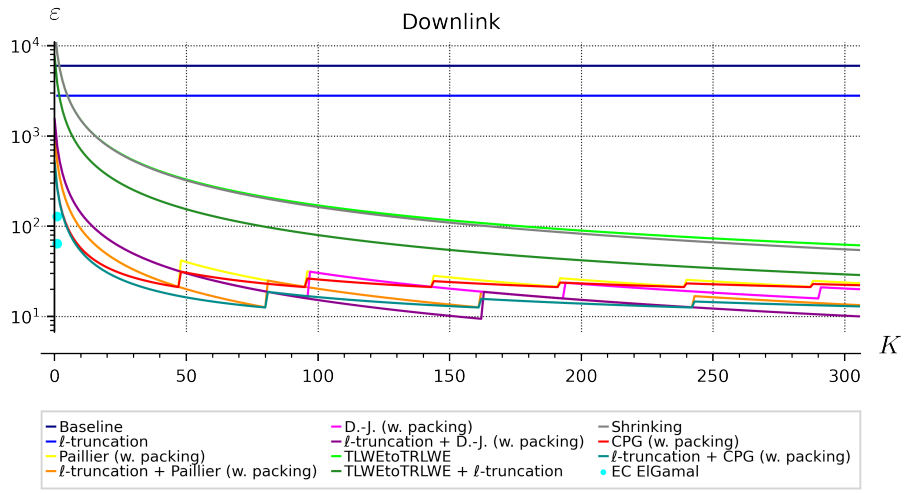

Fig. 2: Comparison of expansion factors for the downlink ciphertext compression methods studied in this paper for $K \leq 300$ and ℓ found using Prop. 1 ($t = 16$)