IACR Communications in Cryptology
ISSN 3006-5496, Vol. 2, No. 1, 23 pages.
https://doi.org/10.62056/abe0iv7sf
Check for updates

Efficient Methods for Simultaneous Homomorphic Inversion

Jean Belo Klamti[1], M. Anwarul Hasan[2] and Koray Karabina[1,3]

[1] National Research Council Canada, Digital Technologies Research Center, Ottawa, Ontario, Canada
[2] University of Waterloo, Electrical and Computer Engineering, Waterloo, Ontario, Canada
[3] University of Waterloo, Combinatorics and Optimization Department, Waterloo, Ontario, Canada

Abstract. Efficient implementation of some privacy-preserving algorithms and applications rely on efficient implementation of homomorphic inversion. For example, a recently proposed homomorphic image filtering algorithm and the privacy-preserving body mass index (BMI) calculations repetitively use homomorphic inversion. In this paper, inspired by Montgomery's trick to perform simultaneous plaintext inversion, we tackle the simultaneous homomorphic inversion problem to compute s inverses simultaneously over ciphertexts. The advantage of Montgomery's trick for plaintext arithmetic is well-known. We first observe that the advantage can quickly vanish when homomorphic encryption is employed because of the increased depth of the circuits. Therefore, we propose three algorithms (Montgomery's trick and two other variants) that reduce the number of homomorphic inversions from s to 1 and that offer different levels of trade-offs between the number of multiplications and the circuit depth. We provide a theoretical complexity analysis of our algorithms and implement them using the CKKS scheme in the OpenFHE library. Our experiments show that, for some cases, the run time of homomorphic s-inversion can be improved up to 35% while in some other cases, regular inversion seems to outperform Montgomery-based inversion algorithms.

Keywords: Fully homomorphic encryption · Homomorphic inversion · Goldschmidt inversion · Montgomery's trick.

1 Introduction

Introduced by Rivest et al. [RAD+78], homomorphic encryption has been considered to be the holy grail of cryptography. It is a kind of public key encryption with an additional evaluation capability that allows computing over encrypted data without decrypting it. More formally, a public key homomorphic encryption scheme $\mathcal{E}$ with a message space $\mathcal{M}$ is a set of four probabilistic polynomial time (PPT) algorithms described as follows [Bra18, Gen09, Rot11]:

- $(\mathsf{sk}, \mathsf{pk}, \mathsf{evk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$: KeyGen is a key generation algorithm that takes as input a security parameter λ, and returns a tuple of secret key sk, public key pk, and evaluation key evk.

- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$: Enc is an encryption algorithm that takes as input the public key pk and a plaintext m; and outputs a ciphertext ct.

E-mail: jeanbelo.klamti@nrc-cnrc.gc.ca (Jean Belo Klamti), ahasan@uwaterloo.ca (M. Anwarul Hasan), koray.karabina@nrc-cnrc.gc.ca (Koray Karabina)

Received: 2025-01-14 Accepted: 2025-03-11

- $m \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: Dec is a decryption algorithm that takes as input the secret key sk and a ciphertext ct; and outputs a plaintext m.
- $\mathsf{ct} \leftarrow \mathsf{Eval}(\mathsf{evk}, f, \mathsf{ct}_1, ..., \mathsf{ct}_w)$: Eval is a homomorphic evaluation algorithm that takes as input the evaluation key evk, a circuit f, and w ciphertexts ct_i of plaintexts m_i; and outputs a ciphertext ct. Here, w must be a polynomial function of λ and the length of evaluated ciphertext ct must be polynomially bounded and independent of w.

Let $\mathcal{F}_w$ be the set of all circuits from $\mathcal{M}^w$ to $\mathcal{M}$. For a given subset of circuits $\mathcal{F} \subseteq \bigcup_{w \geq 1} \mathcal{F}_w$, a homomorphic encryption scheme $\mathcal{E}$ is said to be $\mathcal{F}$*-homomorphic* if it can correctly evaluate any circuit $f \in \mathcal{F}$. Specifically, for all $f \in \mathcal{F}$, the following holds:

$$\mathsf{Pr}[\mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(\mathsf{evk}, f, \mathsf{Enc}(\mathsf{pk}, m_1), \ldots, \mathsf{Enc}(\mathsf{pk}, m_w))) \neq f(m_1, \ldots, m_w)] = \mathsf{negl}(\lambda) \quad (1)$$

where λ is the chosen security parameter, w is a polynomial function of λ such that $f \in \mathcal{F}_w$, and negl is a neligible function. $\mathcal{E}$ is called a *fully homomorphic encryption* (FHE) when $\mathcal{F} = \bigcup_{w \geq 1} \mathcal{F}_w$ is the set of all circuits. Additionally, $\mathcal{E}$ is said to be a leveled FHE when $\mathcal{E}$ can correctly evaluate all circuits of depth at most D for some positive integer D.

1.1 An Overview of HE Schemes

It is noteworthy that for some time after the introduction of homomorphic encryption [RAD+78] by Rivest et al., the schemes that were designed [Gal02, BGN05, GM82, ElG85, Ben94, NS98, Pai99, BGN05, MGH10] supported only one of the addition or multiplication operations (i.e., *partially homomorphic* schemes) or both operations on a subset of circuits, such as unlimited number of aditions but only one multiplication, (i.e., *somewhat homomorphic* schemes). The first fully homomorphic encryption (FHE) scheme was proposed by Gentry in 2009 [Gen09]. His scheme is based on ideal lattices and relies on a key technique called *bootstrapping* that aims to decrease the error in the ciphertext after performing a circuit homomorphic evaluation. Since Gentry's scheme, significant improvements and new fully homomorphic encryption schemes have been developed based on different techniques or security assumptions. Among these, schemes based on ideal lattices and the approximate greatest common divisor (AGCD) are referred to as the first generation of fully homomorphic encryption schemes [SV10, GH11, SS11, SS10, VDGHV10, CMNT11, CNT12, CCK+13, NK15, CLT14, CS15].

The first generation FHE schemes experienced some efficiency and security challenges [CDPR16, CN12]. Research has led to the development of new FHE schemes based on the difficulty of lattice problems (learning with errors LWE, ring learning with errors RLWE, torus with errors TLWE, torus ring learning with errors TRLWE, number theory research unit NTRU) [BV11a, BV11b, LATV12, BLLN13, DS20, GSW13, BGV14, FV12, Bra12, GSW13, KGV15, BV14, ASP14, DM15, CGGI20, CKKS17]. These schemes yield three additional generations of FHE schemes: second, third and fourth generations.

Second-generation FHE schemes, based on LWE or RLWE, were introduced by Brakerski and Vaikuntanathan [BV11a, BV11b]. Designed for integer arithmetic (i.e., finite field and modular arithmetic), they support efficient packing and single instruction multiple data (SIMD) instructions for vector computations, making them ideal for processing large arrays of numbers. However, they are not suitable for evaluating circuits of large depth due to the high complexity of bootstrapping. Fan and Vercauteren (B/FV) [FV12] and Brakerski et al. (BGV) [BGV14] schemes are among widely adopted second-generation FHE schemes.

Third-generation FHE schemes, initiated by Gentry et al. [GSW13], address the challenge for bootstrapping and evaluating large depth circuits, excelling in bit-wise arithmetic (boolean circuits). However, they do not support batching and they are not as efficient as the second generation schemes for evaluating low depth circuits. TFHE by Chillotti et al. [CGGI20] is a widely adopted third-generation FHE scheme.

Fourth-generation schemes, introduced by Cheon et al. [CKKS17], are designed to handle real number arithmetic and they support batching. Similar to the second-generation schemes, they are not suitable for evaluating circuits of large depth due to the high complexity of bootstrapping. The CKKS scheme [CKKS17] is a widely adopted fourth-generation FHE scheme.

1.2 Motivation

The efficiency of homomorphic encryption based privacy-preserving applications rely on the efficient implementation of some specific functions. One of the natural functions to consider for homomorphic implementation is *inversion*, for at least two reasons.

1. Multiplicative inversion is the next simplest arithmetic operation after the addition and multiplication operations. Computing the (multiplicative) inverse of an element differs significantly depending on the domain. For $x \in \mathbb{Z}_p^*$, the inverse of x can be computed via modular exponentiation $x^{p-2} \bmod p$ using an arithmetic circuit of depth $\mathcal{O}(\log(p))$, whereas for $x \in \mathbb{R}$, an (approximate) inverse of x can be computed using (at least) three methods (Chebyshev [Che54], Newton-Rhapson [Rap02], and Goldschmidt [Gol64]) where the multiplicative depth of the circuit depends on the degree of the approximating polynomial, which sometimes can be related to the number of iterations in the algorithm. These methods have already been adapted for computing the inverse function over ciphertexts. OpenFHE [ABBB$^+$22] library uses the Chebyshev method [ABBB$^+$22] for the CKKS scheme, [CDSM15] provides a generic comparison of Newton-Rhapson and Goldschmidt methods, and [CKK$^+$19] implements the Goldschmidt inversion method using the CKKS scheme.

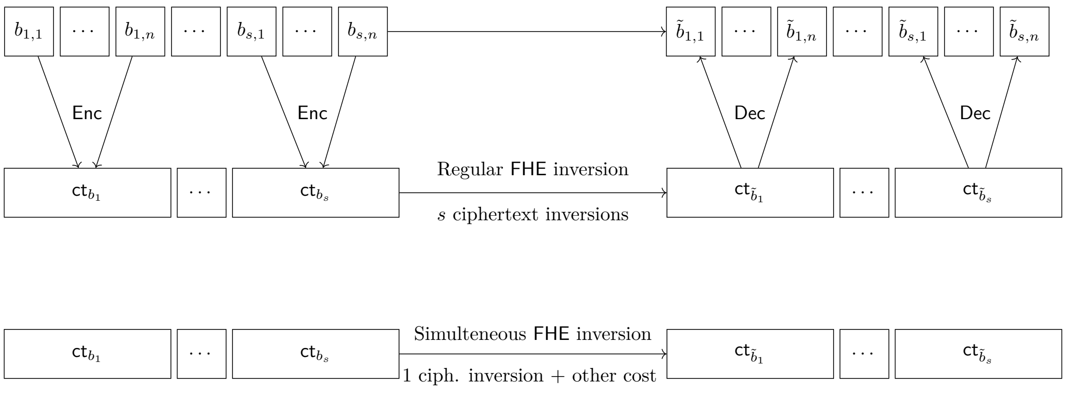
2. Inversion is one of the key operations in implementing privacy-preserving applications. For example, the homomorphic adaptive image filtering algorithm as proposed in [KK21] and the privacy-preserving body mass index (BMI) calculations [IIMP22a, IIMP22b] repetitively use homomorphic inversions.

In this paper, motivated by our discussion above and inspired by Montgomery's trick to perform simultaneous plaintext inversion [Mon87, Har08], we tackle the *simultaneous homomorphic inversion* problem: Suppose that ct_{b_i} is the encryption of a plaintext b_i, for $i = 1, ..., s$ and for some $s > 1$. How do we simultaneously compute $\mathsf{ct}_{\tilde{b}_i}$, where $\tilde{b}_i$ denotes the inverse of b_i in its domain, and $\mathsf{ct}_{\tilde{b}_i}$ is the encryption of $\tilde{b}_i$?

1.3 Reformulating our Problem for CKKS

To the best of our knowledge, the most motivating applications of homomorphic inversion concern computing inverses over the real numbers [CKK$^+$19, KK21, IIMP22a, IIMP22b]. Therefore, we focus on the fixed-point based approximate homomorphic encryption scheme CKKS [CKKS17] throughout the rest of this paper.

In CKKS, we have the plaintext space $\mathbb{Z}[X]/\langle X^N + 1\rangle$ and the ciphertext space $\mathbb{Z}_Q[X]/\langle X^N + 1\rangle \times \mathbb{Z}_Q[X]/\langle X^N + 1\rangle$. Here, N is a power of two, Q is typically a product of $(L + 1)$ primes q_i for some positive level parameter L, and the size of N and q_i are determined as a function of the security and precision parameters. In particular, for a fixed Q, increasing the depth of the circuit would decrease the fixed-point precision. CKKS allows encoding and encrypting $n = N/2$ real numbers $b_{i,j}$, for $j = 1, ..., n$, as a single ciphertext ct_{b_i}, where b_i denotes the vector $[b_{i,1}, ..., b_{i,n}]$. The homomorphic addition ($\oplus$) and multiplication ($\odot$) operations can be performed component-wise. More



$b_{1,1}$ ⋯ $b_{1,n}$ ⋯ $b_{s,1}$ ⋯ $b_{s,n}$
Enc
Enc
$\tilde{b}_{1,1}$ ⋯ $\tilde{b}_{1,n}$ ⋯ $\tilde{b}_{s,1}$ ⋯ $\tilde{b}_{s,n}$
Dec
Dec
ct_{b_1} ⋯ ct_{b_s}
Regular FHE inversion
s ciphertext inversions
$\mathsf{ct}_{\tilde{b}_1}$ ⋯ $\mathsf{ct}_{\tilde{b}_s}$
Simulteneous FHE inversion
1 ciph. inversion + other cost

Figure 1: An illustration of simultaneous homomorphic inversions. $(s \cdot n)$ elements can be packed into s ciphertexts using a suitable FHE scheme. A regular inversion would require s ciphertext inversions, whereas a simultaneous inversion method would aim for only 1 inversion. Simultaneous inversion would introduce other costs, such as additional multiplications.

precisely, decryptions of $\mathsf{ct}_{b_1} \oplus \mathsf{ct}_{b_2}$ and $\mathsf{ct}_{b_1} \odot \mathsf{ct}_{b_2}$ yield $[b_{1,1} + b_{2,1}, ..., b_{1,n} + b_{2,n}]$ and $[b_{1,1} \cdot b_{2,1}, ..., b_{1,n} \cdot b_{2,n}]$, respectively.

Therefore, using CKKS, $(s \cdot n)$ real numbers $b_{i,j}$, for $i = 1, ..., s$ and $j = 1, ..., n$, can be encoded and encrypted as s ciphertexts ct_{b_i}, and that we are interested in simultaneously computing $\mathsf{ct}_{\tilde{b}_i}$, where $\tilde{b}_i = [\tilde{b}_{i,1}, ..., \tilde{b}_{i,n}]$ and $\tilde{b}_{i,j}$ is an (approximate) inverse of $b_{i,j}$. Figure 1 presents an illustration of this problem. For future reference, we state this problem formally as follows:

Problem 1. *(Homomorphic s-inversion) Given the homomorphic encryption ct_{b_i} of the vector of plaintexts $b_i = [b_{i,1}, ..., b_{i,n}]$, compute $\mathsf{ct}_{\tilde{b}_i}$ for $i = 1, ..., s$. Here, $\mathsf{ct}_{\tilde{b}_i}$ is the encryption of the plaintext vector $\tilde{b}_i = [\tilde{b}_{i,1}, ..., \tilde{b}_{i,n}]$ such that $\tilde{b}_{i,j} = 1/b_{i,j}$, for $i = 1, ..., s$ and $j = 1, ..., n$.*

Remark 1. Problem 1 assumes that the underlying homomorphic encryption scheme is correct for all circuits. In the case of CKKS with approximate arithmetic, the problem statement should include $\tilde{b}_{i,j} \approx 1/b_{i,j}$.

1.4 Applications of Simultaneous Homomorphic Inversion

We revisit the homomorphic adaptive image filtering algorithm as proposed in [KK21]. Let I be an $(R \times C)$-image, where $R = r \cdot w$ and $C = c \cdot w$ for some positive integers r, c, and w. The image I contains $R \cdot C = r \cdot c \cdot w^2$ pixels. Let $I_i[j]$ denote the pixel value at row i and column j, for $0 \leq i < R$ and $0 \leq j < C$. I can be partitioned as the union of $(r \cdot c)$ pairwise disjoint kernel blocks, each containing w^2 pixels:

$$\mathsf{KB}(v \cdot w, h \cdot w) = \{I_{v \cdot w + \ell_v}[h \cdot w + \ell_h] : \; 0 \leq \ell_v, \; \ell_h < w\}, \tag{2}$$

where $0 \leq v < r$, $0 \leq h < c$. We call these kernel blocks $\mathsf{KB}(v \cdot w, h \cdot w)$ as *spanning kernel blocks*. If $r \cdot c \leq n$, then the image I can be encoded and encrypted using w^2 ciphertexts $\mathsf{ct}_0, ..., \mathsf{ct}_{w^2-1}$ such that the $k = (v \cdot c + h)$'th slot value $\mathsf{ct}_m[k]$ in the $m = (\ell_v \cdot w + \ell_h)$'th ciphertext corresponds to the encryption of the pixel value $I_{v \cdot w + \ell_v}[h \cdot w + \ell_h]$. In other words, the w^2-encrypted entries of each spanning kernel block $\mathsf{KB}(v \cdot w, h \cdot w)$ can be encoded in the $k = (v \cdot c + h)$'th slot of w^2-ciphertexts $\mathsf{ct}_0, ..., \mathsf{ct}_{w^2-1}$. This encoding allows to pass the locally adaptive Wiener filters through the encrypted spanning blocks simultaneously and the inversion operation dominates the cost because the Wiener filter updates the original pixel value $g(x, y)$ at the center of the spanning block to $f(x, y)$ using the formula

$$f(x,y) = m_\ell + \frac{\sigma_\ell^2}{\sigma_\ell^2 + \sigma_o^2}(g(x,y) - m_\ell), \tag{3}$$

where m_ℓ and σ_ℓ^2 denote the local mean and local variance within the kernel block; σ_o^2 denotes variance of overall noise, which can be considered as a constant. In order to update all of the pixel values in the centers of all of the kernel blocks using the Wiener filter, one can basically repeat the same process (w^2-1) more times. Before each repetition, one needs to transform spanning kernel blocks to other classes of kernel blocks by applying suitably chosen ciphertext rotations to a particular subset of $\mathsf{ct}_0, ..., \mathsf{ct}_{w^2-1}$. We omit the details and refer the reader to [KK21] for the full description of the process. The key point is that, processing an image I requires w^2 homomorphic inversions. An efficient simultaneous homomorphic inversion could potentially be used to process images by performing only 1 homomorphic inversion instead of w^2; see Figure 1. Typical choices for w include 3 and 5, and which would imply reductions from 9 and 25 homomorphic inversions to 1.

For another motivating application, we refer to [IIMP22a], where the authors propose a method for homomorphically counting elements with the same *property*. In the presentation of their paper [IIMP22b], the authors give the body mass index (BMI) of an individual as an example of a property. BMI can be calculated as

$$\mathsf{BMI} = \mathsf{weight}/(\mathsf{height})^2. \tag{4}$$

As a result, calculating BMI over an encrypted database of $(s \cdot n)$ users would require to encode and encrypt the weight and height of users using $(2 \cdot s)$ ciphertexts (i.e., $\mathsf{ct}^{(i)}_{\mathsf{weight}}$, $\mathsf{ct}^{(i)}_{\mathsf{height}}$) and to compute s ciphertext inversions before obtaining encrypted BMI values $\mathsf{ct}^{(i)}_{\mathsf{BMI}}$, for $i = 1, .., s$. Using simultaneous homomorphic inversion one could calculate encrypted BMI values by performing only 1 homomorphic inversion instead of s.

1.5 Contributions and Organization

In this paper, we tackle the homomorphic s-inversion problem (Problem 1). As we explain in Remark 2, a straightforward adaptation of the well-known Montgomery's trick may not be optimal because the depth of the underlying circuits increases as a function of s. Therefore, we propose three simultaneous inversion algorithms: FHE-Montgomery, FHE-relaxed Montgomery, and FHE-optimized Montgomery. The proposed algorithms reduce the number of homomorphic inversions from s to 1, at a cost of introducing additional ciphertext multiplications. Our algorithms provide different levels of trade-off between the computational complexity and the depth of circuit. We provide an in-depth complexity analysis of our algorithms and compare their efficiency against the efficiency of regular homomorphic inversion. We implement our algorithms using the CKKS scheme in the OpenFHE library and the Goldschmidt inversion. Our experiments show that the run time of homomorphic s-inversion can be improved up to 35%.

The rest of this paper is organized as follows: In Section 2, we recall Montgomery's trick for simultaneous plaintext inversion and the Goldschmidt homomorphic inversion method. In Section 3, we describe our proposed algorithms for solving Problem 1 and provide some theoretical complexity analysis. We report on our implementation results in Section 5 and conclude in Section 6.

2 Preliminaries

In this section, we review Montgomery's trick for simultaneous plaintext inversion and the Goldschmidt homomorphic inversion method.

2.1 Montgomery's Trick for $s > 1$ Inversions

We recall Montgomery's trick [Har08, Mon87] which aims to simultaneously compute the inverses $\tilde{b}_1, ..., \tilde{b}_s$ of $s > 1$ elements $b_1, ..., b_s$, using only 1 inversion, instead of s, at a cost of introducing multiplication operations. Since the cost of inversion is generally much more expensive than that of multiplication, the method is expected to offer significant efficiency gains. Montgomery's algorithm comprises two *passes*: a *forward* pass followed by a *backward* pass, which can be presented as in Figure 2. For more details, we refer to [Mon87, Har08].

```
Input : $b_1, ..., b_s$
Output : $\tilde{b}_1, ..., \tilde{b}_s$ such that $\tilde{b}_i = 1/b_i$
Forward Pass :
    Initiate $r_1 \leftarrow b_1$
    For $i = 2, .., s$
        Compute $r_i \leftarrow r_{i-1} b_i$
Backward Pass :
    Initiate $t_s \leftarrow \mathsf{Inv}(r_s)$
    For $i = s, .., 2$ do:
        Compute $\tilde{b}_i \leftarrow t_i r_{i-1}$
        Compute $t_{i-1} \leftarrow t_i b_i$
    Set $\tilde{b}_1 \leftarrow t_1$
```

Figure 2: Montgomery's trick can compute the inverses of s elements at a cost of performing 1 inversion and $3(s-1)$ multiplications. See [Mon87, Har08] for more details.

Proposition 1. *[Mon87, Har08] The multiplicative inverses of s elements can be computed using $3(s-1)$ multiplications and one inversion using Montgomery's trick.*

Proof. See Section 5 in [Har08]. □

2.2 FHE-Goldschmidt Inversion

As motivated in Section 1.3, we focus on computing inverses over the real numbers, and so we fix CKKS as the underlying encryption scheme. As noted in Section 1, Chebyshev, Newton-Raphson, and Goldschmidt algorithms have been adapted for ciphertext inversion using CKKS. Our analysis in this paper is based on the Goldschmidt inversion algorithm because it is commonly implemented in literature [CDSM15, CKK$^+$19, KK21], and that its simple and iterative nature offers explicit and reasonable trade-offs between the depth of circuits and accuracy of the approximation. The main idea of the Goldschmidt inversion method is to use approximation

$$\frac{1}{x} = \frac{1}{1-(1-x)} = \prod_{i=0}^{\infty}(1+(1-x)^{2^i}) \approx \prod_{i=0}^{d}(1+(1-x)^{2^i}), \tag{5}$$

for $x \in (0, 2)$, which can be implemented using an iterative square-and-multiply method as shown in Figure 3.

```
Input : $b \in (0,2)$ and $d \in \mathbb{N}$.
Output : $\tilde{b} \approx 1/b$
        Compute $a_0 \leftarrow 2 - b$ and $b_0 \leftarrow 1 - b$
        For $i = 0, .., d-1$, compute:
            $b_{i+1} \leftarrow {b_i}^2$ and $a_{i+1} \leftarrow a_i(1 + b_{i+1})$
        Return $a_d$
```

Figure 3: FHE-Goldschmidt Inversion [CDSM15, CKK$^+$19] can approximate inverses in d iterations with a circuit of depth $(d+1)$ and exponential accuracy in d.

Proposition 2. *[CKK$^+$19] For a given integer $m > 0$ and $x \in [2^{-m}, 1)$, computing the homomorphic inverse $\mathsf{ct}_{\tilde{x}}$ of the ciphertext $\mathsf{ct}_x = \mathsf{Enc}(\mathsf{pk}, x)$ for an error bound of $2^{-\alpha}$ using FHE-Goldschmidt inversion algorithm, will require only $d = \Theta(\log \alpha + m)$ iterations. Moreover, it will cost d homomorphic additions, 2 homomorphic subtractions, d homomorphic squares, and d homomorphic multiplications, where the depth of the circuit is $d+1$.*

Proof. See Lemma 1 in [CKK$^+$19]. □

3 Homomorphic s-Inversion

A naive method for solving Problem 1 is to compute each of the s inverses separately. In this section, we propose three algorithms to solve the homomorphic s-inversion problem efficiently, by performing only one inversion instead of s. The first algorithm FHE-*Montgomery* is a direct adaptation of Montgomery's trick. The next two algorithms, FHE-*relaxed Montgomery* and FHE-*optimized Montgomery* (see Figure 5 and Figure 7 for plaintext versions) reduce the circuit depth, which is a key efficiency factor in the implementation of homomorphic encryption algorithms. FHE-relaxed Montgomery achieves the minimum depth among the three algorithms but increases the number of multiplications with respect to the base FHE-Montgomery algorithm. FHE-optimized Montgomery provides a balance by reducing the circuit depth while keeping the number of multiplications same with respect to FHE-Montgomery.

3.1 FHE-Montgomery Inversion

Following Montgomery's trick, FHE-Montgomery inversion is divided into two parts: the forward and backward pass. For simultaneously inverting $s > 1$ ciphertexts ct_{b_1},...,ct_{b_s}, the forward pass computes a sequence of product of ciphertexts $\mathsf{ct}_{r_i} = \mathsf{ct}_{b_1} \odot \cdots \odot \mathsf{ct}_{b_i}$ for $i = 1, ..., s$. As a first step in the backward pass, $\mathsf{ct}_{t_s} = \mathsf{ct}_{\tilde{r_s}}$ (the ciphertext that corresponds to the inverse of $r_s = b_1 \cdots \cdot b_s$) is computed using a homomorphic inversion algorithm. Then, $\mathsf{ct}_{\tilde{b}_i}$ (the ciphertext that corresponds to the inverse of b_i) is computed via $\mathsf{ct}_{\tilde{b}_i} = \mathsf{ct}_{t_i} \odot \mathsf{ct}_{r_{i-1}}$ for $i = s, ..., 2$, where ct_{t_i} can be computed using ct_{t_s} and $\mathsf{ct}_{t_{i-1}} = \mathsf{ct}_{t_i} \odot \mathsf{ct}_{b_i}$. Finally, $\mathsf{ct}_{\tilde{b}_1}$ comes for free as it is the same as ct_{t_1}. More formally, we state the computational and depth complexity of FHE-Montgomery in Theorem 1.

Theorem 1. *Let $s \in \mathbb{N}$. FHE-Montgomery yields a circuit of depth $\lceil \log(s) \rceil + s + \mathsf{Depth}_{inv} - 1$ that solves the homomorphic s-inversion problem at a cost of performing $3(s-1)$ homomorphic multiplications and 1 homomorphic inversion, where Depth_{inv} is the depth of the circuit corresponding to the underlying homomorphic inversion algorithm.*

Proof. In the forward pass, computing ct_{r_i} for $i = 1, ..., s$ requires $(s-1)$ homomorphic multiplications. In the backward pass, after performing 1 homomorphic inversion, homomorphic products $\mathsf{ct}_{\tilde{b}_i} = \mathsf{ct}_{t_i} \odot \mathsf{ct}_{r_{i-1}}$ and $\mathsf{ct}_{t_{i-1}} = \mathsf{ct}_{t_i} \odot \mathsf{ct}_{b_i}$ are computed for $i = s, ..., 2$. Hence, the backward pass requires 1 homomorphic inversion and $2(s-1)$ homomorphic multiplications, and we compute the overall cost of the algorithm as 1 homomorphic inversion and $3(s-1)$ homomorphic multiplications. For the circuit depth, one can see that computing ct_{r_i} requires $\lceil \log(s) \rceil$ sequential multiplications if a binary tree is used. Computing ct_{t_s} increases the depth by Depth_{inv}. Next, running the $(s-1)$ iterations in the algorithm increases the depth by $(s-1)$. Hence, we conclude that the depth of the circuit is $\mathsf{Depth}_{inv} + \lceil \log(s) \rceil + s - 1$. □

Corollary 1. *Let $b_i \in [\epsilon, 1)$ for some $\epsilon \geq 0$ and $i = 1, ..., s$. The* FHE-*Montgomery inversion algorithm can employ* FHE-*Goldschmidt and compute homomorphic inverses $\mathsf{ct}_{\tilde{b}_i}$ of the ciphertexts $\mathsf{ct}_{b_i} = \mathsf{Enc}(\mathsf{pk}, x)$ at a cost of $3(s-1) + 2d$ homomorphic multiplications with a depth $D = \lceil \log(s) \rceil + s + d$ circuit, where $d = \lceil \log \alpha - s \log \epsilon \rceil$ is the number of iterations in the Goldschmidt algorithm and the error is bounded by $2^{-\alpha}$.*

Proof. The key observation is that for $b_i \in [\epsilon, 1)$, we have, $\prod_{i=1}^{s} b_i \in [\epsilon^s, 1)$ and so based on Proposition 2, we need to set the number of Goldschmidt iterations to $\lceil \log(\alpha) - \log(\epsilon^s) \rceil = \lceil \log(\alpha) - s \log(\epsilon) \rceil$. The rest of the proof follows from Proposition 2 and Theorem 1. □

Remark 2. Let $b_i \in [\epsilon, 1)$ for some $\epsilon \geq 0$ and $i = 1, ..., s$. Based on Proposition 2, the Goldschmidt-based homomorphic inversion algorithm can compute homomorphic inverses of ct_{b_i}, for an error bound of $2^{-\alpha}$, at a cost of $2sd$ homomorphic multiplications using a depth $D = d + 1$ circuit, where $d = \lceil \log \alpha - \log \epsilon \rceil$ is the number of iterations in the Goldschmidt algorithm. For $\alpha = 4$ and $\epsilon = 1 - 1/2^4$, we get $d = 3$, $D = 4$, and that yield 6 multiplications per inverse. Under the same setting, based on Corollary 1, FHE-Montgomery inversion algorithm (calling Goldschmidt inversion as a subroutine) can compute homomorphic inverses of ct_{b_i}, for an error bound of $2^{-\alpha}$, at a cost of $(3(s-1) + 2d)$ homomorphic multiplications using a depth $D = \lceil \log s \rceil + s + d$ circuit, where $d = \lceil \log \alpha - s \log \epsilon \rceil$ is the number of iterations in the Goldschmidt algorithm. For $\alpha = 4$, $\epsilon = 1 - 1/2^4$, and $s = 2, ..., 8$, we get $d = 3$, (s, D) pairs as $(2, 6)$, $(3, 8)$, $(4, 11)$, $(5, 11)$, $(6, 12)$, $(7, 13)$, $(8, 14)$, and that yield 4.5, 4.0, 3.75, 3.6, 3.5, 3.44, 3.38 multiplications per inverse, respectively. These results are also illustrated in the first 2 plots in Figure 4. Observe that while FHE-Montgomery reduces the number of homomorphic multiplications per inverse, the depth of the underlying circuits increases. Therefore, it is unclear if FHE-Montgomery simultaneous inversion outperforms the regular Goldschmidt-based inversion. Our experiments show that (see the last plot in Figure 4) the choice of $s = 2$ is optimal for FHE-Montgomery, and it loses its advantage for $s \geq 4$ when $\alpha = 4$ and $\epsilon = 1 - 1/2^4$. This motivates us to propose variants of FHE-Montgomery in Sections 3.2 and 3.3, namely FHE-relaxed Montgomery and FHE-optimized Montgomery. As we will see, our proposed variants offer trade-offs between the number of multiplications and circuit depth, and that they improve the runtime of FHE-Montgomery. Figure 4 illustrates these trade-offs and improvements for a particular case. More detailed comparisons will follow in the next sections.

3.2 FHE-Relaxed Montgomery Inversion

In FHE-relaxed Montgomery, the backward pass uses only $\mathsf{ct}_{\tilde{r}_s}$ as opposed to the use of ct_{t_i} for $i = 1, ..., s$ in FHE-Montgomery where $\mathsf{ct}_{t_i} = \mathsf{ct}_{\tilde{r}_i}$. Moreover, in the backward pass, FHE-relaxed Montgomery allows to computation $\mathsf{ct}_{\tilde{b}_i}$ in parallel, which does not increase the circuit depth. More formally, we state the computational and depth complexity of FHE-relaxed Montgomery in Theorem 2 and illustrate its plaintext version in Figure 5.

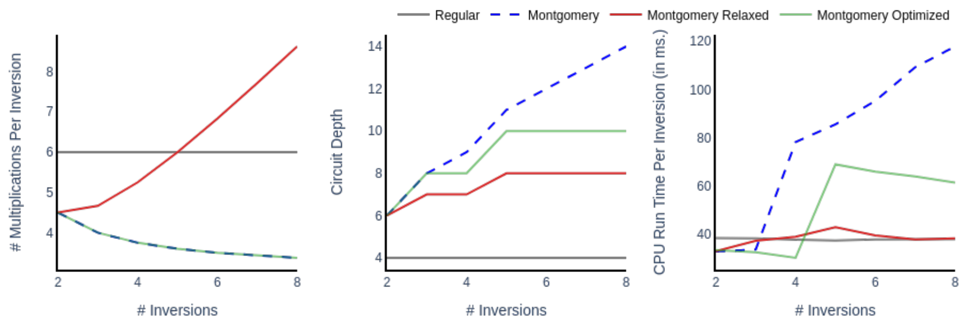


Figure 4: A comparison of FHE-regular, FHE-Montgomery, FHE-relaxed Montgomery, and FHE-optimized Montgomery inversion algorithms with respect to the number of multiplications per inversion, circuit depth, and runtime per inversion. All of the algorithms call FHE-Goldschmidt as a subroutine, plaintexts b_i belong to $[\epsilon, 1)$, and the approximation error is bounded by $1/2^{\alpha}$, where $\epsilon = 1 - 1/2^4$ and $\alpha = 4$.

```
Input : $b_1, ..., b_s$
Output : $\tilde{b}_1, ..., \tilde{b}_s$ such that $\tilde{b}_i = 1/b_i$
    1. Compute $r_s \leftarrow b_1 \cdot ... \cdot b_s$
    2. Compute $\tilde{r}_s = 1/r_s$
    3. For $i = 1, .., s$:
           Compute $t_i \leftarrow b_1 \cdot \ ... \ \cdot b_{i-1} \cdot b_{i+1} \cdot \ ... \ \cdot b_s$
           Compute $\tilde{b}_i \leftarrow \tilde{r}_s \cdot t_i$
    4. Return $\tilde{b}_i$ for $i = 1, ..., s$
```

Figure 5: Plaintext version of the FHE-relaxed Montgomery inversion algorithm. It reduces the depth of FHE-Montgomery inversion circuit by $(s - 2)$ at a cost of increasing the number of multiplications by $(s^2 - 3s + 2)$.

Theorem 2. *Let $s \in \mathbb{N}$. FHE-relaxed Montgomery yields a circuit of depth $\lceil \log(s) \rceil +$ $\mathsf{Depth}_{inv} + 1$ that solves the homomorphic s-inversion problem at a cost of performing 1 homomorphic inversion and $(s^2 - 1)$ homomorphic multiplications, where Depth_{inv} is the depth of the circuit corresponding to the underlying homomorphic inversion algorithm.*

Proof. Based on inspecting the plaintext version of FHE-relaxed Montgomery in Figure 5, computing ct_{r_s} requires $(s - 1)$ homomorphic multiplications. Computing ct_{t_i} and $\mathsf{ct}_{\tilde{b}_i}$ requires $(s-2)$ and 1 homomorphic multiplications respectively, and so $s(s-1)$ homomorphic multiplications are performed in the for loop. Hence, FHE-relaxed Montgomery requires $(s^2 - 1)$ homomorphic multiplications and 1 homomorphic inversion. For the circuit depth, one can see that computing $\mathsf{ct}_{r_s} = \mathsf{ct}_{b_1} \odot \cdots \odot \mathsf{ct}_{b_s}$ requires $\lceil \log(s) \rceil$ sequential multiplications if a binary tree is used. Computing $\mathsf{ct}_{\tilde{r}_s}$ increases the depth by Depth_{inv}, and as a result each $\mathsf{ct}_{\tilde{b}_i}$ can be computed with at most $\lceil \log(s) \rceil + \mathsf{Depth}_{inv} + 1$ sequential multiplications, as required. □

Corollary 2. *Let $b_i \in [\epsilon, 1)$ for some $\epsilon \geq 0$ and $i = 1, ..., s$. The* FHE*-relaxed Montgomery inversion algorithm can employ* FHE*-Goldschmidt and compute homomorphic inverses* $\mathsf{ct}_{\tilde{b_i}}$ *of the ciphertexts* $\mathsf{ct}_{b_i} = \mathsf{Enc}(\mathsf{pk}, x)$ *at a cost of* $(s^2 + 2d - 1)$ *homomorphic multiplications with a depth* $D = \lceil \log(s) \rceil + d + 2$ *circuit, where* $d = \lceil \log \alpha - s \log \epsilon \rceil$ *is the number of iterations in the Goldschmidt algorithm and the error is bounded by* $2^{-\alpha}$.

Proof. The proof is similar to the proof of Corollary 1 and follows from Proposition 2 and Theorem 2. ☐

3.3 FHE-Optimized Montgomery Inversion

FHE-relaxed Montgomery reduced the depth of FHE-Montgomery by $(s-2)$ but introduced $(s^2 - 3s + 2)$ additional multiplications. In this section, we propose FHE-optimized Montgomery that reduces the depth of FHE-Montgomery by $(s - \lceil \log(s) \rceil - 1)$. The reduction is more conservative than the FHE-relaxed Montgomery reduction but the advantage is that FHE-optimized Montgomery does not introduce additional multiplications. FHE-optimized Montgomery uses binary trees and the method is illustrated in Figure 6 for $s = 8$. More formally, we state the computational and depth complexity of FHE-optimized Montgomery in Theorem 3 and illustrate its plaintext version in Figure 7.

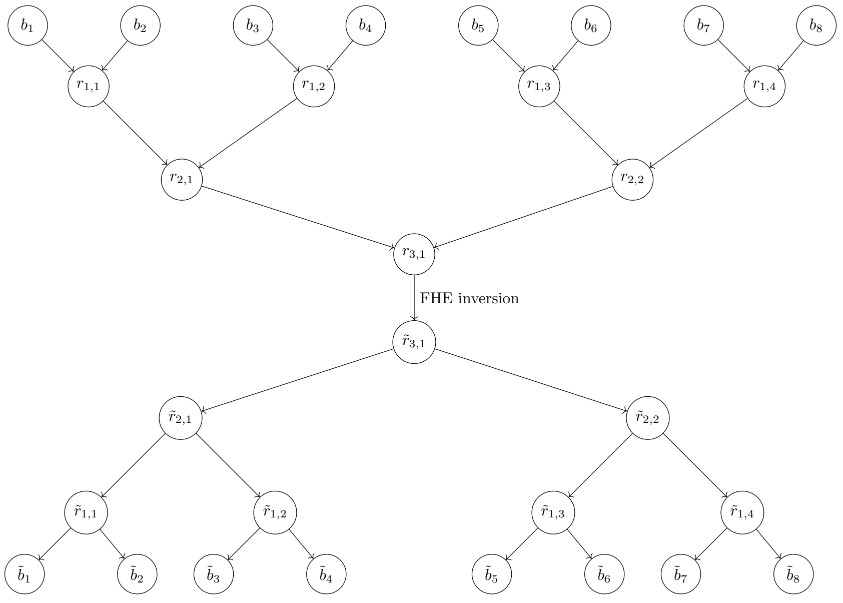


Figure 6: An illustration of optimized Montgomery for $s = 8$. The elements of nodes in the bottom half of the tree are inverses of those in the top half of the tree. The algorithm computes 8 inverses at the cost of performing 21 multiplications and 1 inversion.

Theorem 3. *Let $s \in \mathbb{N}$.* FHE*-optimized Montgomery yields a circuit of depth* $2\lceil \log(s) \rceil + \mathsf{Depth}_{inv}$ *that solves the homomorphic s-inversion problem at a cost of performing* $3(s-1)$ *homomorphic multiplications and* 1 *homomorphic inversion, where* Depth_{inv} *is the depth of the circuit corresponding to the underlying homomorphic inversion algorithm.*

Proof. Based on inspecting the plaintext version of FHE-optimized Montgomery in Figure 7, building the tree in the first step in the algorithm requires $(s-1)$ homomorphic multiplications. After performing the homomorphic inversion in step 2, $2s - 2$ more homomorphic multiplications are required in the algorithm. Hence, FHE-optimized Montgomery

```
Input : $b_1, ..., b_s$ and $s = 2^k$
Output : $\tilde{b}_1, ..., \tilde{b}_s$ such that $\tilde{b}_i = 1/b_i$
  1. Compute a binary tree leaves $r_{ij}$ as follows:
     (a) Compute $r_{1,i} \leftarrow b_{2i-1} \cdot b_{2i}$ for $i = 1, ..., 2^{k-1}$
     (b) For $i = 2$ to $k - 1$ do:
             For $j = 1$ to $2^{k-i}$, do:
                 $r_{i,j} \leftarrow r_{i-1,2j-1} \cdot r_{i-1,2j}$
     (c) $r_{k,1} \leftarrow r_{k-1,1} \cdot r_{k-1,2}$
  2. Compute $\tilde{r}_{k,1} \leftarrow 1/r_{k,1}$.
  3. For $i = k - 1$ to 1 do:
         For $j = 1$ to $2^{k-1-j}$, compute:
             $\tilde{r}_{i,2j-1} \leftarrow \tilde{r}_{i+1,j} \cdot r_{i,2j}$ and $\tilde{r}_{i,2j} \leftarrow \tilde{r}_{i+1,j} \cdot r_{i,2j-1}$.
  4. For $i = 1$ to $2^{k-1}$ compute:
         $\tilde{b}_{2i} \leftarrow \tilde{r}_{1,i} \cdot b_{2i-1}$ and $\tilde{b}_{2i-1} \leftarrow \tilde{r}_{1,i} \cdot b_{2i}$
```

Figure 7: Plaintext version of the FHE-optimized Montgomery algorithm. FHE-optimized Montgomery reduces the depth of FHE-Montgomery inversion circuit by $(s - \lceil \log(s) \rceil - 1)$ while keeping the number of multiplications same.

requires $3(s - 1)$ homomorphic multiplications and 1 homomorphic inversion. Observing the structure of the underlying binary tree, we can conclude that the depth of the circuit is $2\lceil \log(s) \rceil + \mathsf{Depth}_{inv}$. $\square$

Corollary 3. *Let $b_i \in [\epsilon, 1)$ for some $\epsilon \geq 0$ and $i = 1, ..., s$. The FHE-optimized Montgomery inversion algorithm can employ FHE-Goldschmidt and compute homomorphic inverses $\mathsf{ct}_{\tilde{b}_i}$ of the ciphertexts $\mathsf{ct}_{b_i} = \mathsf{Enc}(\mathsf{pk}, b_i)$ at a cost of $3(s - 1) + 2d$ homomorphic multiplications with a depth $D = 2\lceil \log(s) \rceil + d + 1$ circuit, where $d = \lceil \log \alpha - s \log \epsilon \rceil$ is the number of iterations in the Goldschmidt algorithm and the error is bounded by $2^{-\alpha}$.*

Proof. The proof is similar to the proof of Corollary 1 and follows from Proposition 2 and Theorem 3. $\square$

Corollary 4. *For $s = 2$, FHE-Montgomery, FHE-relaxed Montgomery, and FHE-optimized Montgomery have the same circuit depth and require the same number of multiplications.*

Proof. Based on Corollary 1 and 2, FHE-relaxed Montgomery reduces the depth of FHE-Montgomery inversion circuit by $(s-2)$ at a cost of increasing the number of multiplications by $(s^2 - 3s + 2)$. Therefore, when $s = 2$, they have the same circuit depth and require the same number of multiplications. Based on Corollary 1 and 3, FHE-optimized Montgomery reduces the depth FHE-Montgomery inversion circuit by $(s - \lceil \log(s) \rceil - 1)$ while keeping the number of multiplications same. Therefore, for $s = 2$, they have the same circuit depth and require the same number of multiplications. $\square$

4 A Comparison of FHE Inversion Algorithms

In the Goldschmidt-based FHE inversion algorithm, inverses are computed separately, the depth of the circuit is exactly that of the FHE-Goldschmidt circuit, and the computational complexity is given by s times the cost of FHE-Goldschmidt operations (see Remark 2). In particular, the depth of the regular homomorphic s-inversion circuit is independent of s and is only a function of ϵ and α. The complexities of our proposed algorithms are summarized in Corollary 1, 2, and 3. While the number of multiplications is reduced in FHE-Montgomery, FHE-relaxed Montgomery, and FHE-optimized Montgomery, the depths of the circuits increase as a function of s and α. FHE-Montgomery offers the least number of multiplications, but its circuit requires the largest depth. FHE-relaxed Montgomery minimizes the circuit depth but maximizes the number of multiplications. FHE-optimized Montgomery has the same number of multiplications as FHE-Montgomery and its circuit depth is smaller. Table 1 compares the complexity of our proposed algorithms and the regular algorithm in terms of the circuit depth (Depth) and the number of ciphertext multiplications for $s = 2, 3, 4, 5$; $m = 2, 3, 4$ and $\alpha = 4, 8$. Here, s is the number of ciphertext inversions to compute; the input plaintexts belong to the interval $[1 - 1/2^m, 1)$; α is the parameter that controls the error bound; and Depth is a function of d, which is the number of iterations in the Goldschmidt algorithm (also see Corollary 1, 2, and 3). In short, the two variants of FHE-Montgomery improve the circuit depth at a cost of possibly increasing the number of multiplications. To our knowledge, it is not known how to compare multiplication costs at different depths. Therefore we cannot theoretically conclude which inversion algorithm would offer the best time complexity. Instead, we compare their performance based on our implementation in Section 5. Also Figure 8 extends the earlier comparison of our proposed algorithms as illustrated in Figure 4.

It can be observed from Figure 8 that in the 12 of the 18 cases (except the 6 cases in the third and fourth rows in Figure 8), there is always at least one Montgomery based inversion algorithm and a choice of grouping ciphertexts for simultaneous inversion such that the Montgomery method outperforms regular inversion with respect to the per inversion cost, and hence minimizes the overall cost. For example, the last plot in the last row in Figure 8 shows that if the task is to compute 8 ciphertext inversions under $\alpha = 30$ and $m = 4$ parameters, then running FHE-relaxed Montgomery with $s = 4$ (hence computing simultaneous 4-inversions twice) is the best option with minimal per inversion cost and hence with minimal overall cost. Similarly, the first plot in the second row in Figure 8 shows that if the task is to compute 6 ciphertext inversions under $\alpha = 8$ and $m = 2$ parameters, then running FHE-Montgomery with $s = 2$ (hence computing simultaneous 2-inversions three times) is the best option with minimal per inversion cost and hence with minimal overall cost. Montgomery based algorithms lose their advantage in the 6 of the 18 cases because FHE-regular inversion yields the minimal per inversion costs according to the third and fourth rows in Figure 8.

5 Implementation Results

As motivated in Section 1.3, we consider implementing our algorithms using CKKS and choose Goldschmidt inversion as a subroutine in our algorithms. The approximated nature of the ciphertext computations requires choosing the number of iterations parameter d in Goldschmidt as a function of the parameter α that controls the error bound (or accuracy); see Proposition 2.

Selecting CKKS parameters and algorithms For the CKKS parameters, we select the scaling modulus Δ over the lower bound given by Cheon et al. [CKKS17, Lemma 1], i.e., $\Delta > N + 2B_{enc}$, where N is the ring dimension, $B_{enc} = 8\sigma N\sqrt{2} + 6\sigma\sqrt{N} + 16\sigma\sqrt{hN}$ is the encryption noise, with h the Hamming weight and $\sigma = 3.19$ the standard deviation.

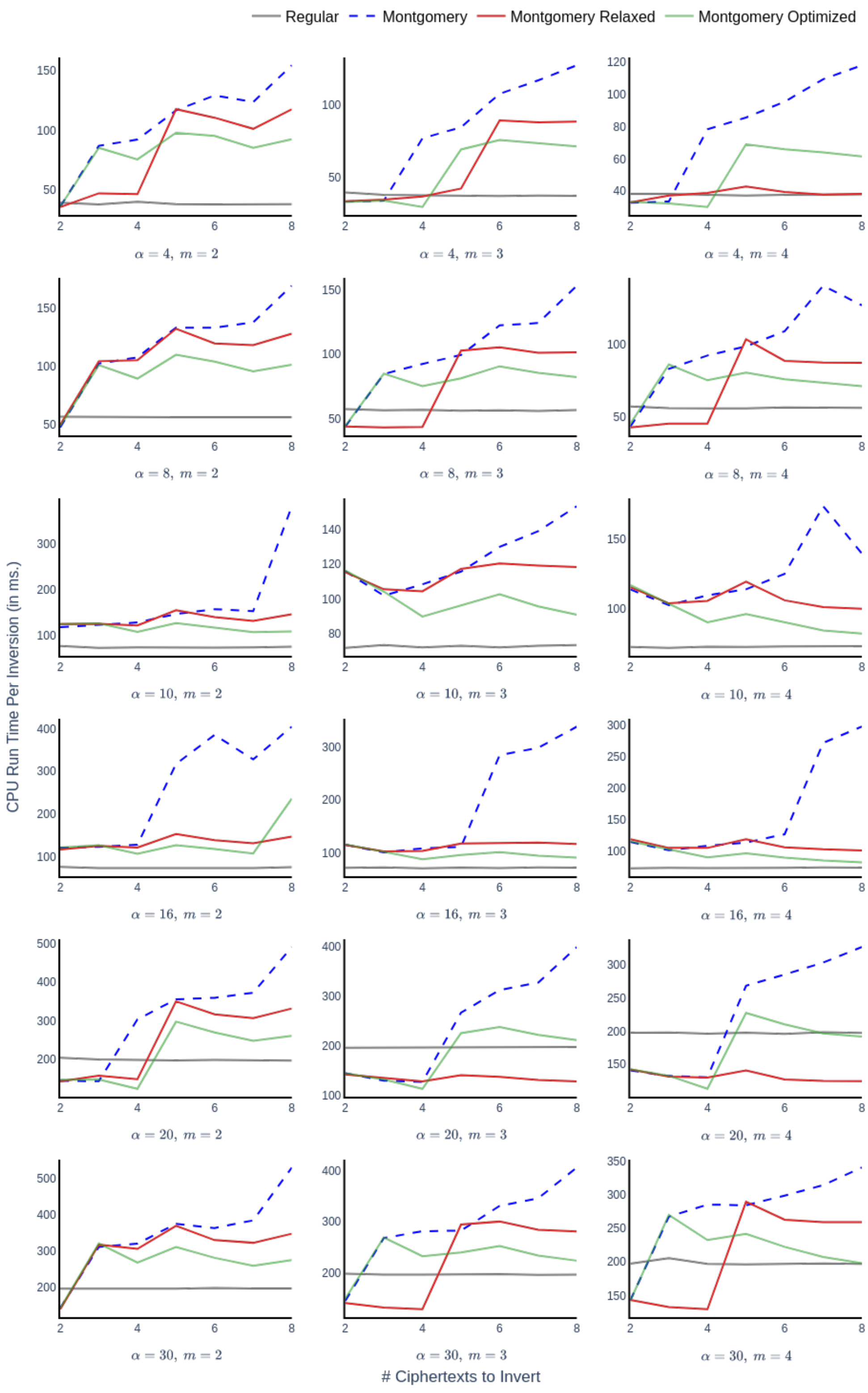


Figure 8: Implementation results for the FHE-regular, FHE-Montgomery, FHE-relaxed Montgomery, and FHE-optimized Montgomery inversion algorithms for $s \in \{2,3,4,5,6,7,8\}$, $m \in \{2,3,4\}$, and $\alpha \in \{4,8,10,16,20,30\}$. Here, s is the number of ciphertexts to invert; the input plaintexts belong to the interval $[1-1/2^m, 1)$; and α is the parameter that controls the error bound. Time is measured in milliseconds and it shows the runtime per inversion. Notice that in the 12 of the 18 cases (except the third and fourth rows in the plot), there is always at least one Montgomery based inversion algorithm and a choice of grouping ciphertexts for simultaneous inversion such that the Montgomery method outperforms regular inversion with respect to the "per-inversion" cost, and hence minimizes the overall cost.

Table 1: The complexities of FHE-regular, FHE-Montgomery, FHE-relaxed Montgomery, and FHE-optimized Montgomery inversion algorithms in terms of the circuit depth and the number of ciphertext multiplications for $s = 2, 3, 4, 5$; $m = 2, 3, 4$; and $\alpha = 4, 8$. Here, s is the number of ciphertexts to invert; the input plaintexts belong to the interval $[1 - 1/2^m, 1)$; α is the parameter that controls the error bound; and Depth is a function of d, which is the number of iterations in the Goldschmidt algorithm. Values in bold font indicate the best complexity among the Montgomery-based algorithms. Values in bold font that are also underlined indicate the best complexity among all of the Montgomery-based algorithms and the regular algorithm.

			d		Depth				# Mult		
s	α	m	Reg	Mont-based	Reg	Mont	RelaxMont	OptMont	Reg	Mont/OptMont	RelaxMont
		2	3	3	$\underline{\mathbf{4}}$	6	6	6	12	$\underline{\mathbf{9}}$	$\underline{\mathbf{9}}$
	4	3	3	3	$\underline{\mathbf{4}}$	6	6	6	12	$\underline{\mathbf{9}}$	$\underline{\mathbf{9}}$
2		4	3	3	$\underline{\mathbf{4}}$	6	6	6	12	$\underline{\mathbf{9}}$	$\underline{\mathbf{9}}$
		2	4	4	$\underline{\mathbf{5}}$	7	7	7	16	$\underline{\mathbf{11}}$	$\underline{\mathbf{11}}$
	8	3	4	4	$\underline{\mathbf{5}}$	7	7	7	16	$\underline{\mathbf{11}}$	$\underline{\mathbf{11}}$
		4	4	4	$\underline{\mathbf{5}}$	7	7	7	16	$\underline{\mathbf{11}}$	$\underline{\mathbf{11}}$
		2	3	4	$\underline{\mathbf{4}}$	9	**8**	9	18	$\underline{\mathbf{14}}$	16
	4	3	3	3	$\underline{\mathbf{4}}$	8	**7**	8	18	$\underline{\mathbf{12}}$	14
3		4	3	3	$\underline{\mathbf{4}}$	8	**7**	8	18	$\underline{\mathbf{12}}$	14
		2	4	5	$\underline{\mathbf{5}}$	10	**9**	10	24	$\underline{\mathbf{16}}$	18
	8	3	4	4	$\underline{\mathbf{5}}$	9	**8**	9	24	$\underline{\mathbf{14}}$	16
		4	4	4	$\underline{\mathbf{5}}$	9	**8**	9	24	$\underline{\mathbf{14}}$	16
		2	3	4	$\underline{\mathbf{4}}$	10	**8**	9	24	$\underline{\mathbf{17}}$	23
	4	3	3	3	$\underline{\mathbf{4}}$	9	**7**	8	24	$\underline{\mathbf{15}}$	21
4		4	3	3	$\underline{\mathbf{4}}$	9	**7**	8	24	$\underline{\mathbf{15}}$	21
		2	4	5	$\underline{\mathbf{5}}$	11	**9**	10	32	$\underline{\mathbf{19}}$	25
	8	3	4	4	$\underline{\mathbf{5}}$	10	**8**	9	32	$\underline{\mathbf{17}}$	23
		4	4	4	$\underline{\mathbf{5}}$	10	**8**	9	32	$\underline{\mathbf{17}}$	23
		2	3	5	$\underline{\mathbf{4}}$	13	**10**	12	30	$\underline{\mathbf{22}}$	34
	4	3	3	3	$\underline{\mathbf{4}}$	11	**8**	10	30	$\underline{\mathbf{18}}$	30
5		4	3	3	$\underline{\mathbf{4}}$	11	**8**	10	30	$\underline{\mathbf{18}}$	30
		2	4	6	$\underline{\mathbf{5}}$	14	**11**	13	40	$\underline{\mathbf{24}}$	36
	8	3	4	4	$\underline{\mathbf{5}}$	12	**9**	11	40	$\underline{\mathbf{20}}$	32
		4	4	4	$\underline{\mathbf{5}}$	12	**9**	11	40	$\underline{\mathbf{20}}$	32

We choose the scaling modulus Δ and the first modulus q_0 such that

$$\Delta < q_0 < \Delta^2, \tag{6}$$

i.e., $\log(\Delta) < \log(q_0) < 2\log(\Delta)$. In our implementation, we fix the value of the first modulus size to the maximum value allowed in the OpenFHE library [ABBB+22], i.e., 60 bits. We set the scaling technique to FixedAuto.

As noted in Section 4, the regular and Montgomery-based FHE inversion algorithms offer various trade-offs between the number of multiplications and circuit depth, and it does not seem to be possible to theoretically identify an inversion algorithm with the best time complexity. We also observed in Corollary 4 that the three variants of Montgomery inversion algorithms have the same circuit depth and require the same number of multiplications when $s = 2$. Moreover, as shown in Figure 8, grouping ciphertexts by $s = 2$ yields an optimal choice with respect to the per inversion cost in the 28 of the 54 Montgomery based algorithms we study in the 18 cases. Therefore, in our experiments, we implement the algorithms for $s = 2$ and, based on Corollary 4, we only focus on FHE-Montgomery. When the number of homomorphic inversions to perform is larger than 2, say $2k$, one can pair the ciphertexts and run Montgomery based algorithms k times with $s = 2$ (i.e., compute simultaneous 2-inversions k times). In our experiments, the regular and FHE-Montgomery inversion algorithms are compared over the same choices of accuracy parameter α and so over the same choices of Δ (see equation (6)).

The number of iterations in the Goldschmidt algorithm and so the multiplicative depth of the circuit change as a function of $\log(\alpha)$ (see Proposition 2). Therefore, critical values of α appear as powers of two and that we choose $\alpha \in \{4, 8, 16\}$ in our experiments. We extend this parameter set for α by the additional values of $\{10, 20, 30\}$ for more granularity. Hence, we choose $\alpha \in \{4, 8, 10, 16, 20, 30\}$. This extended set is particularly useful to observe that in some of the cases, FHE-Montgomery requires strictly larger ring dimension N because of the dependency between N, the security parameter λ, and the multiplicative depth (Depth) given by [CSY22]:

$$N > \frac{(\lambda + 110)\log(Q/\sigma)}{7.2}, \tag{7}$$

where Q is divisible by a product primes q_i for $i = 1, ..., \mathsf{Depth}$. As a result, the optimal ring dimension is not the same for the regular and FHE-Montgomery for the parameters $(\alpha = 10,\ m \in \{2, 3, 4\})$ and $(\alpha = 16,\ m \in \{2, 3, 4\})$. In these cases, the bit size of the ring dimension is equal to 14 and 15 for the regular algorithm and FHE-Montgomery, respectively. Finally, plaintext values in our experiments belong to $[\epsilon, 1)$ with $\epsilon = 1 - 1/2^m$, and we choose $m \in \{2, 3, 4\}$. In summary, the set of the selected parameters for our experiments is given in Table 2.

Table 2: Implementation parameters for the FHE-regular and FHE-Montgomery inversion algorithms for $s = 2$.

			Regular			Montgomery		
m	α	$\log(\Delta)$	$\log(N)$	Gold. Iter. d	MultDepth	$\log(N)$	Gold. Iter. d	MultDepth
2, 3, 4	4	31	14	3	4	14	3	6
2, 3, 4	8	31	14	4	5	14	4	7
2, 3, 4	10	33	14	5	6	15	5	8
2, 3, 4	16	39	14	5	6	15	5	8
2, 3, 4	20	45	15	6	7	15	6	9
2, 3, 4	30	55	15	6	7	15	6	9

Implementation For each pair (m, α), we implemented and measured the CPU times of the algorithms for $s \in \{2, 4, 8\}$, using OpenFHE [ABBB+22] on a Windows 10 machine with the following specifications: 13th Gen Intel(R) Core(TM) i7-13650HX 2.60 GHz, 64-bit operating system, x64-based processor, and 16 GB RAM. The machine was running Ubuntu 22.04.4 LTS (GNU/Linux 5.15.153.1-microsoft-standard-WSL2 x86_64) via WSL. Our experimental results are presented in Table 3 and Figure 9. The computations were performed with respect to the CKKS estimated precision from the OpenFHE library (see the last column in Table 3).

Table 3: Implementation results for inverting $s \in \{2, 4, 8\}$ ciphertexts using FHE-Montgomery $s/2$ times with simultaneous 2-inversions, and for $m \in \{2, 3, 4\}$ and $\alpha \in \{4, 8, 10, 16, 20, 30\}$. CPU run time is measured in milliseconds. See Figure 9 for a visual representation of results.

m	Gold. err. α	s	Regular (CPU)	Montgomery (CPU)	Estimated CKKS Precision (bits)
2	4	2	78.5	70.1	17
		4	157	140	17
		8	306	296	17
	8	2	135	87.5	17
		4	239	179	17
		8	472	354	17
	10	2	145	233	19
		4	290	465	19
		8	592	951	19
	16	2	156	237	25
		4	303	465	25
		8	607	944	25
	20	2	393	282	31
		4	781	566	31
		8	1572	1130	31
	30	2	388	284	41
		4	784	573	41
		8	1552	1123	41
3	4	2	77.9	65.5	17
		4	150	134	17
		8	303	265	17
	8	2	114	88.3	17
		4	227	175	17
		8	455	345	17
	10	2	147	229	19
		4	293	466	19
		8	582	919	19
	16	2	143	230	25
		4	289	465	25
		8	580	924	25
	20	2	391	283	31
		4	788	596	31
		8	1577	1137	31
	30	2	397	282	41
		4	785	574	41
		8	1580	1132	41
4	4	2	76.3	65.4	17
		4	154	133	17
		8	303	263	17
	8	2	113	86.9	17
		4	225	173	17
		8	450	351	17
	10	2	145	227	19
		4	291	460	19
		8	582	918	19
	16	2	148	229	25
		4	296	460	25
		8	585	926	25
	20	2	388	284	31
		4	784	565	31
		8	1572	1141	31
	30	2	401	290	41
		4	825	581	41
		8	1650	1161	41

For $\alpha \in \{4, 8, 20, 30\}$ and $m \in \{2, 3, 4\}$, we found that the FHE-Montgomery inversion outperforms regular inversion, achieving up to 35% speedups. However, for $\alpha \in \{10, 16\}$ and $m \in \{2, 3, 4\}$, the regular homomorphic inversion method demonstrated better time efficiency compared to the FHE-Montgomery inversion. This can be attributed to our previous observation that the ring dimension for the FHE-Montgomery inversion algorithm is significantly larger than that for the regular inversion.

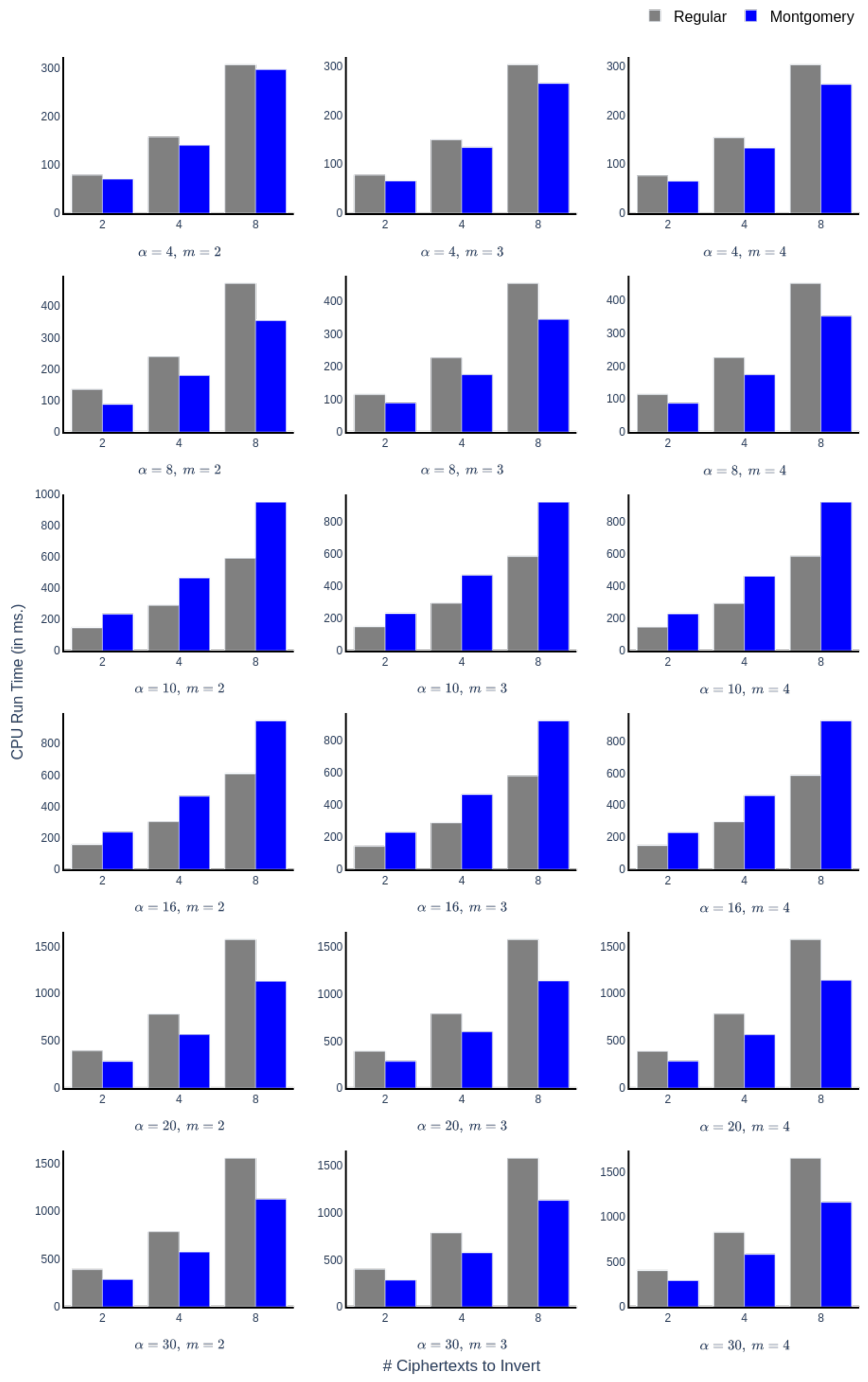


Figure 9: Implementation results for inverting $s \in \{2, 4, 8\}$ ciphertexts using FHE-Montgomery $s/2$ times with simultaneous 2-inversions, and for $m \in \{2, 3, 4\}$ and $\alpha \in \{4, 8, 10, 16, 20, 30\}$. CPU run time is measured in milliseconds. For $\alpha \in \{4, 8, 20, 30\}$, FHE-Montgomery yields up to 35% speed ups. For $\alpha \in \{10, 16\}$, FHE-Montgomery loses advantage. See Table 3 for more details.

6 Conclusion

We proposed three algorithms to solve the homomorphic s-inversion problem. Our algorithms provide different levels of trade-offs between the computational complexity and the depth of circuits and reduce the number of homomorphic inversions from s to 1. We implemented our algorithms using the CKKS scheme in the OpenFHE library and the Goldschmidt inversion. Our experiments show that, for some cases, the run time of homomorphic s-inversion can be improved up to 35% while in some other cases, regular inversion seems to outperform Montgomery-based inversion algorithms.

Acknowledgements

This project was initiated when Jean Belo Klamti was affiliated with the University of Waterloo as a post-doctoral researcher. The authors would like to thank the anonymous reviewers for their comments on the earlier versions of this article.

Author Contributions

The major contributions of the authors are as follows. Jean Belo Klamti: complexity analysis and implementation of the proposed algorithms; parameter selection and benchmarking. M. Anwar Hasan: design of the FHE-optimized Montgomery inversion algorithm. Koray Karabina: supervision of the project; design of the FHE and FHE-relaxed Montgomery inversion algorithms; complexity analysis and implementation of the proposed algorithms. Karabina and Klamti took the lead in writing the manuscript. All of the authors contributed to the reviewing and editing the original and subsequent drafts of this paper.

References

[ABBB+22] Ahmad Al Badawi, Jack Bates, Flavio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, Zeyu Liu, Daniele Micciancio, Ian Quah, Yuriy Polyakov, Saraswathy R.V., Kurt Rohloff, Jonathan Saylor, Dmitriy Suponitsky, Matthew Triplett, Vinod Vaikuntanathan, and Vincent Zucca. OpenFHE: Open-source fully homomorphic encryption library. In *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, WAHC'22, page 53–63, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3560827.3563379.

[ASP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In *Advances in Cryptology–CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I 34*, pages 297–314. Springer, 2014. doi:10.1007/978-3-662-44371-2_17.

[Ben94] Josh Benaloh. Dense probabilistic encryption. In *Proceedings of the Workshop on Selected Areas of Cryptography*, pages 120–128, 1994. URL: https://sacworkshop.org/proc/SAC_94_006.pdf.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Theory of Cryptography: Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005. Proceedings 2*, pages 325–341. Springer, 2005. doi:10.1007/978-3-540-30576-7_18.

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014. doi:10.1145/2090236.2090262.

[BLLN13] Joppe W Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In *Cryptography and Coding: 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings 14*, pages 45–64. Springer, 2013. doi:10.1007/978-3-642-45239-0_4.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Advances in Cryptology–CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 868–886. Springer, 2012. doi:10.1007/978-3-642-32009-5_50.

[Bra18] Zvika Brakerski. Fundamentals of fully homomorphic encryption - a survey. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 25, page 125, 2018. doi:10.1145/3335741.3335762.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 97–106. IEEE, 2011. doi:10.1109/FOCS.2011.12.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *Annual Cryptology Conference*, pages 505–524. Springer, 2011. doi:10.1007/978-3-642-22792-9_29.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In *Proceedings of the 5th Conference on Innovations in Theoretical Computer Science*, pages 1–12, 2014. doi:10.1145/2554797.2554799.

[CCK+13] Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrede Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In *Advances in Cryptology–EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings 32*, pages 315–335. Springer, 2013. doi:10.1007/978-3-642-38348-9_20.

[CDPR16] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 559–585. Springer, 2016. doi:10.1007/978-3-662-49896-5_20.

[CDSM15] Gizem Selcan Cetin, Yarkin Doroz, Berk Sunar, and William J. Martin. Arithmetic using word-wise homomorphic encryption. Cryptology ePrint Archive, Paper 2015/1195, 2015. https://eprint.iacr.org/2015/1195.

[CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 2020. doi:10.1007/s00145-019-09319-x.

[Che54] Pafnuty Lvovich Chebyshev. Memoires des savants etrangers présentés á l'académie de Saint-Pétersbourg. *Ch. Théorie des mécanismes connus sous le nom de parallélogrammes*, 7:539–586, 1854. URL: https://archive.org/details/mmoiresprsentsla07impe/page/n5/mode/2up.

[CKK+19] Jung Hee Cheon, Dongwoo Kim, Duhyeong Kim, Hun Hee Lee, and Keewoo Lee. Numerical method for comparison on homomorphically encrypted numbers. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019*, pages 415–445, Cham, 2019. Springer International Publishing. doi:10.1007/978-3-030-34621-8_15.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 409–437, Cham, 2017. Springer International Publishing. doi:10.1007/978-3-319-70694-8_15.

[CLT14] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Scale-invariant fully homomorphic encryption over the integers. In *Public-Key Cryptography–PKC 2014: 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings 17*, pages 311–328. Springer, 2014. doi:10.1007/978-3-642-54631-0_18.

[CMNT11] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In *Annual Cryptology Conference*, pages 487–504. Springer, 2011. doi:10.1007/978-3-642-22792-9_28.

[CN12] Yuanmi Chen and Phong Q Nguyen. Faster algorithms for approximate common divisors: Breaking fully-homomorphic-encryption challenges over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 502–519. Springer, 2012. doi:10.1007/978-3-642-29011-4_30.

[CNT12] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 446–464. Springer, 2012. doi:10.1007/978-3-642-29011-4_27.

[CS15] Jung Hee Cheon and Damien Stehlé. Fully homomophic encryption over the integers revisited. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 513–536. Springer, 2015. doi:10.1007/978-3-662-46800-5_20.

[CSY22] Jung Hee Cheon, Yongha Son, and Donggeon Yhee. Practical FHE parameters against lattice attacks. *J. Korean Math. Soc*, 59(1):35–51, 2022. doi:10.4134/JKMS.j200650.

[DM15] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 617–640, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. doi:10.1007/978-3-662-46800-5_24.

[DS20] Yarkın Doröz and Berk Sunar. Flattening NTRU for evaluation key free homomorphic encryption. *Journal of Mathematical Cryptology*, 14(1):66–83, 2020. doi:doi:10.1515/jmc-2015-0052.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on Information Theory*, 31(4):469–472, 1985. doi:10.1109/TIT.1985.1057074.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Paper 2012/144, 2012. https://eprint.iacr.org/2012/144.

[Gal02] Steven D Galbraith. Elliptic curve Paillier schemes. *Journal of Cryptology*, 15:129–138, 2002. doi:10.1007/s00145-001-0015-6.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 169–178, New York, NY, USA, 2009. Association for Computing Machinery. doi:10.1145/1536414.1536440.

[GH11] Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 107–109. IEEE, 2011. doi:10.1109/FOCS.2011.94.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 365–377, 1982. doi:10.1145/800070.802212.

[Gol64] Robert E. Goldschmidt. *Applications of division by convergence.* PhD thesis, Massachusetts Institute of Technology, 1964. URL: https://dspace.mit.edu/handle/1721.1/11113.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 75–92. Springer, 2013. doi:10.1007/978-3-642-40041-4_5.

[Har08] David G. Harris. Simultaneous field divisions: an extension of Montgomery's trick. Cryptology ePrint Archive, Paper 2008/199, 2008. https://eprint.iacr.org/2008/199.

[IIMP22a] Ilia Iliashenko, Malika Izabachène, Axel Mertens, and Hilder Vitor Lima Pereira. Homomorphically counting elements with the same property. *Proceedings on Privacy Enhancing Technologies*, 4:670–683, 2022. doi:10.56553/popets-2022-0127.

[IIMP22b] Ilia Iliashenko, Malika Izabachène, Axel Mertens, and Hilder VL Pereira. Homomorphically counting elements with the same property. https://www.youtube.com/watch?v=P2XdA758JUo, 2022. [Online; accessed 30-May-2024].

[KGV15] Alhassan Khedr, Glenn Gulak, and Vinod Vaikuntanathan. SHIELD: Scalable homomorphic implementation of encrypted data-classifiers. *IEEE Transactions on Computers*, 65(9):2848–2858, 2015. doi:10.1109/TC.2015.2500576.

[KK21] Sharmila Devi Kannivelu and Sunwoong Kim. A homomorphic encryption-based adaptive image filter using division over encrypted data. In *2021 IEEE 27th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 67–72, Los Alamitos, CA, USA, aug 2021. IEEE Computer Society. doi:10.1109/RTCSA52859.2021.00016.

[LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, pages 1219–1234, 2012. doi:10.1145/2213977.2214086.

[MGH10] Carlos Aguilar Melchor, Philippe Gaborit, and Javier Herranz. Additively homomorphic encryption with d-operand multiplications. In *Advances in Cryptology–CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings 30*, pages 138–154. Springer, 2010. doi:10.1007/978-3-642-14623-7_8.

[Mon87] Peter Lawrence Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48(177):243–264, 1987. doi:10.1090/s0025-5718-1987-0866113-7.

[NK15] Koji Nuida and Kaoru Kurosawa. (Batch) fully homomorphic encryption over integers for non-binary message spaces. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 537–555. Springer, 2015. doi:10.1007/978-3-662-46800-5_21.

[NS98] David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, pages 59–66, 1998. doi:10.1145/288090.288106.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 223–238. Springer, 1999. doi:10.1007/3-540-48910-X_16.

[RAD$^+$78] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, 4(11):169–180, 1978. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=c365f01d330b2211e74069120e88cff37eacbcf5.

[Rap02] Joseph Raphson. *Analysis aequationum universalis*, volume 1. Typis T.B. prostant venales apud A. and I. Churchill, 1702.

[Rot11] Ron Rothblum. Homomorphic encryption: From private-key to public-key. In *Theory of Cryptography Conference*, pages 219–234. Springer, 2011. doi:10.1007/978-3-642-19571-6_14.

[SS10] Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In *Advances in Cryptology - ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16*, pages 377–394. Springer, 2010. doi:10.1007/978-3-642-17373-8_22.

[SS11] Peter Scholl and Nigel P Smart. Improved key generation for Gentry's fully homomorphic encryption scheme. In *IMA International Conference on Cryptography and Coding*, pages 10–22. Springer, 2011. doi:10.1007/978-3-642-25516-8_2.

[SV10] Nigel P Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Public Key Cryptography–PKC 2010: 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings 13*, pages 420–443. Springer, 2010. doi:10.1007/978-3-642-13013-7_25.

[VDGHV10] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*, pages 24–43. Springer, 2010. doi:10.1007/978-3-642-13190-5_2.