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

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

[1] Microsoft Research, Redmond, USA
{haoche, kim.laine}@microsoft.com
[2] imec-COSIC, Dept. Electrical Engineering, KU Leuven, Belgium
ilia@esat.kuleuven.be

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) [20] 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. [21,15,11,18]), 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 the product of these plaintexts with an error e as follows

$$\mathtt{Decrypt}(\mathtt{ct}_3) = \mathtt{pt}_1 \cdot \mathtt{pt}_2 + e\,.$$

The size of e defines the approximation 'closeness' between decrypted and expected results.

While the notion of AHE is apparently useless for exact computation, it 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 `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 `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 [12,16,17].

Our contribution. In this work, we show how to design a new SHE scheme that can perform computations on numerical data with smaller ciphertexts than in `HEAAN`.

The core idea is to exploit the recent variant of the `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 `FV` variant natively supports homomorphic computation on large cyclotomic integers $\mathbb{Z}[\zeta_{2m}]$ with small encryption parameters.

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

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

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

1.1 Paper organization

Section 2 introduces necessary mathematical notation. Sections 3-5 describe the design of our hybrid scheme. In Section 3, we describe the HEAAN packing map from the vector space $\mathbb{C}^{m/2}$ to the ring of cyclotomic integers $\mathbb{Z}[\zeta_{2m}]$ where m is a power of two. In Section 4, we recall a variant of the FV scheme with the plaintext space $\mathbb{Z}[X]/(X^n+1, X^m+b)$. Next, Section 5 we show how to encode cyclotomic integers to the above plaintext space, thus bridging HEAAN and FV as in (1). In Section 6, we provide a theoretical comparison of our hybrid scheme with HEAAN and outline a family of arithmetic circuits where the hybrid is more memory efficient. A practical comparison of these schemes is given in Section 7.

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].

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 $[-a/2, a/2)$.

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 $K' = \mathbb{Q}(\zeta_{2m})$, 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 $\sigma'_i : K' \to \mathbb{C}$ that fix every element of $\mathbb{Q}$. Each σ'_i is a complex embedding that maps ζ_{2m} to its algebraic conjugate over $K'/\mathbb{Q}$, or to another complex root of $X^m + 1$. In fact, these are the only field homomorphisms from K' to $\mathbb{C}$.

Let H' be a vector subspace of $\mathbb{C}^m$ such that

$$H' = \{(x_1, \ldots, x_m)^\intercal : x_{m-j+1} = \overline{x_j}, \forall j \in [m/2]\}.$$

This space is equipped with a projection map $\pi : H' \to \mathbb{C}^{m/2}$ that discards either of 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 $\|a\|^{\text{can}} = |\sigma(a)|_\infty$ for any $a \in K$. Since n is a power of two, $|a|_\infty \leq \|a\|^{\text{can}}$ as shown in [9]. In addition, $\|ab\|^{\text{can}} \leq \|a\|^{\text{can}} \|b\|^{\text{can}}$ 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

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

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

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

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 $\mathbf{z}' = \texttt{Unpack}(\Delta, \texttt{Pack}(\Delta, \mathbf{z}))$ 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 `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 $\texttt{Unpack}(\Delta, a)$, we obtain

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

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 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 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 FV scheme with polynomial plaintext modulus is defined as follows.

- $\texttt{KeyGen}(1^n)$: 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 $\texttt{sk} = s$,
  - the public key $\texttt{pk} = \left(\left[-(a \cdot s + e)\right]_q, a\right)$,
  - the evaluation key $\texttt{rlk} = \{(b_i, a_i)\}_{i=0}^{\ell}$.
- $\texttt{Encrypt}(\texttt{pk}, \mathsf{msg} \in R_{X^m+b})$: Sample $u \leftarrow \chi_k$ and $e_0, e_1 \leftarrow \chi_e$. Set $p_0 = \texttt{pk}[0]$ and $p_1 = \texttt{pk}[1]$. Output $\texttt{ct} = (c_0, c_1)$, where

$$\begin{aligned} c_0 &= \left[\Delta_b \cdot \mathsf{msg} + p_0 \cdot u + e_0\right]_q\,, \\ c_1 &= \left[p_1 \cdot u + e_1\right]_q \end{aligned}$$

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

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

4.1 Homomorphic operations

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

- $\texttt{Add}(\texttt{ct}_0, \texttt{ct}_1)$: Return

$$\texttt{ct}_{\texttt{Add}} = \left(\left[\texttt{ct}_0[0] + \texttt{ct}_1[0] \right]_q , \left[\texttt{ct}_0[1] + \texttt{ct}_1[1] \right]_q \right) .$$

- $\texttt{BasicMul}(\texttt{ct}_0, \texttt{ct}_1)$: Return $\texttt{ct}_{\texttt{BasicMul}} = (c_0, c_1, c_2)$, where

$$\begin{aligned}
c_0 &= \left[\left\lfloor \frac{X^m + b}{q} \cdot \texttt{ct}_0[0] \cdot \texttt{ct}_1[0] \right\rceil \right]_q , \\
c_1 &= \left[\left\lfloor \frac{X^m + b}{q} \cdot \left(\texttt{ct}_0[0] \cdot \texttt{ct}_1[1] + \texttt{ct}_0[1] \cdot \texttt{ct}_1[0] \right) \right\rceil \right]_q , \\
c_2 &= \left[\left\lfloor \frac{X^m + b}{q} \cdot \texttt{ct}_0[1] \cdot \texttt{ct}_1[1] \right\rceil \right]_q .
\end{aligned}$$

- $\texttt{Relin}(\texttt{ct}_{\texttt{BasicMul}}, \texttt{rlk})$: Writing $\texttt{ct}_{\texttt{BasicMul}} = (c_0, c_1, c_2)$, 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

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

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

$$c_{\texttt{Mul}} = (c_0', c_1') = \texttt{Relin}(\texttt{BasicMul}(\texttt{ct}_0, \texttt{ct}_1), \texttt{rlk}) .$$

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 $\mathtt{ct} = (c_0, c_1)$ 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 $\mathtt{ct}$ be a fresh ciphertext $\mathtt{ct} = \mathtt{Encrypt}(\mathtt{pk}, \mathsf{msg})$, then the invariant noise v of $\mathtt{ct}$ is bounded with very high probability by

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

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

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

with very high probability. Let $v^L_{\mathtt{Mul}}$ be an invariant noise after L multiplicative levels. If $L = 0$, then it follows from (2) and the additive noise growth after homomorphic addition that $\left\|v^0_{\mathtt{Mul}}\right\|^{\mathrm{can}} \in O\left(b^2 n\sqrt{n}/q\right)$. Computing $\left\|v^1_{\mathtt{Mul}}\right\|^{\mathrm{can}}$, one can notice that the first term of the right-hand side in (3) is dominant and thus $\left\|v^1_{\mathtt{Mul}}\right\|^{\mathrm{can}} \in O\left(bnv^0_{\mathtt{Mul}}\right)$, or $\left\|v^1_{\mathtt{Mul}}\right\|^{\mathrm{can}} \in O\left(b^3 n^2\sqrt{n}/q\right)$. By induction, we obtain

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

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

$$q \in \Omega\left(b^{L+2} n^{L+1} \sqrt{n}\right) . \tag{4}$$

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 $\overline{a} \in \mathbb{Z}/(b^{n/m}+1)\mathbb{Z}$ 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 $\overline{b} = \overline{\alpha}^m$ for some α. This assumption might seem too strong for the reader but there exist special forms of b such that $\overline{\alpha}$ 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 $\overline{\alpha}$, denoted $\overline{\beta}$. This implies that βX is a primitive $2m$-th root of unity in R_{X^m+b}, namely

$$(\overline{\beta} X)^m = \overline{\alpha}^{-m} X^m = \overline{b}^{-1} X^m = -1 .$$

Therefore, the map $\zeta_{2m} \mapsto \overline{\beta} X$ induces the following ring homomorphism

$$\mathbb{Z}[\zeta_{2m}] \to R_{X^m+b} : \qquad \sum_{i=0}^{m-1} a_i \zeta_{2m}^i \mapsto \sum_{i=0}^{m-1} \overline{a_i} \overline{\beta}^i X^i . \tag{5}$$

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 (2). 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 (5) 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 $\overline{\alpha} \cdot \zeta_{2m}$, which results in

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

As a result, the homomorphism defined by (5) 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 [23].

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 [19], 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 [22]. 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 1 in Appendix A.

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

$$|a|_\infty < \frac{b^{n/m}+1}{2}\,. \tag{6}$$

Let $a_i \in \mathbb{Z}[\zeta_{2m}]$ be output values of $\texttt{Pack}(\mathbf{z}_i)$ 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

$$|a_i|_\infty = \left|\left\lfloor \frac{\Delta}{m}\cdot \mathbf{\Sigma}^{-1}\mathbf{z}_i \right\rceil\right|_\infty \leq \Delta B + \frac{1}{2}\,. \tag{7}$$

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

$$|a_i|_\infty \leq V = \left(\frac{pm}{2} + \varepsilon\right) \cdot B + \frac{1}{2}\,.$$

For any a_i, a_j it holds $|a_i a_j|_\infty \leq mV^2$. 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 (6) we obtain that $m^{2^L-1}V^{2^L}$ must be smaller than $\frac{b^{n/m}+1}{2}$, which leads to

$$b \in \Omega\left(m^{\frac{m}{n}(2^{L+1}-1)} \cdot (pB)^{\frac{m}{n}\cdot 2^L}\right)\,.$$

Plugging this estimation into (4), 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

$$q \in \Omega\left(m^{\frac{m}{n}(2^{L+1}-1)(L+2)} \cdot (pB)^{\frac{m}{n}2^L(L+2)} \cdot n^{L+1}\sqrt{n}\right)\,. \tag{8}$$

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:

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

$$\begin{aligned} c_0 &= [\mathsf{msg} + p_0 \cdot u + e_0]_{q_L}\,, \\ c_1 &= [p_1 \cdot u + e_1]_{q_L}\,. \end{aligned}$$

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

$$\mathsf{msg}' = [c_0 + c_1 \cdot s]_{q_L}\,.$$

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 $\mathsf{msg}' = \mathsf{msg} + e'$ 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 $\mathbf{z}' = \mathtt{Unpack}(\mathtt{Decrypt}(\mathtt{Encrypt}(\mathtt{Pack}(\mathbf{z}))))$, then, following the reasoning of Lemma 1, we obtain

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

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

$$\Delta > mp\left(\frac{1}{2} + r\right), \tag{9}$$

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 $\mathtt{ct}_1$ and $\mathtt{ct}_2$ are given modulo q_ℓ. The basic homomorphic operations such as addition and multiplication are defined as follows.

- $\mathtt{Add}(\mathtt{ct}_0, \mathtt{ct}_1)$: Return

$$\mathtt{ct}_{\mathtt{Add}} = \left([\mathtt{ct}_0[0] + \mathtt{ct}_1[0]]_{q_\ell}, [\mathtt{ct}_0[1] + \mathtt{ct}_1[1]]_{q_\ell} \right).$$

- $\mathtt{BasicMul}(\mathtt{ct}_0, \mathtt{ct}_1)$: Return $\mathtt{ct}_{\mathtt{BasicMul}} = (c_0, c_1, c_2)$ where

$$\begin{aligned} c_0 &= [\mathtt{ct}_0[0] \cdot \mathtt{ct}_1[0]]_{q_\ell}, \\ c_1 &= [\mathtt{ct}_0[0] \cdot \mathtt{ct}_1[1] + \mathtt{ct}_0[1] \cdot \mathtt{ct}_1[0]]_{q_\ell}, \\ c_2 &= [\mathtt{ct}_0[1] \cdot \mathtt{ct}_1[1]]_{q_\ell}. \end{aligned}$$

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

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

- $\mathtt{Mul}(\mathtt{ct}_0, \mathtt{ct}_1, \mathtt{rlk})$: Return

$$\mathtt{ct}_{\mathtt{Mul}} = (c'_0, c'_1) = \mathtt{Relin}(\mathtt{BasicMul}(\mathtt{ct}_0, \mathtt{ct}_1), \mathtt{rlk}).$$

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

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

Note that if the input ciphertext $\texttt{ct}$ encrypts a plaintext msg, then $\texttt{ct}_{\texttt{Rescale}}$ is a valid encryption of $(q_{\ell'}/q_\ell) \cdot \mathsf{msg}$. 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.

6.2 Ciphertext size

Let msg_i be plaintext messages encoding complex vectors $\mathbf{z}_i \in \mathbb{C}^{m/2}$ with $|\mathbf{z}_i|_\infty \leq B$ 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 $|\Delta \mathbf{z}_i|_\infty + \|e\|^{\mathrm{can}} = \Delta B + m/2$ where e is the rounding error. Let $V = \Delta B + m/2$.

Assume that $q_\ell/q_{\ell-1} \simeq \Delta$ for any $\ell \in [L]$. After multiplication and rescaling we obtain a ciphertext encrypting a plaintext msg such that

$$\|\mathsf{msg}\|^{\mathrm{can}} \leq \frac{(V + \|E_e\|^{\mathrm{can}})^2}{\Delta} + \|E_r\|^{\mathrm{can}}$$

where E_e is the encryption noise and E_r is the noise introduced by $\texttt{Relin}$ and $\texttt{Rescale}$. Since $\|E_e\|^{\mathrm{can}}, \|E_r\|^{\mathrm{can}} \in O(n)$ according to [5, Lemmas 1-3], it follows from (9) that $\|\mathsf{msg}\|^{\mathrm{can}} \in O(\Delta B^2)$. Hence, after L multiplicative levels the canonical norm of a resulting plaintext satisfies $\|\mathsf{msg}\|^{\mathrm{can}} \in O\left(\Delta B^{2^L}\right)$. As $\|\mathsf{msg}\|^{\mathrm{can}}$ should be smaller than $q_0/2$, we obtain that $q_0 \in \Omega\left(\Delta B^{2^L}\right)$. Since rescaling decreases the ciphertext modulus L times to reach q_0, the initial ciphertext modulus $q_L \simeq q_0 \cdot \Delta^L$. Thus, $q_L \in \Omega\left(\Delta^{L+1} B^{2^L}\right)$ and (9) 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 (8) 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 $m/n \leq 1/4$;
- 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 [24]. 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

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 Maclaurin series of degree 8. These approximations are accurate up to 7 bits of binary precision in the following real intervals

$$\begin{array}{ll} [-2.1, 2.1] & \text{if } f(x) = \frac{1}{1+e^{-x}}; \\ [-\pi, \pi] & \text{if } f(x) = \sin(x); \\ [-2.3, 2.3] & \text{if } f(x) = e^x. \end{array}$$

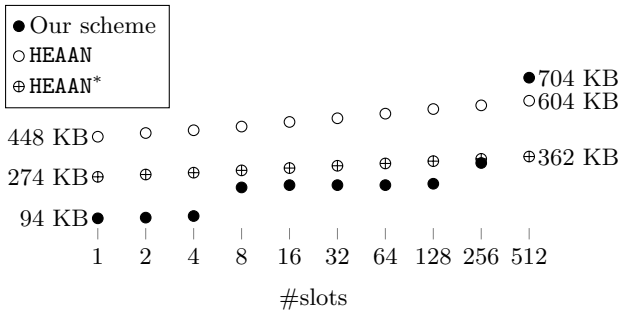

Fig. 1. Minimal ciphertext size needed to evaluate the logistic function in the interval $[-2.1, 2, 1]$ with approximation error 2^{-7} using `HEAAN`, `HEAAN`* and our scheme.

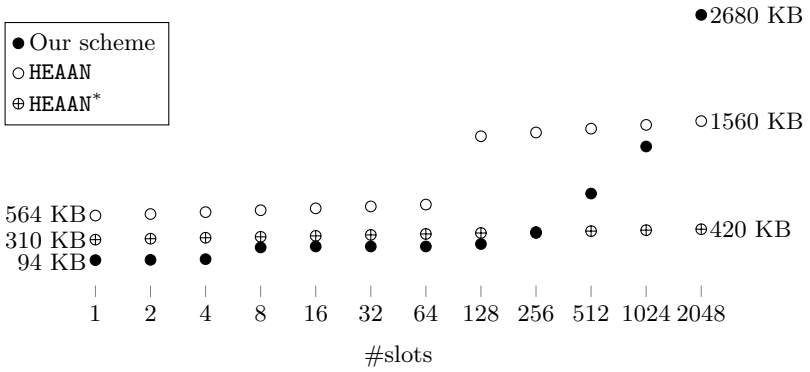

Fig. 2. Minimal ciphertext size needed to evaluate the sine in the interval $[-\pi, \pi]$ with approximation error 2^{-7} using `HEAAN`, `HEAAN`* and our scheme.

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 Figures 1-3. 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

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.

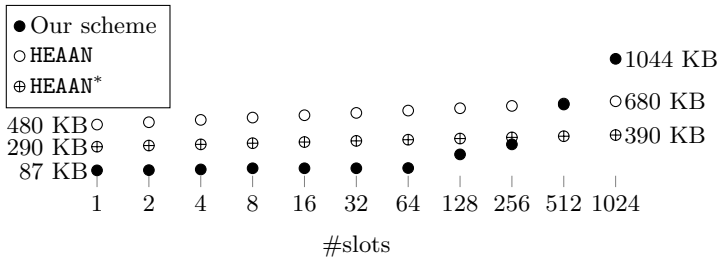

Fig. 3. Minimal ciphertext size needed to evaluate e^x in the interval $[-2.3, 2.3]$ with approximation error 2^{-7} using `HEAAN`, `HEAAN`* and our scheme.

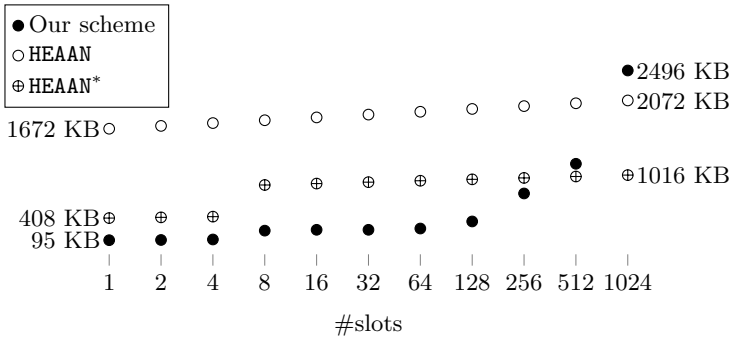

Fig. 4. Minimal ciphertext size needed to evaluate x^{16} in the interval $[-2.1, 2.1]$ with approximation error 2^{-7} using `HEAAN`, `HEAAN`* and our scheme.

As seen in Figures 4 and 5, 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.

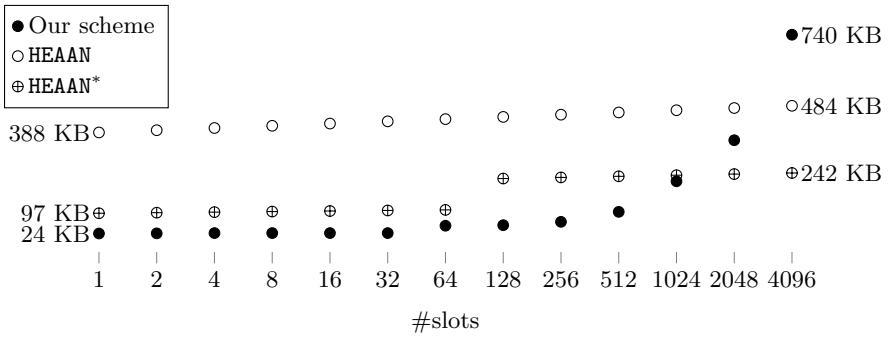

Fig. 5. Minimal ciphertext size needed to evaluate x^2 in the interval $(-2^{15}, 2^{15})$ with approximation error 2^{-7} using `HEAAN`, `HEAAN`* and our scheme.

Acknowledgements. The second author started this work while being an intern at Microsoft Research. He is also supported by a Junior Postdoctoral Fellowship from the Research Foundation – Flanders (FWO).

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. 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)
18. 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)
19. 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)
20. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. Foundations of secure computation **4**(11), 169–180 (1978)
21. 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 **21**(2), 120–126 (1978)
22. Shanks, D.: Five number-theoretic algorithms. In: Proceedings of the Second Manitoba Conference on Numerical Mathematics (1973)
23. Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. Des. Codes Cryptography **71**(1), 57–81 (Apr 2014)
24. The Sage Developers: SageMath, the Sage Mathematics Software System (Version 8.9) (2019), `https://www.sagemath.org`

A Examples of b

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

Table 1. 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 $\left\lfloor \log_2(b^{n/m}+1) \right\rfloor$. 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

#slots	Δ	n	$\log q$	b	ctxt size, KB
1	2^{16}	2^{12}	94	2	94
2	2^{17}	2^{12}	97	2^2	97
2^2	2^{18}	2^{12}	104	2^4	104
2^3	2^{19}	2^{13}	114	46	228
2^4(*)	2^{20}	2^{13}	119	102	238
2^5(*)	2^{21}	2^{13}	119	102	238
2^6	2^{22}	2^{13}	119	102	238
2^7	2^{23}	2^{13}	122	156	244
2^8	2^{24}	2^{13}	167	33116	334
2^9	2^{25}	2^{14}	176	50816	704

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

#slots	Δ	n	$\log q$	ctxt size, KB
1	2^{21}	2^{14}	112	448
2	2^{22}	2^{14}	116	464
2^2	2^{23}	2^{14}	119	476
2^3	2^{24}	2^{14}	123	492
2^4	2^{25}	2^{14}	128	512
2^5	2^{26}	2^{14}	132	528
2^6	2^{27}	2^{14}	137	548
2^7	2^{28}	2^{14}	142	568
2^8	2^{29}	2^{14}	146	584
2^9	2^{30}	2^{14}	151	604

Table 3. Encryption and packing parameters to compute the logistic function in the interval $[-2.1, 2.1]$ using `HEAAN`.

#slots	Δ	n	$\log q$	ctxt size, KB
1	2^{27}	2^{13}	137	274
2	2^{28}	2^{13}	142	284
2^2	2^{29}	2^{13}	146	292
2^3	2^{30}	2^{13}	151	302
2^4	2^{31}	2^{13}	156	312
2^5	2^{32}	2^{13}	161	322
2^6	2^{33}	2^{13}	166	332
2^7	2^{34}	2^{13}	171	342
2^8	2^{35}	2^{13}	176	352
2^9	2^{36}	2^{13}	181	362

Table 4. Encryption and packing parameters to compute the logistic function in the interval $[-2.1, 2.1]$ using `HEAAN` with the `FV` relinearization and without sparse secrets (`HEAAN`*).

#slots	Δ	n	$\log q$	b	ctxt size, KB
1	2^{23}	2^{12}	94	2	94
2	2^{24}	2^{12}	97	2^2	97
2^2	2^{25}	2^{12}	104	2^4	104
2^3	2^{26}	2^{13}	114	46	228
2^4(*)	2^{27}	2^{13}	119	102	238
2^5(*)	2^{28}	2^{13}	119	102	238
2^6	2^{29}	2^{13}	119	102	238
2^7	2^{30}	2^{13}	132	562	264
2^8	2^{31}	2^{13}	190	464890	380
2^9	2^{32}	2^{14}	199	706058	796
2^{10}	2^{33}	2^{14}	323	1131402968600	1292
2^{11}	2^{34}	2^{15}	335	2610566413416	2680

Table 5. Encryption and packing parameters to compute the sine in the interval $[-\pi, \pi]$ using our scheme. The (*) symbol indicates that the maximal number of slots supported by the plaintext space is 2^6.

#slots	Δ	n	$\log q$	ctxt size, KB
1	2^{27}	2^{14}	141	564
2	2^{28}	2^{14}	145	580
2^2	2^{29}	2^{14}	150	600
2^3	2^{30}	2^{14}	155	620
2^4	2^{31}	2^{14}	160	640
2^5	2^{32}	2^{14}	165	660
2^6	2^{33}	2^{14}	170	680
2^7	2^{34}	2^{15}	175	1400
2^8	2^{35}	2^{15}	180	1440
2^9	2^{36}	2^{15}	185	1480
2^{10}	2^{37}	2^{15}	190	1520
2^{11}	2^{38}	2^{15}	195	1560

Table 6. Encryption and packing parameters to compute the sine in the interval $[-\pi, \pi]$ using `HEAAN`.

#slots	Δ	n	$\log q$	ctxt size, KB
1	2^{30}	2^{13}	155	310
2	2^{31}	2^{13}	160	320
2^2	2^{32}	2^{13}	165	330
2^3	2^{33}	2^{13}	170	340
2^4	2^{34}	2^{13}	175	350
2^5	2^{35}	2^{13}	180	360
2^6	2^{36}	2^{13}	185	370
2^7	2^{37}	2^{13}	190	380
2^8	2^{38}	2^{13}	195	390
2^9	2^{39}	2^{13}	200	400
2^{10}	2^{40}	2^{13}	205	410
2^{11}	2^{41}	2^{13}	210	420

Table 7. Encryption and packing parameters to compute the sine in the interval $[-\pi, \pi]$ using `HEAAN` with the `FV` relinearization and without sparse secrets (`HEAAN`*).

#slots	Δ	n	$\log q$	b	ctxt size, KB
1	2^{18}	2^{12}	87	2	87
2	2^{19}	2^{12}	89	2^2	89
2^2	2^{20}	2^{12}	95	2^4	95
2^3(*)	2^{21}	2^{12}	105	96	105
2^4(*)	2^{22}	2^{12}	105	96	105
2^5	2^{23}	2^{12}	105	96	105
2^6	2^{24}	2^{12}	107	132	107
2^7	2^{25}	2^{13}	112	132	224
2^8	2^{26}	2^{13}	155	29254	310
2^9	2^{27}	2^{14}	163	43384	652
2^{10}	2^{28}	2^{14}	261	4080927980	1044

Table 8. Encryption and packing parameters to compute e^x in the interval $[-2.3, 2.3]$ using our scheme. The (*) symbol indicates that the maximal number of slots supported by the plaintext space is 2^5.

#slots	Δ	n	$\log q$	ctxt size, KB
1	2^{23}	2^{14}	120	480
2	2^{24}	2^{14}	125	500
2^2	2^{25}	2^{14}	130	520
2^3	2^{26}	2^{14}	135	540
2^4	2^{27}	2^{14}	140	560
2^5	2^{28}	2^{14}	145	580
2^6	2^{29}	2^{14}	150	600
2^7	2^{30}	2^{14}	155	620
2^8	2^{31}	2^{14}	160	640
2^9	2^{32}	2^{14}	165	660
2^{10}	2^{33}	2^{14}	170	680

Table 9. Encryption and packing parameters to compute e^x in the interval $[-2.3, 2.3]$ using `HEAAN`.

#slots	Δ	n	$\log q$	ctxt size, KB
1	2^{28}	2^{13}	145	290
2	2^{29}	2^{13}	150	300
2^2	2^{30}	2^{13}	155	310
2^3	2^{31}	2^{13}	160	320
2^4	2^{32}	2^{13}	165	330
2^5	2^{33}	2^{13}	170	340
2^6	2^{34}	2^{13}	175	350
2^7	2^{35}	2^{13}	180	360
2^8	2^{36}	2^{13}	185	370
2^9	2^{37}	2^{13}	190	380
2^{10}	2^{38}	2^{13}	195	390

Table 10. Encryption and packing parameters to compute e^x in the interval $[-2.3, 2.3]$ using `HEAAN` with the `FV` relinearization and without sparse secrets (`HEAAN`*).

#slots	Δ	n	$\log q$	b	ctxt size, KB
1	2^{25}	2^{12}	95	2	95
2	2^{26}	2^{12}	98	2^2	98
2^2	2^{28}	2^{12}	105	2^4	105
2^3	2^{29}	2^{13}	115	46	230
2^4(*)	2^{30}	2^{13}	121	120	242
2^5	2^{31}	2^{13}	121	120	242
2^6	2^{32}	2^{13}	130	412	260
2^7	2^{33}	2^{13}	180	137314	360
2^8	2^{34}	2^{14}	189	194162	756
2^9	2^{35}	2^{14}	294	37681177654	1176
2^{10}	2^{36}	2^{15}	312	150724710294	2496

Table 11. Encryption and packing parameters to compute x^{16} in the interval $[-2.1, 2.1]$ using our scheme. The (*) symbol indicates that the maximal number of slots supported by the plaintext space is 2^5.

#slots	Δ	n	$\log q$	ctxt size, KB
1	2^{38}	2^{15}	209	1672
2	2^{39}	2^{15}	214	1712
2^2	2^{40}	2^{15}	219	1752
2^3	2^{41}	2^{15}	224	1792
2^4	2^{42}	2^{15}	229	1832
2^5	2^{43}	2^{15}	234	1872
2^6	2^{44}	2^{15}	239	1912
2^7	2^{45}	2^{15}	244	1952
2^8	2^{46}	2^{15}	249	1992
2^9	2^{47}	2^{15}	254	2032
2^{10}	2^{48}	2^{15}	259	2072

Table 12. Encryption and packing parameters to compute x^{16} in the interval $[-2.1, 2.1]$ using `HEAAN`.

#slots	Δ	n	$\log q$	ctxt size, KB
1	2^{37}	2^{13}	204	408
2	2^{38}	2^{13}	209	418
2^2	2^{39}	2^{13}	214	428
2^3	2^{40}	2^{14}	219	876
2^4	2^{41}	2^{14}	224	896
2^5	2^{42}	2^{14}	229	916
2^6	2^{43}	2^{14}	234	936
2^7	2^{44}	2^{14}	239	956
2^8	2^{45}	2^{14}	244	976
2^9	2^{46}	2^{14}	249	996
2^{10}	2^{47}	2^{14}	254	1016

Table 13. Encryption and packing parameters to compute x^{16} in the interval $[-2.1, 2.1]$ using `HEAAN` with the `FV` relinearization and without sparse secrets (`HEAAN`*).

#slots	Δ	n	$\log q$	b	ctxt size, KB
1	2^{21}	2^{11}	48	2	24
2	2^{23}	2^{11}	49	2^2	24.5
2^2	2^{24}	2^{11}	51	2^4	25.5
2^3(*)	2^{25}	2^{11}	51	30	25.5
2^4(*)	2^{26}	2^{11}	51	30	25.5
2^5	2^{27}	2^{11}	51	30	25.5
2^6	2^{28}	2^{11}	52	44	52
2^7	2^{29}	2^{12}	53	74	54
2^8	2^{30}	2^{12}	66	2872	66
2^9	2^{31}	2^{12}	102	9976096	102
2^{10}	2^{32}	2^{13}	106	14108000	212
2^{11}	2^{33}	2^{13}	180	398065729535360	360
2^{12}	2^{34}	2^{14}	185	796131459067136	740

Table 14. Encryption and packing parameters to compute x^2 in the interval $(-2^{15}, 2^{15})$ using our scheme. The (*) symbol indicates that the maximal number of slots supported by the plaintext space is 2^5.

#slots	Δ	n	$\log q$	ctxt size, KB
1	2^{33}	2^{14}	97	388
2	2^{34}	2^{14}	99	396
2^2	2^{35}	2^{14}	101	404
2^3	2^{36}	2^{14}	103	412
2^4	2^{37}	2^{14}	105	420
2^5	2^{38}	2^{14}	107	428
2^6	2^{39}	2^{14}	109	436
2^7	2^{40}	2^{14}	111	444
2^8	2^{41}	2^{14}	113	452
2^9	2^{42}	2^{14}	115	460
2^{10}	2^{43}	2^{14}	117	468
2^{11}	2^{44}	2^{14}	119	476
2^{12}	2^{45}	2^{14}	121	484

Table 15. Encryption and packing parameters to compute x^2 in the interval $(-2^{15}, 2^{15})$ using `HEAAN`.

#slots	Δ	n	$\log q$	ctxt size, KB
1	2^{33}	2^{12}	97	97
2	2^{34}	2^{12}	99	99
2^2	2^{35}	2^{12}	101	101
2^3	2^{36}	2^{12}	103	103
2^4	2^{37}	2^{12}	105	105
2^5	2^{38}	2^{12}	107	107
2^6	2^{39}	2^{12}	109	109
2^7	2^{40}	2^{13}	111	222
2^8	2^{41}	2^{13}	113	226
2^9	2^{42}	2^{13}	115	230
2^{10}	2^{43}	2^{13}	117	234
2^{11}	2^{44}	2^{13}	119	238
2^{12}	2^{45}	2^{13}	121	242

Table 16. Encryption and packing parameters to compute x^2 in the interval $(-2^{15}, 2^{15})$ using `HEAAN` with the `FV` relinearization and without sparse secrets (`HEAAN`*).