When HEAAN Meets FV: a New Somewhat Homomorphic Encryption with Reduced Memory Overhead

Hao Chen[1], Ilia Iliashenko[2], and Kim Laine[3]

[1] Facebook, Cambridge, USA
`sxxach@gmail.com`
[2] imec-COSIC, Dept. Electrical Engineering, KU Leuven, Belgium
`ilia@esat.kuleuven.be`
[3] Microsoft Research, Redmond, USA
`kim.laine@microsoft.com`

Abstract. We demonstrate how to reduce the memory overhead of somewhat homomorphic encryption (SHE) while computing on numerical data. We design a hybrid SHE scheme that exploits the packing algorithm of the `HEAAN` scheme and the variant of the `FV` scheme by Bootland et al. The ciphertext size of the resulting scheme is 3-18 times smaller than in `HEAAN` to compute polynomial functions of depth 4 while packing a small number of data values. Furthermore, our scheme has smaller ciphertexts even with larger packing capacities (256-2048 values).

1 Introduction

Homomorphic encryption (HE) [21] is a family of encryption schemes that allow computation on encrypted messages without decryption. Several types of such schemes have been proposed in the last 40 years, including *partially* homomorphic encryption (e.g. [22,15,11,19]), which can perform either addition or multiplication, *somewhat* homomorphic encryption (SHE), which supports functions of a limited multiplicative depth, and *fully* homomorphic encryption (FHE) [14] capable to compute any function on encrypted data.

Despite their universality, SHE/FHE schemes have a significant disadvantage in practice. In particular, they introduce a huge memory overhead per encrypted bit value, which makes even simple arithmetic operations on numerical data impractically slow. To mitigate this overhead various encoding algorithms have been proposed that exploit the structure of typical plaintext spaces used in SHE/FHE [7,10,6,2,4,3]. They deviate from bit-wise encryption to so-called 'word-wise' encryption where one (or even several) data values can be encrypted per ciphertext. Unfortunately, these algorithms perform correctly only if the ciphertext modulus grows exponentially with the depth of the circuit.

A more efficient approach was proposed by Cheon et al. [5], who introduced a new type of HE schemes, called *approximate* HE (AHE). The crucial idea is to allow an additional error while performing homomorphic operations. For

example, multiplication of two encrypted plaintexts $\mathtt{ct_1} = \mathtt{Encrypt}(\mathtt{pt_1})$ and $\mathtt{ct_2} = \mathtt{Encrypt}(\mathtt{pt_2})$ results in another ciphertext $\mathtt{ct_3}$, which is decrypted to $\mathtt{Decrypt}(\mathtt{ct_3}) = \mathtt{pt_1} \cdot \mathtt{pt_2} + e$ where e is a so-called 'noise'. The size of e defines the approximation 'closeness' between decrypted and expected results.

AHE is suitable for handling non-integer data types such as real, rational or complex numbers. Computation on such numerical types in computer systems is inherently prone to numerical errors. Thus, the results of such computation are only correct up to a certain precision. Approximation errors introduced by AHE can be treated as a part of these numerical errors.

The drawback of the first AHE scheme from [5], called $\mathtt{HEAAN}$, is that its ciphertext size should be set quite large to be able to compute even simple polynomial functions with a decent precision. This problem is mitigated by packing, which is an encoding technique that allows to encrypt several data values into one ciphertext. In addition, computations on these packed values can be performed in the Single-Instruction Multiple-Data (SIMD) manner.

The packing method of $\mathtt{HEAAN}$ permits thousands of data values to be encrypted into a single ciphertext, thus significantly reducing the amortized ciphertext expansion per data value. However, various applications do not require such a large packing capacity and assume modest computational resources, especially in the use cases where embedded devices are used [12,16,18].

For example, such a device can collect data about vital organs (e.g. heart, blood pressure) and constantly sends it to a special service that runs a private prediction model of a heart attack in a privacy-preserving manner. Each message from the device contains a smaller amount of values (dozens, maybe a hundred of measurements) in comparison to the dimension of plaintexts in FHE schemes (thousands). In addition, each message should be as small as possible to fit constraints of the device, but at the same time it should be large enough to let the service perform homomorphic computations of certain depth.

To reduce ciphertext size, one can resort to so-called 'high-precision' SHE schemes [4,3] that trade packing capacity for additional homomorphic operations. Thus, relatively deep computations can be performed with modest encryption parameters of these schemes. The known high-precision SHE schemes are variants the $\mathtt{FV}$ scheme [13]. $\mathtt{FV}$ is not AHE, but its encryption function is very similar to the one of $\mathtt{HEAAN}$. Furthermore, the variant of $\mathtt{FV}$ by Bootland et al. [3] supports complex-number arithmetic as $\mathtt{HEAAN}$.

Our contribution. In this work, we design a new SHE scheme that can perform computations on numerical data with smaller ciphertexts than in $\mathtt{HEAAN}$.

The core idea is to exploit the recent variant of the $\mathtt{FV}$ scheme [13] due to Bootland et al. [3], where the integer plaintext modulus is replaced by a polynomial $X^m + b$ for some m and b. The plaintext space of this scheme is $\mathbb{Z}[X]/(X^n + 1, X^m + b)$, which is isomorphic to the ring of cyclotomic integers $\mathbb{Z}[\zeta_{2m}]/(b^{n/m} + 1)$ if n and m are powers of 2 and b is an mth power modulo $b^{n/m} + 1$. This $\mathtt{FV}$ variant natively supports homomorphic computation on large cyclotomic integers $\mathbb{Z}[\zeta_{2m}]$ with small encryption parameters.

2

We combine this scheme with the HEAAN packing algorithm, which maps complex-valued vectors into cyclotomic integers. More precisely, it encodes elements of $\mathbb{C}^{m/2}$ into the aforementioned ring $\mathbb{Z}[\zeta_{2m}]$, which can be easily embedded into $\mathbb{Z}[\zeta_{2m}]/(b^{n/m}+1)$, the plaintext space of Bootland's variant of FV.

As a result we obtain a hybrid SHE scheme which follows the following diagram

$$\mathbb{C}^{m/2} \xrightarrow{\text{HEAAN packing}} \mathbb{Z}[\zeta_{2m}]/(b^{n/m}+1) \xrightarrow{\text{FV encryption}} R_q^2 = (\mathbb{Z}[X]/(X^n+1))^2,$$

where R_q^2 is the ciphertext space. This hybrid leverages the advantages of both schemes: the small memory overhead of Bootland's FV variant and the large packing capacity of HEAAN. Furthermore, since our technique only changes the way how the plaintext space of Bootland's scheme is used, the security analysis of our hybrid scheme is exactly the same as for the FV scheme.

We describe a family of arithmetic circuits where our scheme have a smaller memory and running time overhead than HEAAN. In addition, we illustrate the difference between these schemes by computing several important analytic functions.

2   Preliminaries

For any $a \in \mathbb{N}$, we denote the set of integers $\{1, \ldots, a\}$ by $[a]$. Vectors and matrices are denoted by boldface lower- and upper-case letters, respectively. Vectors are written in column form.

Let n be a positive power of 2. Let K be a cyclotomic number field constructed by adjoining a primitive complex $2n$-th root of unity to the field of rational numbers. We denote this root of unity by ζ_{2n}, so $K = \mathbb{Q}(\zeta_{2n})$. The ring of integers of K, denoted by R, is isomorphic to $\mathbb{Z}[X]/(X^n+1)$.

For any $a \in K$ its coefficient vector $(a_0, \ldots, a_{n-1})$ in the power basis is denoted by $\mathbf{a}$. The infinity norm of a is equal to $|a|_\infty = |\mathbf{a}|_\infty = \max_{i=0}^{n-1} |a_i|$. The product of any $a, b \in K$ satisfies the following bound $|ab|_\infty \leq n \cdot |a|_\infty \cdot |b|_\infty$, see [9] for more details.

Let R_a be the quotient of R modulo an ideal (a). If a is a natural number, we take representatives of $\mathbb{Z}/a\mathbb{Z}$ from the half-open interval $[-a/2, a/2)$.

The semantic security of encryption schemes presented in this paper is based on the RLWE problem introduced in [17].

Definition 1 (RLWE problem). *Let $q > 2$ be an integer. Let $s \in R_q$ be a random secret element, $a, a', b' \in R_q$ be uniformly random elements and $e \in R_q$ be a random element sampled from some known distribution over R_q. The RLWE problem is to distinguish between $(a, b = as + e)$ and (a', b').*

The hardness of the RLWE problem implies that the above pair (a, b) is pseudorandom. Thus, it can be exploited as a random mask in encryption and key generation (see more details in [17]).

3 `HEAAN` packing method

In this section, we describe the `HEAAN` method for packing complex-valued vectors as presented in [5]. This packing method exploits the canonical embedding of cyclotomic fields.

Let $K' = \mathbb{Q}\left(\zeta_{2m}\right)$, where m is a power of two and m divides n. Clearly, K' is a subfield of K. We denote the ring of integers of K' by R'. Since $[K' : \mathbb{Q}] = m$, there exist m field homomorphisms $\sigma_i' : K' \to \mathbb{C}$ with $i \in [m]$ that fix every element of $\mathbb{Q}$. In particular, each σ_i' is a complex embedding that maps ζ_{2m} to ζ_{2m}^{2i-1}. These are the only field homomorphisms from K' to $\mathbb{C}$.

Let H' be a vector subspace of $\mathbb{C}^m$ such that $H' = \{(x_1, \ldots, x_m)^{\mathsf{T}} : x_{m-j+1} = \hat{x}_j, \forall j \in [m/2]\}$. This space is equipped with a projection map $\pi : H' \to \mathbb{C}^{m/2}$ that discards either of complex conjugate components. Conversely, the inverse map π^{-1} appends a vector from $\mathbb{C}^{m/2}$ with the conjugates of its coordinates in the order compliant with H'.

The *canonical embedding* of K' is the map $\sigma' : K' \to H'$ defined as $\sigma'(a) = (\sigma_1'(a), \ldots, \sigma_m'(a))$. By analogy, we can define the canonical embedding of K denoted by σ, which endows K with the canonical norm via $\|a\|^{\mathrm{can}} = |\sigma(a)|_\infty$ for any $a \in K$. Since n is a power of two, $|a|_\infty \leq \|a\|^{\mathrm{can}}$ as shown in [9]. In addition, $\|ab\|^{\mathrm{can}} \leq \|a\|^{\mathrm{can}} \|b\|^{\mathrm{can}}$ for any $a, b \in K$.

Let $\mathbf{a} = (a_0, a_1, \ldots, a_{m-1})$ be the coefficient vector of $a \in K'$ in the power basis of K'. Then, the canonical embedding σ' transforms $\mathbf{a}$ into

$$\mathbf{\Sigma} \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_{m-1} \end{pmatrix} = \begin{pmatrix} \sigma_1'(a) \\ \vdots \\ \sigma_m'(a) \end{pmatrix}$$

where $\mathbf{\Sigma} = (\zeta_{2m}^{j(2i+1)})_{i,j}$ is a Vandermonde matrix. Since $\mathbf{\Sigma}$ is nonsingular, the inverse of the canonical embedding is correctly defined by $\mathbf{\Sigma}^{-1} = \left(\frac{1}{m}\zeta_{2m}^{-i(2j+1)}\right)_{i,j}$.

Thus, the composition map $\sigma'^{-1} \circ \pi^{-1}$ encodes vectors from $\mathbb{C}^{m/2}$ into K'.

To finish packing, elements of K' should end up in the ring of cyclotomic integers R'. This can be done using discretization to the lattice $\sigma'(R')$, which boils down to coefficient-wise rounding with relation to the power basis of R' over $\mathbb{Z}$. However, this rounding introduces an error that might damage significant bits of input values. To eliminate this error, an input vector is scaled up by some value Δ. To summarize, the complete packing pipeline consists of the following map chain

- `Pack`$(\Delta) : \mathbb{C}^{m/2} \xrightarrow{\cdot\Delta} \mathbb{C}^{m/2} \xrightarrow{\pi^{-1}} H' \xrightarrow{\sigma'^{-1}} K' \xrightarrow{\lfloor\cdot\rceil} R'.$

The unpacking algorithm is the inverse map of `Pack` without the rounding step, namely

- `Unpack`$(\Delta') : R' \xrightarrow{\sigma'} H' \xrightarrow{\pi} \mathbb{C}^{m/2} \xrightarrow{\cdot\Delta'^{-1}} \mathbb{C}^{m/2}.$

The size of Δ is defined by the input precision p and the input dimension m according to the following lemma.

Lemma 1. *Given an input vector $\mathbf{z} \in \mathbb{C}^{m/2}$ and a positive integer p, the vector $\mathbf{z}' = \mathtt{Unpack}(\Delta, \mathtt{Pack}(\Delta, \mathbf{z}))$ satisfies $|\mathbf{z} - \mathbf{z}'|_\infty < \frac{1}{p}$, if $\Delta > \frac{pm}{2}$.*

Proof. Let $\mathbf{u} \in H'$ be the output of the first two steps of the $\mathtt{Pack}$ algorithm, namely $\mathbf{u} = \pi^{-1}(\Delta \cdot \mathbf{z})$. Then, the final output $a \in R'$ can be represented in matrix notation as

$$\mathbf{a} = \left\lfloor \mathbf{\Sigma}^{-1} \cdot \mathbf{u} \right\rceil = \mathbf{\Sigma}^{-1} \cdot \mathbf{u} + \mathbf{e}$$

with $|\mathbf{e}|_\infty \leq 1/2$. Computing $\mathtt{Unpack}(\Delta, a)$, we obtain

$$\mathbf{z}' = \frac{1}{\Delta} \pi \left(\mathbf{\Sigma} \mathbf{a} \right) = \frac{1}{\Delta} \pi (\mathbf{u} + \mathbf{\Sigma} \mathbf{e}) = \mathbf{z} + \frac{1}{\Delta} \pi(\mathbf{\Sigma} \mathbf{e}) \,.$$

Hence, the difference between the input $\mathbf{z}$ and its packed approximation $\mathbf{z}'$ satisfies the following bound

$$|\mathbf{z} - \mathbf{z}'|_\infty = \max_{i \in [m/2]} \left| \frac{1}{\Delta} \sum_{j=0}^{m-1} e_j \cdot \zeta_{2m}^{j(2i-1)} \right| \leq \frac{m}{2\Delta} \,,$$

which immediately leads to the desired lower bound on Δ.

4   FV scheme with a polynomial plaintext modulus

In this section we describe the variant of the $\mathtt{FV}$ scheme given by Bootland et al. in [3], which is based on the work of Chen et al. [4]. The main difference of this variant from the original $\mathtt{FV}$ scheme [13] consists in switching from an integer plaintext modulus t to a polynomial $X^m + b$.

Let q be an integer. The ciphertext space is defined as $R_q = R/(q)$. Take an integer b such that $2 \leq |b| \ll q$. Let m be a positive integer dividing n. The quotient ring $R_{X^m+b} = R/(X^m + b)$ serves as the plaintext space. We define the encryption scaling factor Δ_b as follows

$$\Delta_b = \left\lfloor \frac{q}{X^m + b} \bmod (X^n + 1) \right\rceil = \left\lfloor -\frac{q}{b^{n/m} + 1} \sum_{i=1}^{n/m} (-b)^{i-1} \cdot X^{n-im} \right\rceil \,.$$

Let χ_e be the error distribution on R, which is a coefficient-wise discrete Gaussian distribution with respect to the power basis. The standard deviation of χ_e is σ. The key distribution χ_k generates uniformly random elements of R with ternary coefficients (again, with respect to the power basis). We also set an integer $w > 1$ and call it the decomposition base. Let $\ell = \lfloor \log_w q \rfloor$.

Given this set-up, the basic $\mathtt{FV}$ scheme with polynomial plaintext modulus is defined as follows.

- KeyGen(1^n): Let $s \leftarrow \chi_k$ and $e, e_0, \ldots e_\ell \leftarrow \chi_e$. Generate uniformly random $a, a_0, \ldots, a_\ell \in R_q$ and compute $b_i = \left[-(a_i \cdot s + e_i) + w^i \cdot s^2\right]_q$. Output
  - the secret key $\mathtt{sk} = s$,
  - the public key $\mathtt{pk} = \left([-(a \cdot s + e)]_q, a\right)$,
  - the evaluation key $\mathtt{rlk} = \{(b_i, a_i)\}_{i=0}^{\ell}$.
- Encrypt($\mathtt{pk}, \mathtt{msg} \in R_{X^m + b}$): Sample $u \leftarrow \chi_k$ and $e_0, e_1 \leftarrow \chi_e$. Set $p_0 = \mathtt{pk}[0]$ and $p_1 = \mathtt{pk}[1]$. Output $\mathtt{ct} = (c_0, c_1)$, where

$$c_0 = [\Delta_b \cdot \mathtt{msg} + p_0 \cdot u + e_0]_q , \qquad c_1 = [p_1 \cdot u + e_1]_q$$

- Decrypt($\mathtt{sk}, \mathtt{ct}$): Return

$$\mathtt{msg}' = \left\lfloor \frac{X^m + b}{q} [c_0 + c_1 \cdot s]_q \right\rceil \mod (X^m + b).$$

As shown in [17], this encryption scheme is semantically secure assuming the hardness of the RLWE problem.

4.1 Homomorphic operations

It is easy to adapt the homomorphic operations of FV to the new plaintext modulus as shown below.

- Add($\mathtt{ct}_0, \mathtt{ct}_1$): Return $\mathtt{ct}_{\mathtt{Add}} = \left([\mathtt{ct}_0[0] + \mathtt{ct}_1[0]]_q, [\mathtt{ct}_0[1] + \mathtt{ct}_1[1]]_q\right)$.
- BasicMul($\mathtt{ct}_0, \mathtt{ct}_1$): Return $\mathtt{ct}_{\mathtt{BasicMul}} = (c_0, c_1, c_2)$, where

$$c_0 = \left[\left\lfloor \left\lfloor \frac{X^m + b}{q} \cdot \mathtt{ct}_0[0] \cdot \mathtt{ct}_1[0] \right\rceil \right\rfloor\right]_q, \quad c_2 = \left[\left\lfloor \left\lfloor \frac{X^m + b}{q} \cdot \mathtt{ct}_0[1] \cdot \mathtt{ct}_1[1] \right\rceil \right\rfloor\right]_q,$$

$$c_1 = \left[\left\lfloor \left\lfloor \frac{X^m + b}{q} \cdot (\mathtt{ct}_0[0] \cdot \mathtt{ct}_1[1] + \mathtt{ct}_0[1] \cdot \mathtt{ct}_1[0]) \right\rceil \right\rfloor\right]_q.$$

- Relin($\mathtt{ct}_{\mathtt{BasicMul}}, \mathtt{rlk}$): Writing $\mathtt{ct}_{\mathtt{BasicMul}} = (c_0, c_1, c_2)$, expand c_2 in base w such that $c_2 = \sum_{i=0}^{\ell} c_{2,i} \cdot w^i$ with $|c_{2,i}|_\infty \leq w/2$. Compute

$$c_0' = c_0 + \sum_{i=0}^{\ell} \mathtt{rlk}[i][0] \cdot c_{2,i}, \qquad c_1' = c_1 + \sum_{i=0}^{\ell} \mathtt{rlk}[i][1] \cdot c_{2,i}$$

and output $c_{\mathtt{Relin}} = (c_0', c_1')$.
- Mul($\mathtt{ct}_0, \mathtt{ct}_1, \mathtt{rlk}$): Return $c_{\mathtt{Mul}} = (c_0', c_1') = \mathtt{Relin}(\mathtt{BasicMul}(\mathtt{ct}_0, \mathtt{ct}_1), \mathtt{rlk})$.

4.2 Ciphertext size

In this section we describe the memory overhead of FV with a polynomial plaintext modulus.

The memory overhead is defined by two encryption parameters: the ciphertext modulus q and the ring dimension n. Furthermore, the same parameters and the standard deviation σ determine the security level of an HE scheme. In practice, n and σ are usually fixed whereas q is chosen according to the desired security level and homomorphic operations to be performed. If no appropriate q is found, then this search is repeated for a larger n.

The security level of the parameter triple (q, n, σ) can be computed via the LWE-estimator of Albrecht et al. [1]. To find q that guarantees decryption correctness for the output of a given homomorphic circuit, one can use the following heuristic analysis with fixed n and σ.

The decryption correctness is closely related to the size of the ciphertext invariant noise. The *invariant noise* of a ciphertext $\mathsf{ct} = (c_0, c_1)$ encrypting a plaintext $\mathsf{msg} \in R_{X^m + b}$ is an element $v \in K$ with the smallest canonical norm such that

$$\frac{X^m + b}{q} \cdot [c_0 + c_1 \cdot s]_q = \mathsf{msg} + v + g \cdot (X^m + b)$$

for some $g \in R$. It is easy to see that $\mathtt{Decrypt}$ returns msg if $|v|_\infty < 1/2$, i.e. the rounding step removes v. Since $|v|_\infty \leq \|v\|^{\mathrm{can}}$, one can switch to the heuristic analysis of the canonical norm to show that $\|v\|^{\mathrm{can}} < 1/2$.

Fresh noise heuristic [3]. Let ct be a fresh ciphertext $\mathsf{ct} = \mathtt{Encrypt}(\mathsf{pk}, \mathsf{msg})$, then the invariant noise v of ct is bounded with very high probability by

$$\|v\|^{\mathrm{can}} \leq \frac{b+1}{q} \left(\|\mathsf{msg}\|^{\mathrm{can}} \cdot n\sqrt{3n} + 2\sigma\sqrt{12n^2 + 9n} \right) . \tag{1}$$

Since the right-hand side should be smaller than $1/2$, the minimal ciphertext modulus supporting the decryption correctness should satisfy $q \in \Omega\left(b^2 n\sqrt{n}\right)$.

Homomorphic arithmetic operations increase the invariant noise. It can be easily seen that homomorphic addition results in an additive noise growth, whereas homomorphic multiplication induces a linear growth as shown below.

Multiplication noise heuristic [3]. Let $\mathsf{ct}(\mathsf{msg}, v)$ be a ciphertext encrypting message $\mathsf{msg} \in R_{X^m+b}$ with invariant noise v. Given two ciphertexts $\mathsf{ct}_1 = \mathsf{ct}(\mathsf{msg}_1, v_1)$ and $\mathsf{ct}_1 = \mathsf{ct}(\mathsf{msg}_2, v_2)$, the function $\mathtt{Mul}(\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{rlk})$ outputs a ciphertext $\mathsf{ct}_{\mathtt{Mul}} = \mathsf{ct}(\mathsf{msg}_1 \cdot \mathsf{msg}_2, v_{\mathtt{Mul}})$ with

$$\begin{aligned} \|v_{\mathtt{Mul}}\|^{\mathrm{can}} \leq (b+1)\sqrt{3n + 2n^2} \left(\|v_1\|^{\mathrm{can}} + \|v_2\|^{\mathrm{can}}\right) &+ 3\|v_1\|^{\mathrm{can}}\|v_2\|^{\mathrm{can}} \\ + \frac{b+1}{q}\sqrt{3n + 2n^2 + 4n^3/3} &+ \frac{b+1}{q}\sigma n w\sqrt{3(\ell+1)} \end{aligned} \tag{2}$$

with very high probability. Let $v_{\mathtt{Mul}}^L$ be an invariant noise after L multiplicative levels. If $L = 0$, then it follows from (1) and the additive noise growth after homomorphic addition that $\left\|v_{\mathtt{Mul}}^0\right\|^{\mathrm{can}} \in O\left(b^2 n\sqrt{n}/q\right)$. Computing $\left\|v_{\mathtt{Mul}}^1\right\|^{\mathrm{can}}$, one can notice that the first term of the right-hand side in (2) is dominant and

thus $\left\|v_{\mathtt{Mul}}^1\right\|^{\mathrm{can}} \in O\left(bnv_{\mathtt{Mul}}^0\right)$, or $\left\|v_{\mathtt{Mul}}^1\right\|^{\mathrm{can}} \in O\left(b^3n^2\sqrt{n}/q\right)$. By induction, we obtain

$$\left\|v_{\mathtt{Mul}}^L\right\|^{\mathrm{can}} \in O\left(\frac{b^{L+2}n^{L+1}\sqrt{n}}{q}\right) .$$

Given that $\left\|v_{\mathtt{Mul}}^L\right\|^{\mathrm{can}}$ should be less than $1/2$ to guarantee the decryption correctness, the ciphertext modulus should satisfy

$$q \in \Omega\left(b^{L+2}n^{L+1}\sqrt{n}\right) . \tag{3}$$

5 Encoding of packed values into FV

To employ the HEAAN packing method in the FV scheme, we need to map elements of R' to the plaintext ring R_{X^m+b}. For this purpose, we resort to the encoding algorithm of Bootland et al. [3], which maps cyclotomic integers from $R' = \mathbb{Z}[\zeta_{2m}]$ to the plaintext space R_{X^m+b} isomorphic to $\mathbb{Z}[X]/(X^n + 1, X^m + b)$. A similar technique was given by Chen et al. [4] for the plaintext modulus $X + b$.

Let $\hat{a} \in \mathbb{Z}/(b^{n/m}+1)\mathbb{Z}$ be the representative of an integer a modulo $b^{n/m}+1$ in the symmetric interval $\left[-(b^{n/m}+1)/2, (b^{n/m}+1)/2\right)$. Assume that $\hat{b} = \hat{\alpha}^m$ for some α. This assumption might seem too strong for the reader but there exist special forms of b such that $\hat{\alpha}$ is efficiently computable; we discuss them later in this section. Since b is co-prime to $b^{n/m}+1$, there exist the multiplicative inverse of $\hat{\alpha}$, denoted $\hat{\beta}$. This implies that βX is a primitive $2m$-th root of unity in R_{X^m+b}, namely $(\hat{\beta}X)^m = \hat{\alpha}^{-m}X^m = \hat{b}^{-1}X^m = -1$. Therefore, the map $\zeta_{2m} \mapsto \hat{\beta}X$ induces the following ring homomorphism

$$\mathbb{Z}[\zeta_{2m}] \to R_{X^m+b} : \qquad \sum_{i=0}^{m-1} a_i\zeta_{2m}^i \mapsto \sum_{i=0}^{m-1} \hat{a}_i\hat{\beta}^iX^i . \tag{4}$$

This map outputs polynomials of degree less than m with coefficients exponential in b. Such large coefficients drastically increase the invariant noise as you can see in (1). Therefore, the next step is to switch to another representative modulo $X^m + b$ by spreading this polynomial across the power range $1, X, \ldots, X^{n-1}$ while making the plaintext coefficients smaller. It can be done by computing the balanced b-ary expansion of each coefficient and then mapping powers of b to corresponding powers of $-X^m$. The result is then lifted to $R = \mathbb{Z}[X]/(X^n + 1)$ and fed to the FV scheme.

The homomorphism (4) is surjective with kernel $\left(b^{n/m} + 1\right)$. Therefore, it induces an isomorphism between cyclotomic integers from $\mathbb{Z}[\zeta_{2m}]/\left(b^{n/m} + 1\right)$ and R_{X^m+b}; thus, the encoding and the decoding maps are well defined. To decode an element $c \in R_{X^m+b}$, we first compute $c' = c \bmod (X^m + b)$ and then map X to $\hat{\alpha} \cdot \zeta_{2m}$, which results in

$$c' = \sum_{i=0}^{m-1} \hat{c}_i'X^i \mapsto \sum_{i=0}^{m-1} \hat{c}_i'\hat{\alpha}^i\zeta_{2m}^i .$$

As a result, the homomorphism defined by (4) serves as an encoding map from cyclotomic integers to the plaintext space R_{X^m+b}. Using this map together with the `Pack` function from Section 3, we can encrypt $m/2$ complex numbers into `FV` without using previously known packing techniques based on the Chinese Remainder Theorem [24].

The advantage of this encoding technique is that the unused part of the plaintext space coming from the large dimension n is transformed into a larger integral modulus, reflected in the exponent n/m. However, the encoding algorithm of `HEAAN`, where ζ_{2m} is mapped to $X^{n/m}$, is not surjective as plaintexts belong to an m-dimensional subspace of the plaintext space. Thus, a large part of the plaintext space remains unused.

5.1 Choice of b.

As mentioned earlier, the encoding algorithm assumes that b is an m-th power residue modulo $b^{n/m} + 1$. Moreover, its m-th root α is efficiently computable. When m is a positive power of 2, finding α is at least as hard as finding a square root of b. Since factoring $b^{n/m}+1$ and extracting square roots modulo $b^{n/m}+1$ are computationally equivalent [20], an efficient algorithm for computing α implies the existence of an efficient factoring algorithm for generalized Fermat numbers of the form $b^{2^k} + 1$. Unfortunately, no efficient prime factorization algorithm for these numbers is found.

There exists a specific b whose m-th roots are efficiently computable. In particular, if $b = 2^{m/2}$ then α must be congruent to the square root of 2 modulo $b^{n/m} + 1 = 2^{n/2} + 1$. In this case, it is easy to verify that $\alpha = 2^{n/8}\left(2^{n/4} - 1\right)$. Unfortunately, such b is exponential in m, so invariant noise grows exponentially faster as the number of packing slots increases. Therefore, fewer homomorphic operations are affordable when the packing capacity increases.

Another interesting choice of b is when $b < 2^{m/2}$ and $b^{n/m} + 1$ becomes a generalized Fermat prime. Thus, α can be efficiently computed by the Tonelli-Shanks algorithm [23]. Note that in this case b must be even, or $b = 2^k c$ for some $k > 0$ and odd c. It follows from [3, Lemma 1] that if b is an m-th power residue, then $2n$ divides $b^{n/m} = 2^{kn/m}c^{n/m}$. As a result, n should divide $2^{kn/m-1}$. Since n is a power of two, we obtain that $\log_2 n \leq kn/m - 1$. Given this constraint and the fact that n/m is at most 2^{16} in practice, we can find numerous suitable bases b of generalized Fermat primes, see Table 6 in Appendix A.

To be decoded correctly, a cyclotomic integer a from $\mathbb{Z}[\zeta_{2m}]$ should have an infinity norm bounded as follows

$$|a|_\infty < \frac{b^{n/m} + 1}{2}\,. \tag{5}$$

Let $a_i \in \mathbb{Z}[\zeta_{2m}]$ be output values of `Pack`$(\mathbf{z}_i)$ for complex vectors $\mathbf{z}_i \in \mathbb{C}^{m/2}$ with $|\mathbf{z}_i|_\infty \leq B$ for some B. According to Section 3, the infinity norm of a_i

represented in the power basis of $\mathbb{Z}[\zeta_{2m}]$ is bounded by

$$|a_i|_\infty = \left\| \left\lfloor \frac{\Delta}{m} \cdot \mathbf{\Sigma}^{-1} \mathbf{z}_i \right\rceil \right\|_\infty \le \Delta B + \frac{1}{2} . \tag{6}$$

It follows from Lemma 1 that the packing scale Δ must be at least $\frac{pm}{2} + \varepsilon$ for small $\varepsilon > 0$ to pack $\mathbf{z}_i$ with precision p. Hence, the infinity norm of a_i has the following upper bound $|a_i|_\infty \le V = \left(\frac{pm}{2} + \varepsilon \right) \cdot B + \frac{1}{2}$. For any a_i, a_j it holds $|a_i a_j|_\infty \le mV^2$. It follows by induction that after L multiplicative levels the infinity norm increases up to $m^{2^L-1} V^{2^L}$. From the decoding requirement (5) we obtain that $m^{2^L-1} V^{2^L}$ must be smaller than $\frac{b^{n/m}+1}{2}$, which leads to $b \in \Omega \left(m^{\frac{m}{n}(2^{L+1}-1)} \cdot (pB)^{\frac{m}{n} \cdot 2^L} \right)$. Substituting this estimation into (3), we can see how the ciphertext modulus depends on the ring dimension n, the packing capacity m, the input precision p, the input bound B and the circuit depth L, namely

$$q \in \Omega \left(m^{\frac{m}{n}(2^{L+1}-1)(L+2)} \cdot (pB)^{\frac{m}{n} 2^L (L+2)} \cdot n^{L+1} \sqrt{n} \right) . \tag{7}$$

6 Asymptotic comparison with HEAAN

We start the comparison of our scheme with HEAAN by estimating how large should be the ciphertext modulus in this scheme to support correct evaluation of given circuits. Let us first describe the HEAAN scheme as defined in [5].

Let $q_L > \cdots > q_\ell > \cdots > q_0$ be a ladder of ciphertext moduli. Take a large integer $P \simeq q_L$. Let h be a positive integer. The key distribution χ_k draws random elements from R with ternary coefficients and Hamming weight h.

The basic encryption functions of HEAAN are the following:

- KeyGen(1^n): Let $s \leftarrow \chi_k$ and $e, e' \leftarrow \chi_e$. Sample uniformly random $a \in R_{q_L}$ and $a' \in R_{P \cdot q_L}$. Output
  – the secret key $\mathtt{sk} = s$,
  – the public key $\mathtt{pk} = \left([-a \cdot s + e]_{q_L} , a \right)$,
  – the evaluation key $\mathtt{rlk} = \left([-a' \cdot s + e' + P \cdot s^2]_{P \cdot q_L} , a' \right)$.
- Encrypt($\mathtt{pk}, \mathtt{msg} \in R_{q_L}$): Sample $u \leftarrow \chi_k$ and $e_0, e_1 \leftarrow \chi_e$. Set $p_0 = \mathtt{pk}[0]$, $p_1 = \mathtt{pk}[1]$ and output $\mathtt{ct} = (c_0, c_1)$ where

$$c_0 = [\mathtt{msg} + p_0 \cdot u + e_0]_{q_L} , \qquad c_1 = [p_1 \cdot u + e_1]_{q_L} .$$

- Decrypt($\mathtt{sk}, \mathtt{ct}$): Return $\mathtt{msg}' = [c_0 + c_1 \cdot s]_{q_L}$.

To encode a cyclotomic integer $a \in \mathbb{Z}[\zeta_{2m}]$ with $|a|_\infty < q_L/2$, we embed a to R_{q_L} using the map $\zeta_{2m} \mapsto X^{n/m}$ and the reduction modulo q_L. Notice that the decryption algorithm outputs $\mathtt{msg}' = \mathtt{msg} + e'$ with a noisy component $e' = e_0 + e_1 s + ue$. Therefore, to encrypt a complex vector $\mathbf{z} \in \mathbb{C}^{m/2}$ with the input precision p in HEAAN, the packing scale Δ must be larger than

in Lemma 1 to compensate a precision loss induced by this noise. Let $\mathbf{z}' = \texttt{Unpack}(\texttt{Decrypt}(\texttt{Encrypt}(\texttt{Pack}(\mathbf{z}))))$, then, following the reasoning of Lemma 1, we obtain

$$|\mathbf{z} - \mathbf{z}'|_\infty = \max_i \left| \frac{1}{\Delta} \sum_{k=0}^{m-1} (e_k + e'_k) \cdot \zeta_{2m}^{ik} \right| \le \frac{m}{2\Delta} + \frac{mr}{\Delta}$$

where $|e'|_\infty \le r \in O(n)$. To have $|\mathbf{z} - \mathbf{z}'|_\infty < 1/p$, the packing scale Δ must then satisfy the following bound

$$\Delta > mp \left(\frac{1}{2} + r \right) , \tag{8}$$

which results in $\Delta \in \Omega(mpn)$.

6.1 Homomorphic operations

In HEAAN, homomorphic operations can output ciphertexts with a smaller ciphertext modulus in comparison to their input. Therefore, ciphertext moduli of input ciphertexts lie between q_0 and q_L. Below we assume that ciphertexts $\texttt{ct}_1$ and $\texttt{ct}_2$ are given modulo q_ℓ. The basic homomorphic operations such as addition and multiplication are defined as follows.

- $\texttt{Add}(\texttt{ct}_0, \texttt{ct}_1)$: Return $\texttt{ct}_{\texttt{Add}} = \left([\texttt{ct}_0[0] + \texttt{ct}_1[0]]_{q_\ell}, [\texttt{ct}_0[1] + \texttt{ct}_1[1]]_{q_\ell} \right)$.
- $\texttt{BasicMul}(\texttt{ct}_0, \texttt{ct}_1)$: Return $\texttt{ct}_{\texttt{BasicMul}} = (c_0, c_1, c_2)$ where

$$c_0 = [\texttt{ct}_0[0] \cdot \texttt{ct}_1[0]]_{q_\ell}, \quad c_1 = [\texttt{ct}_0[0] \cdot \texttt{ct}_1[1] + \texttt{ct}_0[1] \cdot \texttt{ct}_1[0]]_{q_\ell},$$
$$c_2 = [\texttt{ct}_0[1] \cdot \texttt{ct}_1[1]]_{q_\ell}.$$

- $\texttt{Relin}(\texttt{ct}_{\texttt{BasicMul}}, \texttt{rlk})$: Output $c_{\texttt{Relin}} = (c'_0, c'_1)$ where

$$c'_0 = \left[c_0 + \left\lfloor P^{-1} \cdot c_2 \cdot \texttt{rlk}[0] \right\rceil \right]_{q_\ell}, \quad c'_1 = \left[c_1 + \left\lfloor P^{-1} \cdot c_2 \cdot \texttt{rlk}[1] \right\rceil \right]_{q_\ell}$$

- $\texttt{Mul}(\texttt{ct}_0, \texttt{ct}_1, \texttt{rlk})$: Return $\texttt{ct}_{\texttt{Mul}} = (c'_0, c'_1) = \texttt{Relin}(\texttt{BasicMul}(\texttt{ct}_0, \texttt{ct}_1), \texttt{rlk})$.

In addition, HEAAN has a special function called rescaling, which imitates rounding. Rescaling discards least significant bits of a given ciphertext and reduces the ciphertext modulus.

- $\texttt{Rescale}(\texttt{ct}, \ell, \ell')$: Output $\texttt{ct}_{\texttt{Rescale}} = \left(\left\lfloor \frac{q_{\ell'}}{q_\ell} c_0 \right\rceil, \left\lfloor \frac{q_{\ell'}}{q_\ell} c_1 \right\rceil \right) \in R_{q_{\ell'}}^2$.

Note that if the input ciphertext $\texttt{ct}$ encrypts a plaintext $\texttt{msg}$, then $\texttt{ct}_{\texttt{Rescale}}$ is a valid encryption of $(q_{\ell'}/q_\ell) \cdot \texttt{msg}$. Hence, rescaling can help to control the coefficient size of plaintexts, especially after multiplication. Let $\texttt{ct}_0$ and $\texttt{ct}_1$ be ciphertexts of two complex vectors $\mathbf{z}_1$ and $\mathbf{z}_2$ packed with scale Δ. The product of these ciphertexts is an encryption of the Hadamard product $\mathbf{z} = \mathbf{z}_1 \odot \mathbf{z}_2$ with scale Δ^2. If $q_\ell/q_{\ell'} \simeq \Delta$, then $\texttt{Rescale}(\texttt{ct})$ outputs a ciphertext, which again encrypts $\mathbf{z}$ but with packing scale Δ. As a result, the unpacking scale Δ' in HEAAN can be equal to Δ for any circuit, whereas in our scheme the depth of a circuit should be known to set Δ' to a correct power of Δ.

11

6.2 Ciphertext size

Let msg_i be plaintext messages encoding complex vectors $\mathbf{z}_i \in \mathbb{C}^{m/2}$ with $|\mathbf{z}_i|_\infty \le B$ for some B. To be decrypted and then decoded correctly, a plaintext should have an infinity norm smaller than $q_0/2$. As in Section 4, we switch to the canonical norm in order to analyze how plaintexts approach this bound. The canonical norm of each msg_i is bounded by $|\varDelta \mathbf{z}_i|_\infty + \|e\|^{\mathrm{can}} = \varDelta B + m/2$ where e is the rounding error. Let $V = \varDelta B + m/2$.

Assume that $q_\ell/q_{\ell-1} \simeq \varDelta$ for any $\ell \in [L]$. After multiplication and rescaling we obtain a ciphertext encrypting a plaintext msg such that $\|\mathsf{msg}\|^{\mathrm{can}} \le \frac{(V+\|E_e\|^{\mathrm{can}})^2}{\varDelta} + \|E_r\|^{\mathrm{can}}$ where E_e is the encryption noise and E_r is the noise introduced by `Relin` and `Rescale`. Since $\|E_e\|^{\mathrm{can}}, \|E_r\|^{\mathrm{can}} \in O(n)$ according to [5, Lemmas 1-3], it follows from (8) that $\|\mathsf{msg}\|^{\mathrm{can}} \in O(\varDelta B^2)$. Hence, after L multiplicative levels the canonical norm of a resulting plaintext satisfies $\|\mathsf{msg}\|^{\mathrm{can}} \in O\left(\varDelta B^{2^L}\right)$. As $\|\mathsf{msg}\|^{\mathrm{can}}$ should be smaller than $q_0/2$, we obtain that $q_0 \in \Omega\left(\varDelta B^{2^L}\right)$. Since rescaling decreases the ciphertext modulus L times to reach q_0, the initial ciphertext modulus should be set to $q_L \simeq q_0 \cdot \varDelta^L$. Thus, $q_L \in \Omega\left(\varDelta^{L+1} B^{2^L}\right)$ and (8) yields $q_L \in \Omega\left(m^{L+1} \cdot p^{L+1} \cdot B^{2^L} \cdot n^{L+1}\right)$. Comparing the above estimation with its analog for our scheme, we can see that if $\frac{m}{n} = \frac{1}{2^{L+1}}$, then (7) turns into $q \in \Omega\left(m^{(L+2)\left(1-\frac{1}{2^{L+1}}\right)} \cdot (pB)^{\frac{L+2}{2}} \cdot n^{L+1} \sqrt{n}\right)$. It implies that when $B > (m\sqrt{n})^{1/2^{L-1}}$, our scheme requires a smaller ciphertext modulus. More specifically, our approach results in a smaller memory overhead in comparison to `HEAAN` in the following cases:

- in shallow circuits with large ratios between the packing capacity and the dimension of R, namely $m/n \le 1/4$;
- in deep circuits with a small packing capacity, i.e. $m = n/2^{L+1}$.

7 Practical comparison with `HEAAN`

In this section we demonstrate the efficiency of our scheme in comparison to the `HEAAN` scheme. In particular, we homomorphically computed the functions presented in [5, Section 5] including power functions, the exponential function and the logistic regression function. In addition, we performed experiments with the sine function.

We implemented our scheme and two versions of `HEAAN` in SageMath [25]. The implementation script can be found at `https://github.com/iliailia/heaan-vs-fv-sage`. One version of `HEAAN` corresponds to the original scheme given in [5] with sparse secret keys and the relinearization method described in Section 6.1. While these features can speed up computations, they introduce a larger memory overhead than in our scheme as larger encryption parameters are needed to support the same security level. To perform a fair comparison with our scheme, we implemented a second variant of `HEAAN`, denoted `HEAAN*`, without

sparse secret keys and with the same relinearization method (see Section 4.1) as in our scheme.

For all the implemented schemes we found minimal encryption parameters that support both correct computation of the above functions and a security level of at least 128 bits. To achieve this security level we set the parameters of the original HEAAN scheme according to the recent recommendations for sparse-secret RLWE [8]. Namely, we set the sparsity parameter $h = 128$. The standard deviation σ of the error distribution χ_e is set to 3.19.

7.1 Non-polynomial functions: logistic regression, sine and exponential function

#slots	1	2	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9
Our scheme										
Size, KB	94	97	104	228	238	238	238	244	334	704
Time, sec	6.04	6.75	6.21	13.52	12.92	12.76	12.80	12.99	18.16	38.81
HEAAN										
Size, KB	448	464	476	492	512	528	548	568	584	604
Time, sec	18.22	18.18	17.61	17.97	18.76	18.97	18.24	18.09	18.28	18.89
HEAAN*										
Size, KB	274	284	292	302	312	322	332	342	352	362
Time, sec	14.48	14.45	14.28	14.46	14.52	16.98	18.07	18.14	18.00	18.24

Table 1. The ciphertext size and the running time to compute the logistic function in the interval $[-2.1, 2.1]$.

#slots	1	2	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}
Our scheme												
Size, KB	94	97	104	228	238	238	238	264	380	796	1292	2680
Time, sec	5.85	6.52	6.25	12.34	13.22	12.83	12.70	14.63	18.75	43.48	73.84	155.66
HEAAN												
Size, KB	564	580	600	620	640	660	680	1400	1440	1480	1520	1560
Time, sec	18.14	18.13	18.25	18.55	19.67	19.58	19.56	40.21	40.40	44.57	47.20	50.27
HEAAN*												
Size, KB	310	320	330	340	350	360	370	380	390	400	410	420
Time, sec	14.52	16.79	16.93	17.07	17.62	17.58	16.80	17.23	17.49	19.20	19.89	21.74

Table 2. The ciphertext size and the running time to compute the sine in the interval $[-\pi, \pi]$.

As in [5], we approximate the logistic function $1/(1 + e^{-x})$ and the sine with Maclaurin series of degree 9. The exponential function e^x is evaluated via its

13

#slots	1	2	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Our scheme											
Size, KB	87	89	95	105	105	105	107	224	310	652	1044
Time, sec	6.42	6.00	5.83	6.92	6.85	6.68	7.07	14.73	18.23	43.88	68.05
HEAAN											
Size, KB	480	500	520	540	560	580	600	620	640	660	680
Time, sec	19.63	20.02	20.20	20.30	20.25	20.41	20.53	20.76	21.50	21.87	23.74
HEAAN*											
Size, KB	290	300	310	320	330	340	350	360	370	380	390
Time, sec	15.57	15.43	15.66	18.22	18.64	18.67	18.74	18.78	18.65	19.71	20.76

Table 3. The ciphertext size and the running time to compute e^x in the interval $[-2.3, 2.3]$.

Maclaurin series of degree 8. These approximations are accurate up to 7 bits of binary precision in $[-2.1, 2.1]$ for the logistic regression, $[-\pi, \pi]$ for the sine function and $[-2.3, 2.3]$ for the exponential function.

We conducted experiments with encryption and packing parameters that support homomorphic evaluation of the above series within 7 bits of binary precision. More detailed description of these parameters is given in Appendix B. The results of our experiments are presented in Tables 1-2. In particular, our scheme needs 4-6 times and around 3 times less memory than HEAAN and HEAAN*, respectively, to perform computations on a small number of data slots. This advantage is declining with an increasing number of slots as predicted by the theoretical estimations of Section 6. Starting from only 512-2048 packing slots both versions of HEAAN need less memory than our scheme.

7.2 Power functions

#slots	1	2	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}
Our scheme													
Size, KB	24	24.5	25.5	25.5	25.5	25.5	26	54	66	102	212	360	740
Time, sec	0.22	0.20	0.19	0.19	0.19	0.19	0.20	0.44	0.62	0.80	1.71	2.57	4.93
HEAAN													
Size, KB	388	396	404	412	420	428	436	444	452	460	468	476	484
Time, sec	1.47	1.41	1.39	1.47	1.47	1.48	1.42	1.49	1.47	1.50	1.58	1.68	1.98
HEAAN*													
Size, KB	97	99	101	103	105	107	109	222	226	230	234	238	242
Time, sec	0.60	0.61	0.61	0.61	0.61	0.60	0.61	1.33	1.32	1.35	1.43	1.60	1.79

Table 4. The ciphertext size and the running time to compute x^2 in the interval $(-2^{15}, 2^{15})$.

#slots	1	2	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
Our scheme											
Size, KB	95	98	105	230	242	242	260	360	756	1176	2496
Time, sec	2.88	3.08	3.13	6.53	6.37	6.56	7.70	8.88	19.15	33.42	70.72
HEAAN											
Size, KB	1672	1712	1752	1792	1832	1872	1912	1952	1992	2032	2072
Time, sec	20.76	20.38	20.57	20.70	20.45	20.96	21.02	21.91	22.06	23.56	23.70
HEAAN*											
Size, KB	408	418	428	876	896	916	936	956	976	996	1016
Time, sec	8.42	8.43	8.66	18.45	18.98	19.73	19.53	20.29	20.95	20.33	22.46

Table 5. The ciphertext size and the running time to compute x^{16} in the interval $[-2.1, 2.1]$.

We also computed two simple polynomial functions x^{16} and x^2 with input values taken from $[-2.1, 2.1]$ and $(-2^{15}, 2^{15})$, respectively. As for non-linear functions we aim to achieve 7 bits of binary precision for output values.

As seen in Tables 5 and 4, our scheme with a small number of slots requires around 18 and 4 times less memory than HEAAN and HEAAN*, respectively. However, the memory overhead of our method grows exponentially with the number of slots. The maximal number of slots where our scheme still outperforms HEAAN is 512 for x^{16} and 2048 for x^2. Comparing with HEAAN*, these numbers are 256 for x^{16} and 1024 for x^2.

8 Conclusion

While the HEAAN scheme has achieved significant success in recent years, especially in privacy-preserving machine learning applications, in many cases computations are not as highly parallelizable as would be optimal for the HEAAN scheme. In this work we have demonstrated how in these cases an approach generalizing that of Bootland et al. and Chen et al. can yield significant performance improvements in terms of encryption parameter sizes and subsequently in ciphertext sizes. This can be particularly important when using homomorphic encryption in low-latency applications, where communication complexity quickly becomes a bottleneck.

References

1. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology **9**(3), 169–203 (2015)

2. Bonte, C., Bootland, C., Bos, J.W., Castryck, W., Iliashenko, I., Vercauteren, F.: Faster homomorphic function evaluation using non-integral base encoding. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 579–600. Springer, Heidelberg (Sep 2017)

3. Bootland, C., Castryck, W., Iliashenko, I., Vercauteren, F.: Efficiently processing complex-valued data in homomorphic encryption. Special Issue of Journal of Mathematical Cryptology: Mathcrypt 2018 (to be published)

4. Chen, H., Laine, K., Player, R., Xia, Y.: High-precision arithmetic in homomorphic encryption. In: Smart, N.P. (ed.) CT-RSA 2018. LNCS, vol. 10808, pp. 116–136. Springer, Heidelberg (Apr 2018)

5. Cheon, J.H., Kim, A., Kim, M., Song, Y.S.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 409–437. Springer, Heidelberg (Dec 2017)

6. Costache, A., Smart, N.P., Vivek, S.: Faster homomorphic evaluation of discrete fourier transforms. In: Kiayias, A. (ed.) FC 2017. LNCS, vol. 10322, pp. 517–529. Springer, Heidelberg (Apr 2017)

7. Costache, A., Smart, N.P., Vivek, S., Waller, A.: Fixed-point arithmetic in SHE schemes. In: Avanzi, R., Heys, H.M. (eds.) SAC 2016. LNCS, vol. 10532, pp. 401–422. Springer, Heidelberg (Aug 2016)

8. Curtis, B.R., Player, R.: On the feasibility and impact of standardising sparse-secret LWE parameter sets for homomorphic encryption. In: WAHC'19. ACM Press (2019)

9. Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 643–662. Springer, Heidelberg (Aug 2012)

10. Dowlin, N., Gilad-Bachrach, R., Laine, K., Lauter, K.E., Naehrig, M., Wernsing, J.: Manual for using homomorphic encryption for bioinformatics. Proceedings of the IEEE **105**(3), 552–567 (2017)

11. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory **31**, 469–472 (1985)

12. Erkin, Z., Troncoso-Pastoriza, J.R., Lagendijk, R.L., Pérez-González, F.: Privacy-preserving data aggregation in smart metering systems: An overview. IEEE Signal Processing Magazine **30**(2), 75–86 (2013)

13. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012), `http://eprint.iacr.org/2012/144`

14. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) 41st ACM STOC. pp. 169–178. ACM Press (May / Jun 2009)

15. Goldwasser, S., Micali, S.: Probabilistic encryption and how to play mental poker keeping secret all partial information. In: 14th ACM STOC. pp. 365–377. ACM Press (May 1982)

16. Kocabas, O., Soyata, T., Couderc, J.P., Aktas, M., Xia, J., Huang, M.: Assessment of cloud-based health monitoring using homomorphic encryption. In: 2013 IEEE 31st International Conference on Computer Design (ICCD). pp. 443–446. IEEE (2013)

17. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (May / Jun 2010)

18. Malina, L., Hajny, J., Fujdiak, R., Hosek, J.: On perspective of security and privacy-preserving solutions in the internet of things. Computer Networks **102**, 83–95 (2016)

19. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT'99. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (May 1999)
20. Rabin, M.O.: Digitalized signatures and public-key functions as intractable as factorization. Tech. rep., Massachusetts Inst of Tech Cambridge Lab for Computer Science (1979)
21. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. Foundations of secure computation 4(11), 169–180 (1978)
22. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the Association for Computing Machinery $\mathbf{21}$(2), 120–126 (1978)
23. Shanks, D.: Five number-theoretic algorithms. In: Proceedings of the Second Manitoba Conference on Numerical Mathematics (1973)
24. Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. Des. Codes Cryptography $\mathbf{71}$(1), 57–81 (Apr 2014)
25. The Sage Developers: SageMath, the Sage Mathematics Software System (Version 8.9) (2019), https://www.sagemath.org

A Examples of b

As shown in Section 5, the plaintext space parameter b must be an mth power residue modulo $b^{n/m}+1$ and its m root must be efficiently computable to allow the HEAAN encoding of complex numbers. Here we present a collection of these parameters for given practical choices of the ring dimension n and the packing capacity m.

m	n					
	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}
2	2	2	2	2	2	2
	(1024)	(2048)	(4096)	(8192)	(16384)	(32768)
2^2	4	4	4	4	4	4
	(1024)	(2048)	(4096)	(8192)	(16384)	(32768)
2^3	16	16	16	16	16	16
	(1024)	(2048)	(4096)	(8192)	(16384)	(32768)
2^4	120	256	46	256	150	256
	(884)	(2048)	(2828)	(8192)	(14804)	(32768)
2^5	274	120	278	46	824	150
	(518)	(884)	(2078)	(2828)	(9918)	(14804)
2^6	884	274	120	278	46	824
	(313)	(518)	(884)	(2078)	(2828)	(9918)
2^7	984	884	274	120	278	46
	(159)	(313)	(518)	(884)	(2078)	(2828)
2^8	1028	984	884	274	120	278
	(80)	(159)	(313)	(518)	(884)	(2078)
2^9	9872	1028	984	884	274	120
	(53)	(80)	(159)	(313)	(518)	(884)
2^{10}	9600	9872	1028	984	884	274
	(26)	(53)	(80)	(159)	(313)	(518)
2^{11}	2^{15}	9600	9872	1028	984	884
	(16)	(26)	(53)	(80)	(159)	(313)
2^{12}	x	2^{15}	9600	9872	1028	984
		(16)	(26)	(53)	(80)	(159)

Table 6. Examples of b such that b is an m-th power residue modulo $b^{n/m}+1$ for practical choices of m and n. Numbers in parentheses are equal to $\left\lfloor \log_2(b^{n/m}+1) \right\rfloor$, which is the maximal coefficient size of HEAAN encodings. For each b we precomputed its m-th root using several calls of the Tonelli-Shanks algorithm (square_root_mod_prime) in SageMath.

B Results of experiments

The following tables present the detailed encoding and encryption parameters used in the experiments conducted in Section 7. This data is the full version of

Tables 1-5 from Section 7. In all the tables, Δ denotes the packing scale, n is the dimension of the cyclotomic ring R and b is the constant term of the plaintext modulus. The total running time is averaged over 10 runs.

#slots	1	2	2^2	2^3	$2^4(*)$	$2^5(*)$	2^6	2^7	2^8	2^9
Our scheme										
Δ	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}	2^{25}
n	2^{12}	2^{12}	2^{12}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{14}
$\log q$	94	97	104	114	119	119	119	122	167	176
b	2	4	16	46	102	102	102	156	$3.3e4$	$5.1e4$
Size, KB	94	97	104	228	238	238	238	244	334	704
Time, sec	6.04	6.75	6.21	13.52	12.92	12.76	12.80	12.99	18.16	38.81
HEAAN										
Δ	2^{21}	2^{22}	2^{23}	2^{24}	2^{25}	2^{26}	2^{27}	2^{28}	2^{29}	2^{30}
n	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}
$\log q_L$	112	116	119	123	128	132	137	142	146	151
Size, KB	448	464	476	492	512	528	548	568	584	604
Time, sec	18.22	18.18	17.61	17.97	18.76	18.97	18.24	18.09	18.28	18.89
HEAAN*										
Δ	2^{27}	2^{28}	2^{29}	2^{30}	2^{31}	2^{32}	2^{33}	2^{34}	2^{35}	2^{36}
n	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}
$\log q_L$	137	142	146	151	156	161	166	171	176	181
Size, KB	274	284	292	302	312	322	332	342	352	362
Time, sec	14.48	14.45	14.28	14.46	14.52	16.98	18.07	18.14	18.00	18.24

Table 7. Encryption parameters to compute the logistic function in the interval $[-2.1, 2.1]$. The (*) symbol indicates that the maximal number of slots supported by the plaintext space is 2^6 for our scheme.

#slots	1	2	2^2	2^3	$2^4(*)$	$2^5(*)$	2^6	2^7	2^8	2^9	2^{10}	2^{11}
Our scheme												
Δ	2^{23}	2^{24}	2^{25}	2^{26}	2^{27}	2^{28}	2^{29}	2^{30}	2^{31}	2^{32}	2^{33}	2^{34}
n	2^{12}	2^{12}	2^{12}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{14}	2^{14}	2^{15}
b	2	4	16	46	102	102	102	562	$4.6e5$	$7.1e5$	$1.1e12$	$2.6e12$
$\log q$	94	97	104	114	119	119	119	132	190	199	323	335
Size, KB	94	97	104	228	238	238	238	264	380	796	1292	2680
Time, sec	5.85	6.52	6.25	12.34	13.22	12.83	12.70	14.63	18.75	43.48	73.84	155.66
HEAAN												
Δ	2^{27}	2^{28}	2^{29}	2^{30}	2^{31}	2^{32}	2^{33}	2^{34}	2^{35}	2^{36}	2^{37}	2^{38}
n	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{15}	2^{15}	2^{15}	2^{15}	2^{15}
$\log q_L$	141	145	150	155	160	165	170	175	180	185	190	195
Size, KB	564	580	600	620	640	660	680	1400	1440	1480	1520	1560
Time, sec	18.14	18.13	18.25	18.55	19.67	19.58	19.56	40.21	40.40	44.57	47.20	50.27
HEAAN*												
Δ	2^{30}	2^{31}	2^{32}	2^{33}	2^{34}	2^{35}	2^{36}	2^{37}	2^{38}	2^{39}	2^{40}	2^{41}
n	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}
$\log q_L$	155	160	165	170	175	180	185	190	195	200	205	210
Size, KB	310	320	330	340	350	360	370	380	390	400	410	420
Time, sec	14.52	16.79	16.93	17.07	17.62	17.58	16.80	17.23	17.49	19.20	19.89	21.74

Table 8. Encryption parameters to compute the sine in the interval $[-\pi, \pi]$. The (*) symbol indicates that the maximal number of slots supported by the plaintext space is 2^6 for our scheme.

#slots	1	2	2^2	2^3(*)	2^4(*)	2^5	2^6	2^7	2^8	2^9	2^{10}
Our scheme											
Δ	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}	2^{25}	2^{26}	2^{27}	2^{28}
n	2^{12}	2^{12}	2^{12}	2^{12}	2^{12}	2^{12}	2^{12}	2^{13}	2^{13}	2^{14}	2^{14}
b	2	4	16	102	102	102	132	156	2.9e4	4.3e4	4.1e9
$\log q$	87	89	95	105	105	105	107	112	155	163	261
Size, KB	87	89	95	105	105	105	107	224	310	652	1044
Time, sec	6.42	6.00	5.83	6.92	6.85	6.68	7.07	14.73	18.23	43.88	68.05
HEAAN											
Δ	2^{23}	2^{24}	2^{25}	2^{26}	2^{27}	2^{28}	2^{29}	2^{30}	2^{31}	2^{32}	2^{33}
n	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}
$\log q_L$	120	125	130	135	140	145	150	155	160	165	170
Size, KB	480	500	520	540	560	580	600	620	640	660	680
Time, sec	19.63	20.02	20.20	20.30	20.25	20.41	20.53	20.76	21.50	21.87	23.74
HEAAN*											
Δ	2^{28}	2^{29}	2^{30}	2^{31}	2^{32}	2^{33}	2^{34}	2^{35}	2^{36}	2^{37}	2^{38}
n	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}
$\log q_L$	145	150	155	160	165	170	175	180	185	190	195
Size, KB	290	300	310	320	330	340	350	360	370	380	390
Time, sec	15.57	15.43	15.66	18.22	18.64	18.67	18.74	18.78	18.65	19.71	20.76

Table 9. Encryption parameters to compute e^x in the interval $[-2.3, 2.3]$. The (*) symbol indicates that the maximal number of slots supported by the plaintext space is 2^5 for our scheme.

#slots	1	2	2^2	2^3(*)	2^4(*)	2^5	2^6	2^7	2^8	2^9	2^{10}
Our scheme											
Δ	2^{25}	2^{26}	2^{28}	2^{29}	2^{30}	2^{31}	2^{32}	2^{33}	2^{34}	2^{35}	2^{36}
n	2^{12}	2^{12}	2^{12}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{14}	2^{14}	2^{15}
b	2	4	16	46	120	120	412	$1.4e5$	$1.9e5$	$3.8e10$	$1.5e11$
$\log q$	95	98	105	115	121	121	130	180	189	294	312
Size, KB	95	98	105	230	242	242	260	360	756	1176	2496
Time, sec	2.88	3.08	3.13	6.53	6.37	6.56	7.70	8.88	19.15	33.42	70.72
HEAAN											
Δ	2^{38}	2^{39}	2^{40}	2^{41}	2^{42}	2^{43}	2^{44}	2^{45}	2^{46}	2^{47}	2^{48}
n	2^{15}	2^{15}	2^{15}	2^{15}	2^{15}	2^{15}	2^{15}	2^{15}	2^{15}	2^{15}	2^{15}
$\log q_L$	209	214	219	224	229	234	239	244	249	254	259
Size, KB	1672	1712	1752	1792	1832	1872	1912	1952	1992	2032	2072
Time, sec	20.76	20.38	20.57	20.70	20.45	20.96	21.02	21.91	22.06	23.56	23.70
HEAAN*											
Δ	2^{37}	2^{38}	2^{39}	2^{40}	2^{41}	2^{42}	2^{43}	2^{44}	2^{45}	2^{46}	2^{47}
n	2^{13}	2^{13}	2^{13}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}
$\log q_L$	204	209	214	219	224	229	234	239	244	249	254
Size, KB	408	418	428	876	896	916	936	956	976	996	1016
Time, sec	8.42	8.43	8.66	18.45	18.98	19.73	19.53	20.29	20.95	20.33	22.46

Table 10. Encryption parameters to compute x^{16} in the interval $[-2.1, 2.1]$. The (*) symbol indicates that the maximal number of slots supported by the plaintext space is 2^5 for our scheme.

#slots	1	2	2^2	2^3(*)	2^4(*)	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}
Our scheme													
Δ	2^{21}	2^{23}	2^{24}	2^{25}	2^{26}	2^{27}	2^{28}	2^{29}	2^{30}	2^{31}	2^{32}	2^{33}	2^{34}
n	2^{11}	2^{11}	2^{11}	2^{11}	2^{11}	2^{11}	2^{11}	2^{12}	2^{12}	2^{12}	2^{13}	2^{13}	2^{14}
b	2	4	16	30	30	30	44	74	$2.9e3$	$1.0e7$	$1.4e7$	$4.0e14$	$8.0e14$
$\log q$	48	49	51	51	51	51	52	54	66	102	106	180	185
Size, KB	24	24.5	25.5	25.5	25.5	25.5	26	54	66	102	212	360	740
Time, sec	0.22	0.20	0.19	0.19	0.19	0.19	0.20	0.44	0.62	0.80	1.71	2.57	4.93
HEAAN													
Δ	2^{33}	2^{34}	2^{35}	2^{36}	2^{37}	2^{38}	2^{39}	2^{40}	2^{41}	2^{42}	2^{43}	2^{44}	2^{45}
n	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}	2^{14}
$\log q_L$	97	99	101	103	105	107	109	111	113	115	117	119	121
Size, KB	388	396	404	412	420	428	436	444	452	460	468	476	484
Time, sec	1.47	1.41	1.39	1.47	1.47	1.48	1.42	1.49	1.47	1.50	1.58	1.68	1.98
HEAAN*													
Δ	2^{33}	2^{34}	2^{35}	2^{36}	2^{37}	2^{38}	2^{39}	2^{40}	2^{41}	2^{42}	2^{43}	2^{44}	2^{45}
n	2^{12}	2^{12}	2^{12}	2^{12}	2^{12}	2^{12}	2^{12}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}	2^{13}
$\log q_L$	97	99	101	103	105	107	109	111	113	115	117	119	121
Size, KB	97	99	101	103	105	107	109	222	226	230	234	238	242
Time, sec	0.60	0.61	0.61	0.61	0.61	0.60	0.61	1.33	1.32	1.35	1.43	1.60	1.79

Table 11. Encryption parameters to compute x^2 in the interval $(-2^{15}, 2^{15})$. The (*) symbol indicates that the maximal number of slots supported by the plaintext space is 2^5 for our scheme.