Bootstrapping for Approximate Homomorphic Encryption with Negligible Failure-Probability by Using Sparse-Secret Encapsulation

Jean-Philippe Bossuat[13], Juan Ramón Troncoso-Pastoriza[13], and Jean-Pierre Hubaux[12]

[1] Tune Insight SA, Switzerland
first@tuneinsight.com
[2] École polytechnique fédérale de Lausanne (EPFL)
first.last@epfl.ch
[3] Part of this work was carried out at EPFL

Abstract. Bootstrapping parameters for the approximate homomorphic-encryption scheme of Cheon et al., CKKS (Asiacrypt 17), are usually instantiated using sparse secrets to be efficient. However, using sparse secrets constrains the range of practical parameters within a tight interval, as they must support a large enough depth for the bootstrapping circuit but also be secure with respect to the sparsity of their secret.
We present a bootstrapping procedure for the CKKS scheme that combines both dense and sparse secrets. Our construction enables the use of parameters for which the homomorphic capacity is based on a dense secret, yet with a bootstrapping complexity that remains the one of a sparse secret and with a large security margin. Moreover, this also enables us to easily parameterize the bootstrapping circuit so that it has a negligible failure probability that, to the best of our knowledge, has never been achieved for the CKKS scheme. When using the parameters of previous works, our bootstrapping procedures enables a faster procedure with an increased precision and lower failure probability. For example we are able to bootstrap a plaintext of $\mathbb{C}^{32768}$ in 20.2 sec, with 32.11 bits of precision, 285 bits of modulus remaining, a failure probability of $2^{-138.7}$ and 128 bit security.

Keywords: Fully Homomorphic Encryption · Bootstrapping · Implementation

1 Introduction

1.1 The CKKS Scheme

The CKKS scheme by Cheon et al. [10] is a *leveled* ring learning with error (R-LWE) [23] homomorphic-encryption scheme that enables approximate arithmetic over vectors of complex numbers. Since its introduction, this scheme has grown in popularity, as it is currently the most efficient scheme for performing encrypted floating-point arithmetic. Ciphertexts are pairs of polynomials of $R_Q = \mathbb{Z}_Q[X]/(X^N+1)$, with the main cryptographic parameters being the polynomial-ring degree N and its modulus Q; and for a given security parameter λ and fixed N, an upper bound on Q can be derived (a smaller Q leads to a more secure instance).

A fresh CKKS ciphertext is of the form $(c_0, c_1) = (-as + m + e, a) \in R_Q^2$ for a a random polynomial, s and e low-norm secret polynomials and m a message polynomial. The decryption is obtained by evaluating $\langle (c_0, c_1), (1, s) \rangle = m + e$, which is equivalent to an evaluation of the ciphertext polynomial at the point s.

Messages are vectors $\boldsymbol{v} \in \mathbb{C}^n$, for n a power of two smaller than N. Before being encrypted, messages are subjected to the transformation $R_Q \leftrightarrow \mathbb{C}^n$: $m = \lfloor \Delta\tau(\boldsymbol{v}) \rceil$, for τ a canonical embedding and Δ a scaling factor. This encoding preserves the group operations $+$ and $\times$ such that additions and multiplications in R translate to point-wise additions and multiplications in $\mathbb{C}^n$. With this encoding the CKKS scheme is notably able to homomorphically evaluate polynomials over vectors of complex numbers.

A message Δm is encrypted at the modulus Q (maximum *level*) and each subsequent multiplication *consumes* a *level* and reduces the size of the modulus Q. Indeed, when two ciphertext encrypting the messages $\Delta\tau(\boldsymbol{v}_0)$ and $\Delta\tau(\boldsymbol{v}_1)$ are multiplied together, the resulting encrypted message is $\Delta^2\tau(\boldsymbol{v}_0)\tau(\boldsymbol{v}_1)$. To avoid an exponential growth of the scaling factor, the ciphertext is divided by Δ after each multiplication, reducing Q by that amount. This operation is called *rescaling*. Hence, the upper bound on Q fixes the maximum homomorphic capacity of fresh ciphertexts (the maximum circuit's depth). Once a ciphertext reaches its smallest possible modulus q and no further rescaling is possible, it can be *bootstrapped* back to a larger modulus, thus enabling the evaluation of arbitrary depth circuits.

1.2 Bootstrapping

The bootstrapping procedure for the CKKS scheme was first proposed by Cheon et al. [8] and can be summarized in four steps: (i) ModRaise: raise the ciphertext, currently at its smallest modulus q, back to its highest modulus Q. (ii) CoeffsToSlots: homomorphically evaluate the canonical embedding τ. (iii) EvalMod: homomorphically evaluate a modular reduction approximated by the scaled sine function $q/(2\pi) \cdot \sin(2\pi x/q)$. (iv) SlotsToCoeffs: homomorphically evaluate τ^{-1}. The procedure outputs a ciphertext at modulus Q' with $Q > Q' > q$, the difference between Q' and q being the *residual* homomorphic capacity after the bootstrapping. Cheon et al. evaluate τ with a matrix (plaintext) $\times$ vector (encrypted) multiplication and compute the scaled sine function using the Taylor series of e^{ix} followed by an extraction of the imaginary part to retrieve $\sin(x)$.

Extensive works have since improved the efficiency of the original procedure of Cheon et al. The first improvement was proposed by Chen et al. [4]. In their work they improved the efficiency of the homomorphic evaluation of τ by multiple order of magnitude by adopting an FFT-like approach instead of a single matrix-vector multiplication. They also proposed a more efficient polynomial approximation by directly approximating $\sin(x)$ with a Chebyshev interpolant.

In a concurrent work Cheon et al. [6] proposed a similar technique to improve the evaluation of τ.

These works were followed by researches aimed at improving the homomorphic modular reduction, which is the most difficult step of the bootstrapping since the CKKS scheme does not support the evaluation of non-polynomial functions. Han and Ki [13] proposed a polynomial interpolation that takes into account the distribution of the message and instead uses a scaled cosine to enable the double angle formula, which allowed them to greatly reduce the degree of the interpolant. Lee et al. [19] proposed a modified multi-interval Remez algorithm to find the optimal minimax approximation of the scaled sine/cosine functions, as well as using the inverse sine function to remove the error introduced by the approximation of the ideal modular function by a trigonometric function. Lee et al. [20] proposed a polynomial interpolation that minimises the variance of the interpolant, thus reducing the error introduced by the homomorphic evaluation of the polynomial. Jutla and Manohar [15] proposed a novel variant of the Lagrange interpolation, called modular Lagrange interpolation, that allows them to directly approximate the modular reduction function, without having to rely on trigonometric functions. They also proposed in [16] to use a sine series to approximate the modular reduction and achieved a much higher precision than the previous works. Lee et al. [21] proposed a polynomial approximation method for the modular reduction based on the L2-norm minimization. Similarly to Jutla and Manohar, their technique allows them to avoid using trigonometric functions and directly approximate the modular reduction.

Bossuat et al.[3] proposed a more efficient algorithm to evaluate general linear transformations and polynomial evaluation algorithm that preserves the ciphertext scale and does not introduce rescaling errors, as well as several other smaller improvements to the bootstrapping procedure. They show that when combined together, these improvements lead to a bootstrapping procedure an order of magnitude more efficient than the previous works. Additionally they proposed the first practical instance of a bootstrapping with a dense secret as well as the first open source implementation[4] of the bootstrapping for the full-RNS variant of the CKKS scheme [7]. Finally, Yu and Hayato [14] proposed a more efficient way to evaluate the trace function.

Put together, these works improved the bootstrapping procedure to be orders of magnitude more efficient and precise than the original proposal by Cheon et al. However, the bootstrapping circuit has fundamentally remained the same since its first introduction. One of its limitations is its high sensitivity to the density h of the secret s: the larger h is (the more non-zero elements s has), the more complicated the $\mathsf{EvalMod}$ step is and the more depth the bootstrapping requires. The density h also has an impact on the bootstrapping failure probability. Indeed, the magnitude of the plaintext coefficient on which the homomorphic modular reduction must be applied is a function of h, and if a single coefficient falls outside of the approximation interval (a ciphertext typically encrypts 2^{14} to 2^{16} values), the bootstrapping procedure fails and returns unusable

[4] `https://github.com/ldsec/lattigo`

values. Bossuat et al. [3] observed that commonly used bootstrapping parameters have a high failure probability and it is only recently that works started to quantify and address this failure probability.

For these reasons, bootstrapping procedures are instantiated with sparse secrets (with small h). But recent improvements on attacks targeting sparse secrets [9, 24, 11] have reduced the upper bound on the modulus Q, consequently, parameters using sparse keys must regularly be updated. This situation has, for the moment, lead the standardization initiatives to exclude sparse secrets from the currently proposed standards [1]. Being able to mitigate this dependency on sparse secrets would therefore be an important step for the adoption and practicality of CKKS bootstrapping.

1.3 Our Contributions

In this work, we propose a *sparse-secret encapsulation* technique for the CKKS bootstrapping; the technique takes advantage of the security margin provided by having evaluation keys at a small modulus to improve the CKKS bootstrapping security and efficiency. Our main contributions can be summarized as follows:

Minimized Security-Dependency on Sparse Secrets. The *leveled* property of the CKKS scheme is tightly related to its security, as the security of an R-LWE sample is notably based on the size of its modulus Q. For a fixed ring degree N and a security parameter λ, an upper bound for Q is derived and the public keys are generated using this Q. Although R-LWE samples at modulus Q have security λ, previous works do not take into account that elements at a lower *level* have a proportionally smaller modulus, hence a larger security.

We propose a modification to the bootstrapping circuit that enables the generation of all evaluation keys using a secret that is independent from the one on which the complexity of the $\mathsf{EvalMod}$ step is based. Instead of the usual single secret instance, our bootstrapping instance uses an additional ephemeral secret that determines the complexity of the bootstrapping procedure. As such, the maximum modulus Q does not depend anymore on the sparse secret that defines the complexity of the $\mathsf{EvalMod}$ and a denser secret can be used to generate the evaluation keys.

This construction has a two-fold benefit: (i) It enables to more freely choose bootstrapping parameters, such as ones with a denser secret and larger homomorphic capacity, as the security of all evaluation keys at the maximum modulus is based on a different secret than the one defining the circuit complexity, therefore achieving a better security-performance trade-off. (ii) The dependency on the sparse secret is minimized by limiting it to an evaluation key at the smallest modulus; we will show that this evaluation key also has a large security margin against attacks that target sparse (and dense) secrets.

Negligible Failure Probability. By having the $\mathsf{EvalMod}$ step rely on a ephemeral sparse secret of low h we are able to greatly reduce the complexity of this step, allowing for a higher precision and a much lower failure probability. In fact, we can easily make this failure probability smaller than the security parameter $2^{-\lambda}$, which has never been achieved before to the best of our knowledge.

Empirical Experiments and Open Source Implementation. We evaluate our contribution with empirical experiments and provide an open source implementation of this work.

The remaining of the article is organized as follow: In **Section 2**, we introduce the used notation and recall the necessary background for the CKKS scheme and its bootstrapping; In **Section 3**, we present our core contributions; In **Section 4**, we give a security argument that our modification does not introduce any new security assumption and examine the security of our construction for concrete parameters; In **Section 5**, we empirically analyse the impact of our contribution on the noise and bootstrapping precision. In **Section 6**, we evaluate our contribution with empirical experiments and discuss the implications of our modified bootstrapping procedure.

2 Background

In this section, we introduce the notation used in the remaining of this paper, as well as the necessary technical background relating to our contribution.

2.1 Notation

For a fixed power of two N, let $R_Q = \mathbb{Z}_Q[X]/(X^N + 1)$ be the cyclotomic polynomial ring over the integers modulo Q with coefficients in $[-\lfloor Q/2 \rfloor, \lfloor Q/2 \rfloor)$. Define $Y = X^{N/2n}$ for some power of two n smaller than N (a polynomial in Y is a polynomial in X with zero at coefficient degree that are not a multiple of $N/2n$). We denote single elements (polynomials or numbers) in italics, e.g., a, and vectors of such elements in bold, e.g., $\boldsymbol{a}$. We denote $a^{(i)}$ the element at position i of the vector $\boldsymbol{a}$ or the degree-i coefficient of the polynomial a. We denote $||\cdot||$ the infinity norm, $[\cdot]_Q$, $\lfloor \cdot \rfloor$, $\lfloor \cdot \rceil$ the reduction modulo Q, rounding to the previous and to the closest integer, respectively (coefficient-wise for polynomials), and $\langle \cdot, \cdot \rangle$ the dot product.

We define the following distributions over R: χ_Q with coefficients uniformly distributed over $\mathbb{Z}_Q$. χ_h with coefficients uniformly distributed over $\{-1, 0, 1\}$ and exactly h non-zero coefficients. χ_σ with coefficients distributed according to a discrete Gaussian distribution with standard deviation σ. Unless otherwise specified, σ is assumed to be 3.19 (Homomorphic Encryption Standard [1]) and truncated to $[-\lfloor 6\sigma \rfloor, \lfloor 6\sigma \rfloor]$. $\leftarrow \chi$ is the act of sampling a polynomial with coefficients from the given distribution.

For a given $s \leftarrow \chi_h$, the tuple $\{N, Q, h, \sigma\}$ parameterizes the R-LWE distribution that is sampled as $(-as + e, a) \in R_Q^2$ with $a \leftarrow \chi_Q$ and $e \leftarrow \chi_\sigma$. We say that a parameter set $\{N, Q, h, \sigma\}$ with s is λ-secure if the advantage of an adversary $\mathcal{A}$ to distinguish between the distribution $(-as + e, a) \in R_Q^2$ and the uniform distribution $U(R_Q^2)$ is bounded by $2^{-\lambda}$:

$$\mathrm{Adv}_{\mathcal{A}} = |\Pr[\mathcal{A}^{(-as+e,a)} = 1] - \Pr[\mathcal{A}^{U(R_Q^2)} = 1]| \leq 2^{-\lambda}.$$

2.2 Approximate Homomorphic Encryption (CKKS)

A CKKS plaintext is a polynomial $m(Y) \in R[Y]/(Y^{2n} + 1)$ (with $X^{N/2n} = Y$). We define the following plaintext encoding:

- The *coefficient* encoding, for which the message $\boldsymbol{m} \in \mathbb{R}^{2n}$, is directly encoded on $R[Y]/(Y^{2n} + 1)$ as $m(Y) = \lfloor \varDelta \boldsymbol{m} \rceil$.
- The *slot* encoding, for which the message $\boldsymbol{m} \in \mathbb{C}^n$ is subjected to the canonical embedding $\tau : \mathbb{C}^n \rightarrow Y^{2n}$ which preserves the coefficient-wise complex arithmetic in $\mathbb{R}[Y]/(Y^{2n} + 1)$. The *coefficient* encoding is then applied to encode the result on $R[Y]/(Y^{2n} + 1)$.

A CKKS ciphertext ct_Q^s is an R-LWE sample $(c_0, c_1) = (-as+m+e, a) \in R_Q^2$ for m a plaintext polynomial, and the decryption circuit is its evaluation at the secret-key s: $\langle (c_0, c_1), (1, s) \rangle = m + e \in R$. Message polynomials $m(Y)$ do not live in R_Q but in R, as such their norm must at all time, including during homomorphic computations, be smaller than Q to avoid a modular reduction.

A CKKS switching key $\mathsf{swk}_{QP}^{s \rightarrow s'}$ is a vector of R-LWE samples $(-a^{(i)}s' + w^{(i)}Ps + e^{(i)}, a^{(i)}) \in R_{QP}^{2\times\beta}$ for $1 \leq i \leq \beta$, $\boldsymbol{w} = (w^1, \ldots, w^\beta)$ an integer basis decomposition and P a secondary modulus such that $P \approx \sum w^{(i)}$. Note that the security of switching keys is based on the modulus QP. Using the public algorithm $\mathsf{KeySwitch}$, $\mathsf{swk}_{QP}^{s\rightarrow s'}$ can be used to homomorphically re-encrypt a ciphertext $\mathsf{ct}^s = (c_0, c_1)$ to a ciphertext $\mathsf{ct}^{s'} = (c_0', c_1')$ by computing $(c_0', c_1') = (c_0, 0) + \lfloor P^{-1} \cdot \langle \boldsymbol{w}^{-1}(c_1), \mathsf{swk}^{s\rightarrow s'} \rangle \rceil$. The additional modulus P is used to control the magnitude of the error (which is $\langle \boldsymbol{w}^{-1}(c_1), \boldsymbol{e} \rangle$) added during the key-switching. The public encryption key is a switching key $\mathsf{swk}_{QP}^{0\rightarrow s}$. In addition to the access structure management that this procedure provides, it is a fundamental building block of the CKKS scheme as it is used to ensure the correctness and compactness of the decryption circuit for several core homomorphic operations (e.g. ciphertext-ciphertext multiplication and homomorphic plaintext-slots cyclic-rotations).

2.3 Bootstrapping

The bootstrapping procedure of the CKKS scheme [8] aims at raising the ciphertext to a higher modulus to enable further homomorphic evaluation. More specifically, upon the input of a ciphertext ct_q^s such that $\langle \mathsf{ct}_q^s, (1, s) \rangle = m(Y) + e$, for s a secret with h non-zero coefficients, the CKKS bootstrapping outputs a ciphertext $\mathsf{ct}_{Q'}^s$ that decrypts to $m'(Y) = m(Y) + e'$, where $Q > Q' > q$ for Q the maximum modulus for a given λ, Q' the modulus after the bootstrapping and q the modulus before the bootstrapping. It is important to note that $||e'|| \geq ||e||$. This implies that, although this procedure is referred to as *bootstrapping*, it only approximates an ideal bootstrapping procedure. The procedure consists of the following four steps: ModRaise, CoeffsToSlots, EvalMod, and SlotsToCoeffs. We now briefly explain them, omitting the error terms for clarity.

ModRaise: the ciphertext at modulus q is expressed in the modulus $Q \gg q$. This yields a ciphertext that decrypts to $[c_0 + sc_1]_Q = q \cdot I(X) + m(Y) = m'(Y)$, where $q \cdot I(X) = \big[- [sc_1]_q + sc_1 \big]_Q$ is an integer polynomial and $||I(X)|| \leq h$. Note that this step does not modify the coefficient of the ciphertext (thus has no effect on the error), as it only represent them in a different RNS basis.

If $2n \neq N$ (*sparse* packing), then $Y \neq X$ and $I(X)$ is not a polynomial in Y. In other words, we have multiples of q in the coefficients X that are not multiples of $N/2n$. In this case, we can map $q \cdot I(X) + m(Y)$ to $(N/2n) \cdot (q \cdot \tilde{I}(Y) + m(Y))$ by evaluating a trace-like map [8] that makes coefficients of X whose degree is not a multiple of $N/2n$ vanish and multiplies the others by $N/2n$. This map can be efficiently evaluated with $\log(N/2n)$ key-switching [8].

The remaining steps of the bootstrapping remove this unwanted $q \cdot \tilde{I}(Y)$ polynomial by homomorphically evaluating a modular reduction by q on $m'(Y)$.

CoeffsToSlots: the canonical embedding τ is homomorphically evaluated on $m'(Y)$. Indeed, $m'(Y)$ can be seen as a fresh message in the *coefficient* domain. To enable the parallel (slot-wise) evaluation, it needs to be encoded in the *slot* domain.

EvalMod: a polynomial approximation of the modular reduction by q is homomorphically evaluated on $m'(Y)$, thus removing the unwanted $\tilde{I}(Y)$ polynomial.

SlotsToCoeffs: the inverse of the canonical embedding, τ^{-1}, is evaluated on $m'(Y)$ and a close approximation of original message $m(Y)$ minus the unwanted polynomial is retrieved. After this last step the ciphertext has modulus $Q' > q$ and we can evaluate further operations, until it reaches modulus q and a new bootstrapping is needed.

3 Proposed Technique

Our contribution is based on two observations: (i) The complexity of the EvalMod step is determined by the secret distribution of the ciphertext during the ModRaise step (we further specify this dependency in Section 3.1). (ii) The *leveled* behavior of the CKKS scheme positively affects its security. In other words, ciphertexts entering the ModRaise procedure are at a low *level*, and a sparser secret can be used for the same security.

We use theses observations to modify the ModRaise step of the bootstrapping by encapsulating it between two KeySwitching procedures: the first one is to switch the low-*level* ciphertext to a sparser secret $\tilde{s}$ before the ModRaise and second one, after the ModRaise, is to switch the high-*level* ciphertext back to a dense secret s.

Here, we detail the original ModRaise procedure and the improvement we bring to it.

3.1 Original ModRaise and Bootstrapping Failure Probability

The original ModRaise (see Section 2.3) takes a ciphertext $\mathsf{ct} = (c_0, c_1) \in R_q^2$ that decrypts to $m(Y)$, a polynomial of $2n$ coefficients, and outputs a new ciphertext $\mathsf{ct}' = (c_0', c_1') \in R_Q^2$ that decrypts to a new message of the form $q \cdot \tilde{I}(Y) + m(Y)$.

The norm of the polynomial $\tilde{I}(Y)$ is upper-bounded by the Hamming weight h of the secret, hence the EvalMod step has to evaluate a polynomial approximation of the modular reduction in the interval $[-h, h]$. However, this upper bound h can be quite large; since $\tilde{I}(Y)$ follows an Irwin-Hall distribution [19], we have that $||\tilde{I}(Y)||$ is $\mathcal{O}(\sqrt{h})$ with high probability [8] and, in practice, a smaller probabilistic bound $K < h$ is used instead. Given that the ciphertext encrypts a message $m(Y)$ with $Y = X^{N/2n}$ under a secret $s \leftarrow \chi_h$ before the ModRaise, the exact probability $f(K, h, n) = \Pr[||\tilde{I}(Y)|| > K]$ can be computed by adapting the cumulative probability function of the Irwin-Hall distribution [3]:

$$1 - \left(\frac{2}{(h+1)!} \left(\sum_{i=0}^{\lfloor K+0.5(h+1) \rfloor} (-1)^i \binom{h+1}{i} (K + 0.5(h+1) - i)^{h+1} \right) - 1 \right)^{2n} . \tag{1}$$

We refer to $f(K, h, n)$ as the *bootstrapping failure probability*. For example, if we upper bound the failure probability to $f(K, h, n) \leq 2^{-15}$ for a fixed $n = 2^{15}$ slots and variable h, then $\lim_{h \to \infty} K \approx 1.81\sqrt{h}$ [3].

Therefore, the density h of the secret has a two-fold effect on the practicality of the bootstrapping. On one hand, the sparser the secret, the smaller the range of parameters that can securely and efficiently evaluate the bootstrapping circuit, as a smaller h implies a smaller upper-bound on the modulus Q for a fixed ring degree N and a security parameter λ. On the other hand, the denser the secret, the more *levels* are required for the EvalMod step. Indeed, this step is to homomorphically evaluate a modular reduction on the interval $[-K, K]$ that, as shown, is proportional to $\sqrt{h}$.

3.2 ModRaise with *Sparse-Secret Encapsulation*

For a given security parameters λ, we propose to instantiate the base scheme, as well as its bootstrapping circuit, with a secret s of density h such that the R-LWE samples, of the keys, under s and at modulus QP are at least λ-secure and to encapsulate the ModRaise step between two KeySwitch such that the ciphertext is only temporarily switched to a sparser secret $\tilde{s}$ with density $\tilde{h} < h$ during the ModRaise, with R-LWE samples under $\tilde{s}$ and at modulus $qp \ll QP$ being at least λ-secure. Consequently, the unwanted polynomial $q \cdot \tilde{I}(Y)$ depends on the distribution of $\tilde{s}$, but the bootstrapping circuit remains evaluated under s.

To do this instantiation, we generate two sets of parameters $\{N, QP, h, \sigma\}$ and $\{N, qp, \tilde{h}, \sigma\}$, with $qp \ll QP$ and $\tilde{h} < h$, which are both at least λ-secure; and we sample a secret $s \leftarrow \chi_h$, as well as a secret $\tilde{s} \leftarrow \chi_{\tilde{h}}$. Let $\mathsf{swk}_{qp}^{s \to \tilde{s}}$ be a switching key at modulus qp, which can be used to publicly re-encrypt a ciphertext from the secret s to the secret $\tilde{s}$. And let $\mathsf{swk}_{QP}^{\tilde{s} \to s}$ be a switching key at modulus QP, which can be used to publicly re-encrypt a ciphertext from $\tilde{s}$ to s.

Given a ciphertext ct_q^s at modulus q encrypted under s, our proposed algorithm first key-switches ct_q^s from s to $\tilde{s}$ using $\mathsf{swk}_{qp}^{s \to \tilde{s}}$. Then, it applies the regular $\mathsf{ModRaise}$ algorithm that expresses its coefficients in a larger modulus. The ciphertext is now expressed in the modulus Q, but with coefficients whose norm remains unchanged and bounded by $\lfloor q/2 \rceil$. Finally, the algorithm key-switches it back to the key s by using $\mathsf{swk}_{QP}^{\tilde{s} \to s}$. We detail our modified $\mathsf{ModRaise}$ in Algorithm 1.

Algorithm 1: Encapsulated ModRaise

```
  Input: $\mathsf{ct}_q^s$, $\mathsf{swk}_{qp}^{s \to \tilde{s}}$, $\mathsf{swk}_{QP}^{\tilde{s} \to s}$
  Output: $\mathsf{ct}_Q^s$
1 $\mathsf{ct}_q^{\tilde{s}} \leftarrow \mathsf{KeySwitch}(\mathsf{ct}_q^s, \mathsf{swk}_{qp}^{s \to \tilde{s}})$
2 $\mathsf{ct}_Q^{\tilde{s}} \leftarrow \mathsf{ModRaise}(\mathsf{ct}_q^{\tilde{s}}, Q)$
3 $\mathsf{ct}_Q^s \leftarrow \mathsf{KeySwitch}(\mathsf{ct}_Q^{\tilde{s}}, \mathsf{swk}_{QP}^{\tilde{s} \to s})$
4 return $\mathsf{ct}_Q^s$
```

Remark 1. Algorithm 1 is implementation-agnostic therefore compatible with both the original [10] and the full-RNS variants of the CKKS scheme proposed by Cheon et al. [7]. If the implementation of the $\mathsf{KeySwitch}$ begins with a modulus basis extension (for example, from Q to QP), Algorithm 1 can be optimized by merging the $\mathsf{ModRaise}$ step in the second $\mathsf{KeySwitch}$ (now from q to QP), such that it essentially becomes two consecutive key-switches.

3.3 Impact on the Evaluation-Key Generation

Our modification to the bootstrapping slightly changes how evaluation keys are generated, as we now need to generate two sets of evaluation keys instead of one:

1. A set parameterized by $\{N, QP, h, \sigma\}$ that uses a key s and comprises the encryption key, all the necessary evaluation keys for the linear transformations and homomorphic modular reduction, as well as the switching key $\mathsf{swk}_{QP}^{\tilde{s} \to s}$.
2. A set parameterized by $\{N, qp, \tilde{h}, \sigma\}$, with $qp \ll QP$ and $\tilde{h} < h$, and that uses uses a secret $\tilde{s}$ and comprises the switching key $\mathsf{swk}_{qp}^{s \to \tilde{s}}$.

Although we increase the number of evaluation keys by two, this is only marginal with respect to the total number of switching keys needed for the bootstrapping, as it is largely dominated by the number rotations keys needed for the linear transformations, which is in the order of a hundred for $n = 2^{15}$ slots.

Our construction allows to use a dense secret for s and to instantiate all the evaluation keys at a larger modulus, which will inevitably increases their size. We however stress that the increase in size of the key material is an normal behavior of the scheme as it is directly related to the homomorphic capacity of a parameter set.

In Section 4, we show that the modification to the $\mathsf{ModRaise}$ algorithm and the addition of the switching keys $\mathsf{swk}_{QP}^{\tilde{s}\rightarrow s}$ and $\mathsf{swk}_{qp}^{s\rightarrow \tilde{s}}$ do not introduce new security assumptions and that our construction is secure.

4 Security Analysis

In this section, we provide a security argument for Algorithm 1 (Section 3.2) that shows our modification does not change nor introduces new security assumptions to the CKKS scheme or its original bootstrapping. We then discuss the benefit on the security of using a small ephemeral secret during the bootstrapping and estimate security of such ephemeral secrets for concrete parameters.

Note that, regardless of our proposition, users should always be aware of the security implications of using the CKKS scheme [22] as well as sparse secrets, and that they should carefully choose how it is parameterized.

4.1 Modified $\mathsf{ModRaise}$ Security

We consider an adversary $\mathcal{A}$ who has access to the public transcript of Algorithm 1:

- ct_q^s, an R-LWE sample $(-as+m+e, a) \in R_q^2$ with m a message, and security parameterized by the tuple $\{N, q, h, \sigma\}$.

- $\mathsf{swk}_{qp}^{s\rightarrow \tilde{s}}$, a switching-key composed of a set of R-LWE samples $(-a^{(i)}\tilde{s} + w^{(i)}ps + e^{(i)}, a^{(i)}) \in R_{qp}^{2\times\beta}$ with $a^{(i)} \leftarrow R_{qp}$, $\tilde{s} \leftarrow \chi_{\tilde{h}}$, $e^{(i)} \leftarrow \chi_\sigma$ and $\boldsymbol{w} = (w^1, \ldots, w^\beta)$ a decomposition basis. The security of this set of R-LWE samples is parameterized by the tuple $\{N, qp, \tilde{h}, \sigma\}$.

- $\mathsf{swk}_{QP}^{\tilde{s}\rightarrow s}$, a switching-key composed of a set of R-LWE samples $(-a^{(i)}s + w^{(i)}P\tilde{s} + e^{(i)}, a^{(i)}) \in R_{QP}^{2\times\beta}$ with $a^{(i)} \leftarrow R_{QP}$, $s \leftarrow \chi_h$, $e^{(i)} \leftarrow \chi_\sigma$ and $\boldsymbol{w} = (w^1, \ldots, w^\beta)$ a decomposition basis. The security of this set of R-LWE samples is parameterized by the tuple $\{N, QP, h, \sigma\}$.

and wins if it can distinguish $(\mathsf{ct}_q^s, \mathsf{swk}_{qp}^{s\to\tilde{s}}, \mathsf{swk}_{QP}^{\tilde{s}\to s})$ from the uniform distribution $U(R_q^2, R_{qp}^{2\times\beta}, R_{QP}^{2\times\beta})$ with an advantage greater than $2^{-\lambda}$. Therefore, to ensure that

$$\mathrm{Adv}_{\mathcal{A}} = |\Pr[\mathcal{A}^{(\mathsf{ct}_q^s, \mathsf{swk}_{qp}^{s\to\tilde{s}}, \mathsf{swk}_{QP}^{\tilde{s}\to s})} = 1] - \Pr[\mathcal{A}^{(R_q^2, R_{qp}^{2\times\beta}, R_{QP}^{2\times\beta})} = 1]| \leq 2^{-\lambda},$$

it suffices to select the parameter sets $\{N, qp, \tilde{h}, \sigma\}$ and $\{N, QP, h, \sigma\}$ to be at least λ-secure ($\{N, q, h, \sigma\}$ is naturally at least λ-secure if $\{N, QP, h, \sigma\}$ is itself λ-secure since $q \ll QP$). Regarding their joint distribution, the security argument holds under the assumption of circular security, which is presupposed to be able to generate evaluation keys. For a parameterization example, we take two sets of parameters from the work of Cheon et al. [5]: $\{2^{15}, 2^{881}, 2^{14}, 3.2\}$ and $\{2^{15}, 2^{431}, 2^{6}, 3.2\}$; both are $\lambda = 128$-bit secure. Note that, in practice, the second set of parameters $\{N, qp, \tilde{h}, \sigma\}$ has a much smaller qp (e.g. 120 bits) than the 431 bits used in this example, because it is instantiated at the smallest possible modulus. Hence, this parameter set is actually more secure than the one that uses the dense key (see Table 1).

Table 1. Parameters' security for the low-*level* switching key. The modulus of the switching key is composed of q and an additional modulus p used during the key-switching. W denotes log(keyspace size), i.e., $\log\left(\binom{N}{\tilde{h}} \cdot 2^{\tilde{h}}\right)$. * (the asterix) indicates that the estimator failed to provide a result and instead the security was extrapolated.

$\log N$	$\log qp$	$\tilde{h}$	W	Primal[2]	Dual[2]	Dec[2]	Hybrid-Primal[5]	Hybrid-Dual[5]
16	60+61	64	792	368.0	340.4	376.0	260.4	317.8
		32	427	222.7	192.5	226.6	168.5*	283.8
15	55+56	64	728	309.2	415.0	315.6	217.7	227.8
		32	395	187.9	191.5	319.0	140.9	162.2

4.2 Minimizing the use of Sparse Secret and Higher Security

Previous works on the CKKS bootstrapping assumed predefined single sparse-secret parameters and were focused on improving its efficiency [8, 4, 6, 13, 14, 19, 20, 16, 15, 21, 5]. The use of a sparse secrets was deemed necessary to make the bootstrapping sufficiently practical. Although Bossuat et al. [3] showed that using a single dense secret can also be practical, this comes at the cost of a reduced efficiency and precision (they need to evaluate a polynomial of several hundred coefficients).

Our work changes this paradigm by, instead, proposing a higher-level change that directly removes this constraint. Although simple, our *sparse-secret encapsulation* brings a significant improvement to the security and practicality of the CKKS bootstrapping. It enables the user to instantiate the bootstrapping evaluation keys with a dense secret, which brings more freedom in making choices about the parameters and isolates the security assumption related to the sparse-secret to a single low-*level* (small modulus) key. Being at a low *level*, this key benefits from a large security margin against the most recent attacks [9, 24] and will be, in practice, more secure than the evaluation keys generated with the dense secret. This result is an important step towards a more practical and secure CKKS bootstrapping. Table 1 provides parameterization examples and their security for the low-*level* switching key that uses a sparse secret.

5 Empirical Noise Analysis

In this section we will quantify the impact of our modification on the noise of the bootstrapping procedure. Our modification impacts the noise in two dimensions: it slightly modifies the circuit by adding additional key-switching operations and it allows the use of denser keys.

The initial noise of the CKKS scheme is well understood and numerous works have proposed noise analysis [10, 8, 20, 17]. These works, however, keep their analysis to single operations because when operations are composed the estimation of the noise (either as a bound or an average case) becomes less and less meaningful since the initial exact noise is unknown.

For this reason, noise analysis, especially for complicated circuits such as the bootstrapping, remain heuristic and with loose bounds. This is especially true when other factors than the initial noise have to be taken into account such as polynomial approximations or plaintext distribution. As such, noise analysis for circuits are in practice conducted with experiments.

In this section we empirically demonstrate with experiments the following propositions:

Proposition 1. *The modification to the* ModRaise *step only adds a small additive noise which has a negligible impact on the bootstrapping precision.*

Proposition 2. *Then noise terms which are a function of the density h of the secret quickly dominate the additive noise of the bootstrapping circuit.*

5.1 Proposition 1

Our modification to the ModRaise step adds two key-switching operations, one before and one after. The noise introduced by the key-switching is additive and can be made close to the error added by a decryption if correctly parameterized [13, 20, 17].

The ModRaise step is followed by the CoeffsToSlots step, which homomorphically evaluates the encoding algorithm. This step is carried out by evaluating a linear transformation [8] on the ciphertext vector and for n slots requires $O(\sqrt{n})$ plaintext multiplications and $O(\sqrt{r}\log_r(n))$ rotations [12] (which are key-switching operations), for a radix $r \leq n$.

Hence the two additional key-switching happening during the ModRaise step should only have a small, up to negligible, impact on the overall additive bootstrapping error. We verify Proposition 1 with the following empirical experiment:

we compare the error of the reference circuit of Bossuat et al. [3] and the same circuit where only the ModRaise step differs. We use the exact same parameters as the one used by Bossuat et al. for all parameter sets, as well as the same secret key density. For our modified circuit, the ModRaise step switches the secret to a different one of the same density h.

Table 2 reports the results of the experiment and we observe that there is no significant difference between the noise of the original bootstrapping of Bossuat et al. and our modified circuit. The largest differences are coming from the sets IV and V but remain small (0.09 to 0.25 bits of difference). Both can be attributed to parameters that lead to a bootstrapping instance that is more sensitive to the distribution of the initial noise (larger secret density) and/or the additive noise (smaller plaintext scale). We will see in Section 6.1 that the reduction in the EvalMod complexity from a smaller $\tilde{h}$ step allows to entirely mitigate this small loss of precision and even increase the bootstrapping precision.

Table 2. Impact of the modified ModRaise on the bootstrapping precision, by comparison between the results of Bossuat et al. [3] and the same bootstrapping circuit but with the modified ModRaise. Our work uses identical parameters to the ones of Bossuat et al. for all sets and our modified ModRaise switches the secret to a different secret of same density h. N is the ring degree, $\log QP$ the modulus of the switching keys, h the density of the secret, n the number of plaintext slots, K the probabilistic upper bound of $||\tilde{I}(Y)||$, $d_{\sin(x)}$ the degree of the scaled cosine interpolant (Han and Ki's method [13]), r the number of double angle evaluation, $d_{\arcsin(x)}$ the degree of the arcsine interpolant (Taylor series) and $\log \epsilon^{-1}$ the negative log of the error, which is interpreted as the plaintext precision.

Set [3]	$\log N$	$\log QP$	h	$\log n$	K	$d_{\sin(x)}$	r	$d_{\arcsin(x)}$	$\log \epsilon^{-1}$	
									[3]	Ours
I	16	1546	192	15	25	63	2	0	25.70	25.71
				14					26.00	26.07
II	16	1547	192	15	25	63	2	7	31.50	31.51
				14					31.60	31.68
III	16	1553	192	15	25	63	2	0	19.10	19.09
				14					18.90	18.92
IV	16	1792	32768	15	325	255	4	0	16.80	16.65
				14					17.30	17.21
V	15	768	192	14	25	63	2	0	15.50	15.15
				13					15.40	15.29

5.2 Proposition 2

The error of individual homomorphic operations of the CKKS scheme has been studied and is well understood [10, 8, 20, 17]. Notably, the error of a decrypted and decoded message in the CKKS scheme is a function of $\sqrt{h}$, h being the number of non zero elements of the secret key. Although the error related to the secret distribution can be controlled and minimized for most operations with a careful parameterization and scale management (such as addition, plaintext multiplication or key-switching), ciphertext multiplication amplifies the error at a much greater rate because their error terms are compounded. This is specifically the case for polynomial evaluation, which involves ciphertext exponentiation when computing the power basis.

We empirically verify this statement with the following experiment: we compare the precision of the bootstrapping circuit and its different individual parts for an increasing main secret density h. For the full bootstrapping circuit we use our modified ModRaise step with an ephemeral secret with $\tilde{h} = 32$ ($\lambda \approx 168$ for $N = 2^{16}$ and $\log(qp) = 121$, see Table 1 in Section 4). The results of this experiment are in Figure 1.

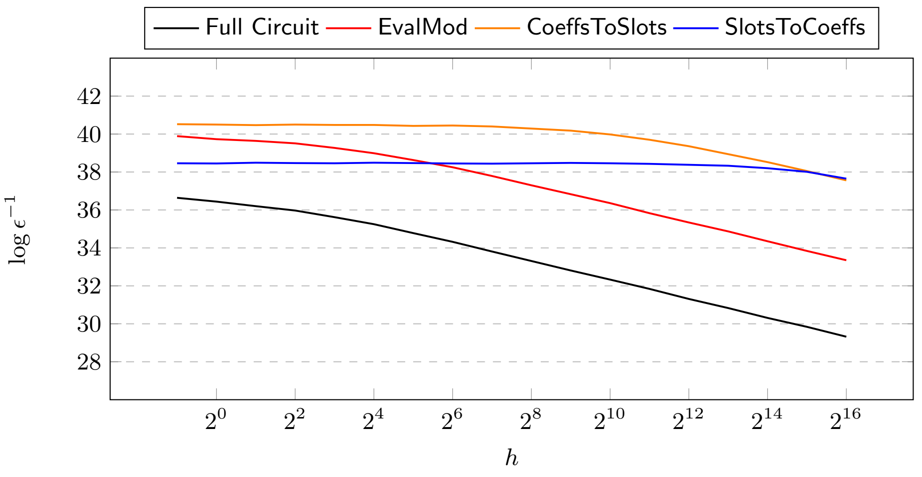

Fig. 1. Precision of the full bootstrapping circuit and its different individual parts for a secret s with variable h. The full circuit uses our modified ModRaise with an ephemeral secret $\tilde{s}$ with $\tilde{h} = 32$. The parameters are $\log N = 2^{16}$ and $\log n = 2^{15}$, an initial scale of 2^{52} and 60-bit moduli (the maximum allowed size) are used for all operations. The EvalMod parameters are $K = 10$, $d_{\sin(x)} = 30$, $r = 3$ and $d_{\arcsin(x)} = 7$. $\log \epsilon^{-1}$ is the negative log of the error, which is interpreted as the precision.

We observe that for operations involving a controlled noise augmentation (CoeffsToSlots and SlotsToCoeffs can be summarized as sums of plaintext multiplications and key-switching operations) the initial encryption noise (which includes the encoding error of both the plaintext vector and plaintext matrices) is much larger than the noise added by the homomorphic operations and the decryption process. This results in a constant precision, until the noise terms related to h become dominants, which happens at around $h = 2^8$ for CoeffsToSlots and $h = 2^{14}$ for SlotsToCoeffs. At this point, the line starts to follow the $\sqrt{h}$ relationship (doubling h induces a loss of precision of 0.5 bits).

As expected the EvalMod step, which is an operation involving ciphertext exponentiation, has a noise growth that quickly overcomes the initial noise with a steady $\sqrt{h}$ relationship that already starts at $h = 2^4$. We observed that increasing the degree of the interpolant actually reduced the precision instead of increasing it. Meaning that the expected gain in precision from the additional higher degree terms of the interpolant was actually overcome by the error resulting from higher exponentiation to compute the additional terms of the power basis.

The precision of the full circuit follows the one of the EvalMod step with a stable offset of about 4 bits, confirming that the EvalMod step is the bottleneck of the bootstrapping circuit precision. This is not surprising since the EvalMod step is the only non-linear part of the bootstrapping circuit. This offset is the result of the composition of the different part of the bootstrapping circuit and compounding of their errors.

In practice the loss of precision caused by a higher secret density can be compensated by increasing the initial scale Δ if needed. However when using the full-RNS variant of the CKKS scheme [7] one cannot simply arbitrarily increase this scale since the primes used have a size that is limited by machine words, which are usually of 64 bits. In practice the maximum size of the primes is even smaller, typically of 61 bits, to enable more efficient implementations. A solution is to use multiple words per prime or multiple primes per *level*, but both will induce an overhead.

5.3 Conclusion

In this section we empirically showed that: (i) our modification to the ModRaise step has a negligible impact on the bootstrapping precision and that (ii) the noise term related to the density h of the secret quickly dominates all other terms when ciphertext multiplication is involved. Our experiments allow to conclude that our construction by itself only has a negligible impact the bootstrapping precision and that an increased noise, when using a main secret with a higher density h comes from the inherent noise of the scheme and not our modification of the ModRaise step. These results are further confirmed with the experimental results shown in Section 6.

6 Evaluation

In this section we evaluate the performance of our proposed modification against the recent work of Bossuat et al. [3], which is currently the state of art in term of *bootstrapping throughput* (number of plaintext bits bootstrapped per second). We will show that, when using the same parameters, our construction enables a more efficient and precise bootstrapping with a much lower failure probability.

We implemented our work in the Lattigo library[5] [18]. All benchmarks were conducted on hardware with the same specifications as the one used by Bossuat

[5] `https://github.com/ldsec/lattigo/tree/btp_eprint`

et al. (Windows 10, i5-6600K CPU @ 3.50GHz, 32 GB of RAM, single threaded), ensuring meaningful comparisons.

All parameter sets, for all experiments, have a security $\lambda \approx 128$.

6.1 Better Precision, Reduced Failure Probability and Smaller Interpolant

The EvalMod step of the bootstrapping procedure evaluates a polynomial approximation of the modular reduction. The *bootstrapping failure probability* is given by the function $f(K, h, n)$ (see Equation 1 in Section 3.1), with $[-K, K]$ the range of the approximation, h the density of the secret at the moment of the ModUp step and n the number of plaintext slots. Indeed, if a coefficient falls outside of the bound K, the polynomial approximation fails and therefore the bootstrapping procedure fails as well. Hence, the precision of the polynomial approximation evaluated during the EvalMod is a trade off between the degree of the approximation, which has an impact on the number of levels consumed during this step, and the range of the approximation, which, for a given approximation degree, determines both the precision and failure probability of the EvalMod step. So, for a fixed h, n and approximation degree, the greater K is, the smaller the failure probability, but the smaller the precision. Therefore, if h can be reduced, K can be reduced and the precision increased (up to the inherent precision of the interpolant).

Table 3 compares the precision and failure probability of the bootstrapping from the original work of Bossuat et al. [3] with ours. Complementary information about the used interpolant can be found in Table 4.

We observe that, for all sets, the bootstrapping precision is either similar or improved, showing that we are able to achieve a failure probability that is many orders of magnitude smaller without compromising the security or precision. The largest improvement is for Set IV, which is not surprising since Bossuat et al. had to use a very large interpolant for their EvalMod step.

The configuration of Bossuat et al. lead to a failure probability of $2^{-31.6}$ per plaintext slot ($2^{-15.6}$ for $n = 2^{15}$ slots). Our failure probability per slot is now $2^{-50.1}$ ($2^{-34.11}$ for $n = 2^{15}$ slots) for $K = 12$ and $2^{-154.7}$ ($2^{-138.7}$ for $n = 2^{15}$ slots) for $K = 16$, which is smaller than the security parameter. Note that when using Han and Ki's interpolation method, the interpolant has a minimum degree of $d_{\sin(x)} = 2(K-1)$, hence $K = 16$ is the maximum value to get a depth $\log(d+1) \leq 5$ interpolant.

For all parameter sets except for Set IV, Bossuat et al. used $K = 25$, $d_{\sin(x)} = 63$ (the degree of the scaled sine interpolant) and $r = 2$ (the number of double angle evaluation), for a total depth of $6 + 2 = 8$. By reducing K to 12 and 16, we were originally able to reduce the interpolant degree to $d_{\sin(x)} \approx 40$ for an equivalent bootstrapping precision. In this configuration, it turns out that $d_{\sin(x)}$ is now small enough to be able to increase r to 3 and further reduce $d_{\sin(x)}$ to a value equal or smaller to 31. This allows us to keep the same depth and precision, but with a more efficient polynomial evaluation since each double angle evaluation only needs one multiplication.

Table 3. Bootstrapping precision and failure probability when reducing the complexity of the EvalMod step. The original results of of Bossuat et al. [3] are given for reference and we use the same cryptographic parameter sets for all experiments (which are identical to the ones in Table 2). n is the number of plaintext slots, h the density of the main secret, $\tilde{h}$ the density of the ephemeral secret, K the range for the approximation of the scaled sine function, $f(K, h, n)$ the failure probability function, and $\log \epsilon^{-1}$ the negative log of the error, which is interpreted as the plaintext precision. Details about the interpolant used for each set can be found in Table 4. The security of the ephemeral secret is, for $\tilde{h} = 32$, $\lambda \approx 168$ for the Sets I to IV, and $\lambda \approx 141$ for Set V (see Table 1 in Section 4).

Set [3]	$\log n$	h	Bossuat et al. [3]			This work				This work			
			K	$f(K,h,n)$	$\log \epsilon^{-1}$	$\tilde{h}$	K	$f(K,\tilde{h},n)$	$\log \epsilon^{-1}$	$\tilde{h}$	K	$f(K,\tilde{h},n)$	$\log \epsilon^{-1}$
I	15	192	25	-15.58	25.70	32	12	-34.11	27.32	32	16	-138.70	26.63
	14			-16.58	26.00			-35.11	27.39			-139.70	26.89
II	15	192	25	-15.58	31.50	32	12	-34.11	32.36	32	16	-138.70	32.11
	14			-16.58	31.60			-35.11	32.17			-139.70	32.04
III	15	192	25	-15.58	19.10	32	12	-34.11	19.14	32	16	-138.70	19.13
	14			-16.58	18.95			-35.11	18.92			-139.70	18.90
IV	15	32768	325	-14.90	16.80	32	12	-34.11	23.80	32	16	-138.70	23.12
	14			-15.90	17.30			-35.11	24.29			-139.70	23.62
V	14	192	25	-16.58	15.50	32	12	-34.11	15.48	32	16	-139.70	15.45
	13			-17.58	15.40			-35.11	15.66			-140.70	15.55

In conclusion, Table 3 shows that, when using the parameters of Bossuat et al. in conjunction with our modified ModRaise step and a small ephemeral secret, the bootstrapping precision and failure probability can be noticeably improved.

6.2 Higher Bootstrapping Throughput

The *bootstrapping utility* is a metric that enables the evaluation of the performance of a bootstrapping circuit. It is a concept that was first introduced by Chen et al. [4] as $n \times$ Levels/Time, for n the number of plaintext slots, Levels the number of *levels* available after the bootstrapping and Time the bootstrapping complexity represented in CPU time (single threaded). It was then expanded to the *bootstrapping throughput* by Bossuat et al. [3], which measures the number of plaintext bits bootstrapped per second as $n \times \log \epsilon^{-1} \times \log Q'/\text{Time}$, for $\log \epsilon^{-1}$ the bootstrapping precision and $\log Q'$ the number of modulus bits available after the bootstrapping. Bossuat et al. use $\log Q'$ instead of the number of remaining *levels* because this value is more representative of the actual remaining homomorphic capacity. Indeed, optimizing a homomorphic circuit often leads to a dynamic scale, in which case the notion of level does not make sense anymore (e.g. one can artificially increase or decrease the number of levels available by changing the ciphertext scale).

Table 5 reports the *bootstrapping throughput* of the experiments of Table 3 (along with Table 4). This comparison between the results of Bossuat et al. and

Table 4. Complementary table detailing the interpolants used for the EvalMod step of the experiments of Table 3. K is the range of the interpolation, $d_{\sin(x)}$ the degree of the scaled cosine interpolant (Han and Ki's method [13]), r the number of double angle evaluations and $d_{\arcsin(x)}$ the degree of the arcsine interpolant (Taylor series).

Set [3]	Bossuat et al. [3]				This work				This work			
	K	$d_{\sin(x)}$	r	$d_{\arcsin(x)}$	K	$d_{\sin(x)}$	r	$d_{\arcsin(x)}$	K	$d_{\sin(x)}$	r	$d_{\arcsin(x)}$
I	25	63	2	0	12	22	3	0	16	30	3	0
II	25	63	2	7	12	24	3	7	16	30	3	7
III	25	63	2	0	12	22	3	0	16	30	3	0
IV	325	255	4	0	12	22	3	0	16	44	2	0
V	25	63	2	0	12	22	3	0	16	30	3	0

ours shows that our modification allows for better timings, even though two additional key-switching operations are added to the ModUp step.

Table 5. Comparison of the *bootstrapping throughput* [3] with $\log(\text{bits}/s) = \log(n \times \log Q' \times \log \epsilon^{-1}/\text{Time})$, where n is the number of plaintext slots, Q' the residual modulus after the bootstrapping, $\log \epsilon^{-1}$ the bootstrapping precision and Time the CPU cost in seconds.

Set [3]	$\log n$	Bossuat et al. [3]				This work				Ratio
		$\log \epsilon^{-1}$	$\log Q'$	Time	log bits/s	$\log \epsilon^{-1}$	$\log Q'$	Time	log bits/s	bits/s
I	15	25.7	420	23.0	23.87	26.63	420	19.9	24.13	1.19×
	14	26.0	420	16.9	23.33	26.89	420	14.9	23.56	1.17×
II	15	31.5	285	23.4	23.59	32.11	285	20.2	23.82	1.17×
	14	31.6	285	16.0	23.13	32.04	285	14.5	23.30	1.12×
III	15	19.1	505	18.1	24.06	19.13	505	15.9	24.24	1.13×
	14	18.9	505	13.1	23.50	18.90	505	11.9	23.64	1.10×
IV	15	16.8	410	39.2	22.70	23.12	420	19.9	23.93	2.34×
	14	17.3	410	24.9	22.15	23.62	420	14.9	23.37	2.33×
V	14	15.5	110	7.5	21.82	15.45	110	5.9	22.17	1.27×
	13	15.4	110	6.0	21.14	15.55	110	4.5	21.57	1.34×

We observe that all parameter sets of our work have a larger *bootstrapping throughput*, with Set IV achieving a throughput that is 2.34× the one of Bossuat et al. This shows that our proposed change to the ModUp steps enables a better *bootstrapping throughput* in addition to a negligible failure probability (note that this failure probability is not taken into account in the *bootstrapping throughput*).

6.3 Dense Key Bootstrapping

In the previous sections we evaluated the performance of our modified bootstrapping against the results of Bossuat et al., for which the parameters use a sparse secret as the main secret. In this section we evaluate the performance of our modified bootstrapping with parameters that use a dense secret as the main secret.

By increasing the density of the main secret from 192 to $N/2$, we are able to increase $\log QP$ to ≈ 1790 for $N = 2^{16}$ and ≈ 881 for $N = 2^{15}$, thus increasing the remaining homomorphic capacity ($\log Q'$) after the bootstrapping, and still retaining a security of $\lambda \approx 128$ bits.

Table 6. *Bootstrapping throughput* [3] of various parameter sets with a $\log QP$ based on a dense secret as the main secret, with $\log(\text{bits}/s) = \log(n \times \log Q' \times \log \epsilon^{-1}/\text{Time})$, where n is the number of plaintext slots, Q' the residual modulus after the bootstrapping, $\log \epsilon^{-1}$ the bootstrapping precision and Time the CPU cost in seconds.

$\log N$	$\log QP$	$(h, \tilde{h})$	$\log n$	$\log \epsilon^{-1}$	$\log Q'$	Time	log bits/s
16	$1401 + 366$	$(N/2, 32)$	15	23.0	580	25.1	24.05
16	$1483 + 305$	$(N/2, 32)$	15	29.8	465	26.3	24.04
16	$1488 + 305$	$(N/2, 32)$	15	17.8	745	21.5	24.26
15	$768 + 112$	$(N/2, 32)$	14	17.3	166	7.9	22.50

Table 6 reports the result of this experiment and shows that despite the expected and unavoidable loss of precision of $0.5 \cdot \log((N/2)/192) \approx 3.7$ for $N = 2^{16}$ and ≈ 3.2 for $N = 2^{15}$ due to a higher-density secret (see Section 5), we are still able to obtain a similar if not greater *bootstrapping throughput* than when using a sparse secret as the main secret. Timings are slightly larger than the ones reported in Table 5 because all operations are happening at a higher modulus, thus are more costly. The parameter set for $N = 2^{15}$ shows a significant larger *bootstrapping throughput* compared to Set V of both Bossuat et al. and our work (1.6× and 1.25× respectively). The reason is that this parameter set could only accommodate for a small homomorphic capacity when using a sparse secret, and the bootstrapping precision had to be deliberately tuned down to end up with meaningful remaining homomorphic capacity after the bootstrapping. Being able to increase the homomorphic capacity also allowed us to allocate larger moduli to the bootstrapping circuit, thus increasing its precision.

Although the *bootstrapping throughput* reported in Table 6 is only slightly larger than the one shown in Table 5 (with the exception of the parameter set using $N = 2^{15}$, for which it is significantly larger), the parameters used in Table 6 would likely not need to be updated even if attacks on sparse secrets were improved, because these would not apply to the main secret (which is dense), and the low-level sparse secret benefits from a large security margin.

7 Conclusion

In this work, we have presented a *sparse-secret encapsulation* technique for the bootstrapping of the CKKS scheme. We have shown that by temporarily switching during the ModRaise step the low-*level* ciphertext to a sparser secret, we can optimize the efficiency-security trade-off of the bootstrapping circuit, by breaking the dependency between the sparse-secret security and the largest modulus. This enables all high-*level* evaluation keys to use a denser secret, thus to provide a greater initial homomorphic capacity and more resilience to attacks targeting sparse secrets, while still enjoying a low-complexity bootstrapping. Moreover our technique also enables the parameterization of the EvalMod step in an interval that is large enough to make its failure probability arbitrarily small, which, to the best of our knowledge, has never been achieved before.

When using the parameters of previous works, our experiments show that the proposed modification allows for a 128-bit secure bootstrapping with negligible failure probability that also benefits from a greater remaining homomorphic capacity, greater precision, and smaller complexity. Moreover, when using a dense secret, our bootstrapping circuit has greater *bootstrapping throughput* than previous state-of-the-art approaches that use a sparse secret, especially for small parameters.

We believe that these improvements are a major step forward for the security, stability, efficiency and reliability of the bootstrapping of the CKKS scheme, which is a necessary building block to enable practical high-depth arithmetic circuit evaluation under encryption.

8 Acknowledgments

We would like to thank Christian Mouchet for his valuable feedback. This work was supported in part by the grant #2017-201 of the ETH Domain PHRT Strategic Focal Area.

References

[1] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. *Homomorphic Encryption Security Standard*. Tech. rep. Toronto, Canada: HomomorphicEncryption.org, Nov. 2018.

[2] Martin R. Albrecht, Rachel Player, and Sam Scott. "On the concrete hardness of Learning with Errors". In: *Journal of Mathematical Cryptology* 9 (3 2015), pp. 169–203.

[3] Jean-Philippe Bossuat, Christian Mouchet, Juan Troncoso-Pastoriza, and Jean-Pierre Hubaux. "Efficient Bootstrapping for Approximate Homomorphic Encryption with Non-sparse Keys". In: *Advances in Cryptology – EUROCRYPT 2021*. Ed. by Anne Canteaut and François-Xavier Standaert. Cham: Springer International Publishing, 2021, pp. 587–617. ISBN: 978-3-030-77870-5.

[4] Hao Chen, Ilaria Chillotti, and Yongsoo Song. "Improved bootstrapping for approximate homomorphic encryption". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2019, pp. 34–54.

[5] J. Cheon, Yongha Son, and Donggeon Yhee. "Practical FHE parameters against lattice attacks". In: *IACR Cryptol. ePrint Arch.* 2021 (2021), p. 39.

[6] Jung Hee Cheon, Kyoohyung Han, and Minki Hhan. "Faster Homomorphic Discrete Fourier Transforms and Improved FHE Bootstrapping". In: *IACR Cryptol. ePrint Arch.* 2018 (2018), p. 1073.

[7] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. "A full RNS variant of approximate homomorphic encryption". In: *International Conference on Selected Areas in Cryptography*. Springer. 2018, pp. 347–368.

[8] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. "Bootstrapping for approximate homomorphic encryption". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2018, pp. 360–384.

[9] Jung Hee Cheon, Minki Hhan, Seungwan Hong, and Yongha Son. "A hybrid of dual and meet-in-the-middle attack on sparse and ternary secret LWE". In: *IEEE Access* 7 (2019), pp. 89497–89506.

[10] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. "Homomorphic encryption for arithmetic of approximate numbers". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2017, pp. 409–437.

[11] Thomas Espitau, Antoine Joux, and Natalia Kharchenko. "On a Dual/Hybrid Approach to Small Secret LWE". In: *Progress in Cryptology – INDOCRYPT 2020*. Ed. by Karthikeyan Bhargavan, Elisabeth Oswald, and Manoj Prabhakaran. Cham: Springer International Publishing, 2020, pp. 440–462. ISBN: 978-3-030-65277-7.

[12] Kyoohyung Han, Minki Hhan, and Jung Hee Cheon. "Improved Homomorphic Discrete Fourier Transforms and FHE Bootstrapping". In: *IEEE Access* 7 (2019), pp. 57361–57370. DOI: `10.1109/ACCESS.2019.2913850`.

[13] Kyoohyung Han and Dohyeong Ki. "Better bootstrapping for approximate homomorphic encryption". In: *Cryptographers' Track at the RSA Conference*. Springer. 2020, pp. 364–390.

[14] Yu Ishimaki and Hayato Yamana. "Faster Homomorphic Trace-Type Function Evaluation". In: *IEEE Access* 9 (2021), pp. 53061–53077. DOI: `10.1109/ACCESS.2021.3071264`.

[15] Charanjit S. Jutla and Nathan Manohar. *Modular Lagrange Interpolation of the Mod Function for Bootstrapping of Approximate HE*. Cryptology ePrint Archive, Report 2020/1355. `https://eprint.iacr.org/2020/1355`. 2021.

[16] Charanjit S. Jutla and Nathan Manohar. *Sine Series Approximation of the Mod Function for Bootstrapping of Approximate HE*. Cryptology ePrint Archive, Report 2021/572. `https://eprint.iacr.org/2021/572`. 2021.

[17] Andrey Kim, Antonis Papadimitriou, and Yuriy Polyakov. *Approximate Homomorphic Encryption with Reduced Approximation Error*. Cryptology ePrint Archive, Report 2020/1118. `https://ia.cr/2020/1118`. 2020.

[18] *Lattigo 2.4.0*. Online: `https://github.com/ldsec/lattigo`. EPFL-LDS. Jan. 2022.

[19] Joon-Woo Lee, Eunsang Lee, Yongwoo Lee, Young-Sik Kim, and Jong-Seon No. *High-Precision Bootstrapping of RNS-CKKS Homomorphic Encryption Using Optimal Minimax Polynomial Approximation and Inverse Sine Function*. Cryptology ePrint Archive, Report 2020/552. `https://eprint.iacr.org/2020/552`. 2020. Accepted to Eurocrypt 2021.

[20] Yongwoo Lee, Joon-Woo Lee, Young-Sik Kim, HyungChul Kang, and Jong-Seon No. *High-Precision and Low-Complexity Approximate Homomorphic Encryption by Error Variance Minimization*. Cryptology ePrint Archive, Report 2020/1549. `https://eprint.iacr.org/2020/1549`. 2020.

[21] Yongwoo Lee, Joon-Woo Lee, Young-Sik Kim, and Jong-Seon No. "Near-Optimal Polynomial for Modulus Reduction Using L2-Norm for Approximate Homomorphic Encryption". In: *IEEE Access* 8 (2020), pp. 144321–144330. DOI: `10.1109/ACCESS.2020.3014369`.

[22] Baiyu Li and Daniele Micciancio. "On the Security of Homomorphic Encryption on Approximate Numbers". In: Springer-Verlag, 2021.

[23] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On Ideal Lattices and Learning with Errors over Rings". In: *Advances in Cryptology – EUROCRYPT 2010*. Ed. by Henri Gilbert. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 1–23. ISBN: 978-3-642-13190-5.

[24] Yongha Son and Jung Hee Cheon. "Revisiting the Hybrid attack on sparse and ternary secret LWE". In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 1019.