DA-Encrypt: Homomorphic Encryption via Non-Archimedean Diophantine Approximation — Preliminary Report

Jeff Hoffstein[1⋆], Jill Pipher[1], John M. Schanck[2,3], Joseph H. Silverman[1⋆], William Whyte[3], Zhenfei Zhang[3]

[1] Mathematics Department, Brown University, Providence, RI 02912 USA
{jhoff,jpipher,jhs}@math.brown.edu,
[2] University of Waterloo, Waterloo, Canada,
[3] Security Innovation, Wilmington, MA 01887, USA
{jschanck,wwhyte,zzhang}@securityinnovation.com

Abstract. We give a theoretical description of a new homomorphic encryption scheme DA-Encrypt that is based on (non-archimedean) Diophantine Approximation.

1 Introduction

The concept of Fully Homomorphic Encryption (FHE) was introduced almost 40 years ago [25], and an efficient embodiment of such a scheme is a holy grail of cryptography. It would allow one to arbitrarily carry out computations on a plaintext without the necessity of decrypting the ciphertext first. In his breakthrough work, Gentry [11–13] presented the very first homomorphic encryption scheme in the context of ideal lattices. Moreover, he presented a framework for achieving such schemes. It has been an active field of research in cryptography since then, due to many potential applications (such as verifiable computing [10]) and its use in cryptographic tools (such as multi-linear maps [9] and program obfuscation [8]).

Gentry's framework consists of two steps. In the first step, one constructs a somewhat homomorphic encryption scheme (sometimes known as partial homomorphic encryption or leveled homomorphich encryption), a homomorphic scheme that allows a limited number of multiplication and addition gates. Ciphertexts in these schemes contain noise that grows with homomorphic operations. In the second

⋆ Research partially supported by NSF EAGER DMS-1349908

step, one bootstraps the scheme by converting a noisy ciphertext into an almost noise free ciphertext. Repeating the above steps, one is able to evaluate a circuit with arbitrary depth, and hence one achieves a fully homomorphic encryption scheme.

To date, there have been three types of fully homomorphic encryption schemes that have followed Gentry's framework. There have been schemes based on ideal lattices [11–14, 27, 28], based on integers [6, 7, 29], and based on Learning With Error [1, 2, 15]. To the best of our knowledge, there have been three implementations [14, 6, 16], corresponding to each type. Cryptanalysis of the first two implementations can be found in [3, 23, 24, 4].

A crucial point in Gentry's theoretical framework is the ability to bootstrap. While a somewhat homomorphic encryption scheme can be simple and efficient, the introduction of bootstrapping makes fully homomorphic schemes impractical for the foreseeable future. Because of this, it has been proposed that the notion of fully homomorphic public key encryption should be relaxed for practical use. In particular, the following questions have been asked:

- Would a practical somewhat homomorphic encryption scheme be sufficient for many purposes?
- Would an efficient symmetric somewhat homomorphic encryption scheme have practical applications?

The first question has been studied intensively, for instance, in [22]. It has certainly been shown that somewhat homomorphic encryption schemes enable important applications; see [17] for a survey of such applications.

Concerning the second question, in many applications, only a symmetric homomorphic encryption scheme is required. This often occurs in multi-party (2 party) computation scenarios, where one party contributes sensitive data, while the other contributes algorithms and public data.

In a two party computation scenario, for instance, Alice might want to outsource some computations to Bob, but not want Bob to learn the relevant data. She would encrypt her data using a symmetric homomorphic encryption scheme and send the ciphertexts to Bob. Bob would operate on this data using his (public or private)

algorithm and his public data. A crucial point here is that Bob would not need to possess the ability to encrypt his own data. In almost all homomorphic encryptions a plaintext is already a ciphertext with no noise injected. Thus, Bob would be able to do computations using both confidential encrypted data from Alice and public data of his own. Upon receipt of the encrypted results from Bob, Alice could decrypt it with her secret key.

During this procedure, Alice's confidential data remains hidden from Bob. There are, however, other issues beyond the scope of homomorphic encryption that arise. These include fairness (Alice learns everything, Bob learns almost nothing), as well as other aspects, such as Bob proving that he has computed honestly, etc.

1.1 Our contribution

In this paper, we present a theoretical description of a new symmetric somewhat homomorphic encryption scheme inspired by ideas from the theory of (non-archimedean) Diophantine approximation. We also show how, with a few modifications, it can be turned into a public key system. At a high level, our construction is a sort of inverse of schemes based on principal ideal lattices [14, 27]. In [14] and [27], the public keys are a bad basis (Hermite normal form basis) of a principal ideal lattice, while the secret key is a good basis (the generator) of the principal ideal lattice. In this preliminary note we have not attempted to quantify specific parameters that give particular security levels, nor do we give operating characteristics of an implementation.

In our construction, we use a good basis (with some noise injected) to generate the ciphertexts, while the secret key is the lattice itself (in the form of a bad basis). We ensure that an attacker is not able to obtain a noise-free basis from a noisy basis, which can be seen as an analogue of the learning with error problem over a principal ideal lattice.

The ciphertexts that an attacker sees are points close to an unknown principal ideal lattice. We conjecture that even if an attacker is able to solve the shortest vector problem in a principal ideal lattice, he will not be able to break our scheme, because our lattice is secret. This allows our scheme to operate on lattices with smaller

dimensions. We note that the relevance of this remark stems from several recent results which suggest that the shortest vector problem in principal ideal lattice may be solvable with a quantum computer, and that the shortest vector problem in certain types of principal ideal lattices may be solvable sub-exponentially using classic computers. If these algorithms work, they will allow an attacker to break earlier principal ideal lattice based cryptosystems such as [14, 27].

2 Background

We begin by recalling some background.

2.1 Notation

A vector from $\mathbb{R}^n$ is represented as n-tuple $\boldsymbol{v} = \langle v_1, \dots v_n \rangle$. We use $|\boldsymbol{v}| = \sqrt{v_1^2 + \cdots + v_n^2}$ and $|\boldsymbol{v}|_\infty = \max |v_i|$ for the Euclidean norm and infinity norm, respectively. We use row vectors of a matrix to represent a lattice basis. A basis $\mathbf{B}$ of a lattice $\mathcal{L}$ is thus a matrix $(\boldsymbol{b}_1, \dots, \boldsymbol{b}_n)$ whose rows are the vectors $\boldsymbol{b}_1, \dots, \boldsymbol{b}_n$. We often identify a vector $\boldsymbol{f} = \langle f_1, f_2, \dots, f_n \rangle$ with the polynomial $f(x) = f_1 + f_2 x + f_3 x^2 + \cdots + f_n x^{n-1}$ whose coefficients are the coordinates of $\boldsymbol{f}$. When there is no ambiguity, we will mix these two notations without further clarification.

Table 1 presents a list of notation used throughout the paper. The definitions and further explanation are given later.

n	A parameter for the degree of polynomials and dimension of lattices.
$F(x)$	a polynomial with integer coefficients. (In this paper, we use $F(x) = x^n + 1$ as an example.)
$\mathbb{Z}_q$	the ring $\mathbb{Z}/q\mathbb{Z}$ of integers modulo q.
R	The truncated polynomial ring $\mathbb{Z}_q(x)/(F(x))$.
$f(x), \beta, q$	A polynomial and integers satisfying $F(\beta) \equiv f(\beta) \equiv 1 \pmod q$. (Used to define a principal ideal lattice.)
t	The number of public polynomials. (Used for asymmetric version.)

Table 1. A list of notation

2.2 Lattices

Lattice theory, also known as the geometry of numbers, was introduced by Minkowski in 1896 [21]. We refer readers to [18, 20] for a more detailed account.

A *lattice* $\mathcal{L}$ is a discrete sub-group of $\mathbb{R}^n$, or equivalently the set of all the integral combinations of a set of $\mathbb{R}$-linearly independent vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m$ over $\mathbb{R}$:

$$\mathcal{L} = \mathbb{Z}\boldsymbol{b}_1 + \mathbb{Z}\boldsymbol{b}_2 + \cdots + \mathbb{Z}\boldsymbol{b}_m.$$

A basis of $\mathcal{L}$ is denoted $\mathbf{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$, and m is called the dimension of $\mathcal{L}$ and denoted $\dim(\mathcal{L})$. In this paper we deal with ideal lattices which are full rank lattices, i.e., m equals n.

Let $F(x) \in \mathbb{Z}[x]$ be a monic irreducible polynomial of degree n and let R be the quotient polynomial ring $\mathbb{Z}[x]/(F(x))$, which we view as a copy of $\mathbb{Z}^n$ sitting inside $\mathbb{R}^n$. An *ideal lattice* $\mathcal{L}$ over R is, as its name suggests, simply an ideal of the ring R; cf. [19]. Alternatively, an ideal lattice is a $\mathbb{Z}$-submodule $\mathcal{L}$ of R with the property that for for every $\boldsymbol{v} \in \mathcal{L}$, the vector corresponding to the polynomial $x\boldsymbol{v} \bmod F(x)$ also belongs to $\mathcal{L}$. An ideal lattice that is generated by a single element (as an ideal) is called a *principal ideal lattice*.

Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be lattices in $\mathbb{R}^n$. Then $\mathcal{L}_1$ is a *sub-lattice* of $\mathcal{L}_2$ if all points of $\mathcal{L}_1$ are also in $\mathcal{L}_2$, i.e., if $\mathcal{L}_1 \subseteq \mathcal{L}_2$. We denote the intersection of two lattices by $\mathcal{L}_1 \cap \mathcal{L}_2$. It is a lattice, and if $\mathcal{L}_1$ and $\mathcal{L}_2$ are have dimension n, then so does $\mathcal{L}_1 \cap \mathcal{L}_2$.

2.3 Homomorphic encryption

A homomorphic encryption scheme ξ consists of four algorithms:

KeyGen, Encrypt, Decrypt and Eval.

- KeyGen(λ): Input a security parameter λ; output a pair of keys $\mathbf{k_1}$ and $\mathbf{k_2}$.
- Encrypt(m, $\mathbf{k_1}$): Input a plaintext m and $\mathbf{k_1}$; output a corresponding ciphertext c.
- Decrypt(c, $\mathbf{k_2}$): Input a ciphertext c and $\mathbf{k_2}$; output a corresponding plaintext m.

– $\textsc{Eval}(c_1, c_2, \ldots, c_t, \mathcal{C}^t, \mathbf{k_1})$: Input n ciphertexts $c_1, c_2, \ldots, c_t$, a permitted circuit $\mathcal{C}^t$, and an optional key $\mathbf{k_1}$; output $\mathcal{C}^t(c_1, c_2, \ldots, c_t)$.

The first three algorithms follow the definition of an encryption scheme. In a public key scheme, $\mathbf{k_1}$ is public and $\mathbf{k_2}$ is secret, while in a symmetric encryption scheme, both $\mathbf{k_1}$ and $\mathbf{k_2}$ are secret, and in most cases $\mathbf{k_1} = \mathbf{k_2}$. The last algorithm Eval is defined as follows: input a set of ciphertexts $\{c_i\}$ whose corresponding plaintexts are $\{m_i\}$ and a circuit $\mathcal{C}$, and output another ciphertext c. (The key used to encrypt the m_i is an optional input to the algorithm.) The homomorphic scheme is correct if it satisfies the following:

$$\textsc{Decrypt}\Big(\textsc{Eval}\big(\mathcal{C}, \{c_i\}, \mathbf{k_1}\big), \mathbf{k_2}\Big) = \mathcal{C}(m_1, \ldots, m_t). \tag{1}$$

Definition 1 (Homomorphic Encryption). *The scheme*

$$\xi = (\textsc{KeyGen}, \textsc{Encrypt}, \textsc{Decrypt}, \textsc{Eval})$$

is homomorphic for a class $\mathcal{C}$ of circuits if Equation (1) *is valid for all circuits $C \in \mathcal{C}$. The scheme ξ is fully homomorphic if it is correct for all boolean circuits. It is somewhat homomorphic if it is correct for boolean circuits up to a certain depth. It is compact if, for any circuit $C \in \mathcal{C}$ with the number of inputs polynomial in λ, the size of ciphertexts output by* Eval *is bounded by a polynomial function of λ.*

Looking ahead, we will show that our proposed scheme is somewhat homomorphic. We conjecture our scheme can also be fully homomorphic using Gentry's bootstrapping technique. However, such modification would require larger parameters and hence would be less practical. Since our goal is practical homomorphic encryption, we focus on efficient somewhat homomorphic encryption schemes.

3 The DA-Encrypt symmetric encryption scheme

3.1 Overview

Let $F(x)$ be the monic irreducible polynomial that defines our polynomial ring $\mathsf{R} = \mathbb{Z}[x]/(F(x))$. For simplicity, we use $F(x) = x^n + 1$.

Our scheme starts by finding a polynomial $f(x)$ and integers β and q satisfying

$$f(\beta) \equiv F(\beta) \equiv 0 \bmod q.$$

The polynomial $f(x)$ is a generator of a principal ideal lattice, which is spanned as a $\mathbb{Z}$-module by the vectors $x^i f(x) \bmod F(x)$. (These are more-or-less the rotations of the vector $\boldsymbol{f}$, with some negative signs due to the fact that we're using $x^n + 1$ instead of $x^n - 1$.) We show in Section 5 how to generate such a principal ideal lattice efficiently. The coefficients of $f(x)$ have size on the order of $q^{1/n}$, i.e., $|\boldsymbol{f}|_\infty \approx q^{1/n}$.

We choose a random integer p that is smaller than and relatively prime to q. The encryption and decryption keys are then given by:

$$\text{Encrypt: } \mathbf{k_1} = (f(x), p) \qquad\qquad \text{Decrypt: } \mathbf{k_2} = (\beta, q).$$

To encrypt a plaintext $m \in \mathbb{Z}$, one randomly generates a polynomial $a(x)$ with (small) coefficients[1] and a small integer b, and computes

$$c(x) = a(x)f(x) + bp + m \bmod F(x).$$

To decrypt a ciphertext $c(x)$, one computes

$$m = \Big(c(\beta) \bmod q\Big) \bmod p.$$

During decryption we perform computations over $\mathbb{Z}_q$, but since $f(\beta) \equiv 0 \pmod q$ and since b, p, and m are small, a restricted number of multiplications of ciphertexts over $\mathbb{Z}$ will not cause any wraparound of the coefficients modulo q. It is quite straightforward to see that $\big(c(\beta) \bmod q\big) \bmod p$ is equal to $m \bmod p$, provided that b, p, and m are sufficiently small compared with q.

The scheme that we have described is somewhat homomorphic. A ciphertext is a vector that is close to being in a (secret) principal ideal lattice, where the distance to the lattice is the message plus some noise. This property is sufficient to perform a limited number of homomorphic operations.

[1] It is not necessary for $a(x)$ to have small coefficients if one merely wants decryption to work properly; but there will be coefficient explosion in the product of ciphertexts if the coefficients are chosen to be very large.

3.2 The DA-Encrypt algorithm (symmetric version)

DAE_SYM is a symmetric somewhat homomorphic encryption scheme that uses four PPT algorithms:

DAE_SYM.KeyGen, DAE_SYM.Encrypt,
DAE_SYM.Decrypt, DAE_SYM.Eval.

- **DAE_SYM.KeyGen**: Input a security parameter λ. Output keys $(\mathbf{k_1}, \mathbf{k_2})$:
  - Generate $\{f(x), \beta, q\} \in \mathsf{R} \times \mathbb{Z} \times \mathbb{Z}$ such that
  $$f(\beta) \equiv F(\beta) \equiv 0 \bmod q.$$
  - Pick a random integer p that is co-prime with q and sufficiently small compared to q.
  - Output keys $\mathbf{k_1} = \{f(x), q\}$ and $\mathbf{k_2} = \{\beta, q, p\}$.
- **DAE_SYM.Encrypt**: Input a plaintext m from plaintext space and a key $\mathbf{k_1}$. Output a ciphertext $e(x)$:
  - Choose a random polynomial $a(x)$ and a random integer b such that $|a|_\infty$ and $|b|$ are small and use them to generate the polynomial
  $$e(x) = a(x)f(x) + bp + m \bmod F(x).$$
  - Output $e(x)$ as the ciphertext.
- **DAE_SYM.Decrypt**: Input a ciphertext, and a secret key. Output a plaintext using the formula
$$m = \Big(e(\beta) \bmod q\Big) \mod p.$$
- **DAE_SYM.Eval**: Input two ciphertexts e_1, e_2 and an evaluation circuit $\mathrm{Eval}(\cdot\,,\cdot)$. Output a new ciphertext $e(x)$ that encrypts $\mathrm{Eval}(m_1, m_2)$:
$$e(x) = \mathrm{Eval}(e_1(x), e_2(x)).$$

3.3 Correctness of decryption

For decryption, we have

$$\begin{aligned} e(\beta) &= a(\beta)f(\beta) + bp + m \\ &\equiv bp + m \pmod{q} \\ &\equiv m \pmod{p}, \end{aligned}$$

where the last line is valid provided that $|bp+m| < q/2$, so the least residue of $bp + m$ modulo q is exactly equal to $bp + m$. Therefore under this assumption, we have

$$m = \Big(e(\beta) \bmod q\Big) \bmod p.$$

3.4 Homomorphic correctness of evaluation

The DAE_SYM scheme is partially homomorphic, i.e., it supports a limited number of additions and multiplications.

Wet take two ciphertexts

$$\begin{aligned} e_1 &= a_1(x)f(x) + b_1p + m_1, \\ e_2 &= a_2(x)f(x) + b_2p + m_2. \end{aligned}$$

Their sum is

$$e_+ = e_1 + e_2 = \big(a_1(x) + a_2(x)\big)f(x) + (b_1 + b_2)p + m_1 + m_2.$$

Then

$$\begin{aligned} e_+(\beta) &= \big(a_1(\beta) + a_2(\beta)\big)f(\beta) + (b_1 + b_2)p + m_1 + m_2 \\ &\equiv (b_1 + b_2)p + m_1 + m_2 \pmod{q}. \end{aligned}$$

If $\big|(b_1+b_2)p+m_1+m_2\big| < q/2$, then the least residue of $e_+(\beta) \bmod q$ is exactly equal to $(b_1 + b_2)p + m_1 + m_2$, which leads to the correct decryption

$$\Big(e_+(\beta) \bmod q\Big) \bmod p = \Big((b_1+b_2)p+m_1+m_2\Big) \mod p = m_1+m_2.$$

Similarly for multiplication,

$$\begin{aligned} e_\times(\beta) &= e_1(\beta)e_2(\beta) \\ &\equiv \big(a_1(\beta)f(\beta) + b_1p + m_1\big)\big(a_2(\beta)f(\beta) + b_2p + m_2\big) \pmod q \\ &\equiv (b_1p + m_1)(b_2p + m_2) \pmod q \\ &\equiv b_1b_2p^2 + b_1pm_2 + b_2pm_1 + m_1m_2 \pmod q. \end{aligned}$$

If $|b_1b_2p^2 + b_1pm_2 + b_2pm_1 + m_1m_2| < q/2$, then we see that

$$\begin{aligned} \Big(e_\times(\beta) \bmod q\Big) \bmod p &= (b_1b_2p^2 + b_1pm_2 + b_2pm_1 + m_1m_2) \bmod p \\ &= m_1m_2 \bmod p. \end{aligned}$$

Clearly, the error term grows with the number of operations, particularly multiplications. At some point, the error becomes greater than $q/2$, and then decryption fails to give the correct plaintext. To be more specific, when evaluating a circuit C over many ciphertexts $e_1, e_2, \dots$, if the noise term is greater than q, then the message is totally destroyed and decrypting $C(e_1, e_2, \dots)$ does not lead to any useful information. We also note that plaintexts are only known modulo p, so if a computation $C(m_1, m_2, \dots)$ gives a value larger than p, then the decrypted value is really $C(m_1, m_2, \dots) \bmod p$.

3.5 Security Analysis

We assume that an attacker has access to an arbitrarily long list of encryptions of 0, which we denote by $e^{(1)}, e^{(2)}, e^{(2)}, \dots$. We write $e^{(j)}$ as

$$e^{(j)}(x) = a_0^{(j)} + a_1^{(j)}x + \cdots + a_{n-1}^{(j)}x^{n-1}.$$

The attacker would like to determine integers β and q such that some (large) subset of the numbers

$$e^{(1)}(\beta) \bmod q, \;\; e^{(2)}(\beta) \bmod q, \;\; e^{(3)}(\beta) \bmod q, \dots$$

has a moderately large greatest common divisor, which will be the secret value of p. In fact, rather than searching for β and q, an attacker could search for a vector of integers $(\beta_0, \beta_1, \dots, \beta_{n-1}, q)$ such that for each j there exists an integer $\ell^{(j)}$ such that the quantity

$$\left|a_0^{(j)}\beta_0 + a_1^{(j)}\beta_1 + \cdots + a_{n-1}^{(j)}\beta_{n-1} - \ell^{(j)}q\right|$$

is considerably smaller than q. Such a vector of integers will, if it exists, with high probability be unique and will give the actual β, with β_i necessarily equal to β^8. However there does not appear to be a way to set up such a search as a lattice problem. If q were public, we could search for the vector $(\beta_0, \ldots, \beta_{n-1})$ as the solution to a closest vector problem, but with q also private, the problem seems to be moved outside of the realm of lattice problems.

3.6 Connection with the Approximate GCD problem

Definition 2 (Approximate GCD problem). *An approximate greatest common divisor problem, parameterized by* (α, γ, δ)*, is given a set of integers* $\{X_1, \ldots, X_n\}$ *where* $X_i = \mathsf{g}_i\mathsf{p} + \mathsf{r}_i$, $\log|\mathsf{g}_i| < \alpha$, $\log|\mathsf{p}| < \gamma$ *and* $\log|\mathsf{r}_i| < \delta$*, find* p*.*

Given many ciphertexts $e_1(x), \ldots, e_t(x)$, one can extract the constant terms $e_i(0)$. Note that $e_i(0)$ is in the form of $z_i + m_i + b_i p$, where z_i is the constant term of $a_i(x)f(x) \bmod F(x)$ and m_i is the message. Given those integers $e_1(0), \ldots, e_t(0)$, finding p is an approximate greatest common divisor problem.

Proposition 1. *If there exists an algorithm* $\mathcal{A}$ *that finds the secret key* p *from the ciphertexts* $\{e_i(x)\}$*, where* $\log|p| < \gamma$*,* $\log|f_i(x)| < \alpha$ *and* $\log|b_i| < \delta$*, then there exists another algorithm* $\mathcal{B}$ *that solves the Approximate GCD problem with* (α, γ, δ)*.*

Proof.

A lattice attack from this point of view would take the following form: we know that

$$e_1(0)c_2 - e_2(0)c_1 = z_1c_2 - z_2c_1.$$

This tells us that within the lattice spanned by the row vectors of $\mathbf{B}$, where

$$\mathbf{B} = \begin{vmatrix} 1 & e_2(0) & e_3(0) & \ldots & e_t(0) \\ 0 & e_1(0) & 0 & \ldots & 0 \\ 0 & 0 & e_1(0) & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & e_1(0) \end{vmatrix},$$

there exists a vector

$$\boldsymbol{v} = \langle b_1, (z_1 b_2 - z_2 b_1), (z_1 b_3 - z_3 b_1), \ldots, (z_1 b_t - z_t b_1) \rangle .$$

From $\boldsymbol{v}$ one should be able to recover the secret key p. Hence, we need our parameters to ensure that lattice reduction on basis $\mathbf{B}$ will not be able to find $\boldsymbol{v}$. This problem has been recently analyzed by Cohn and Henninger, [5], and our parameters are chosen to be resistant to this type of attack.

4 Conversion to a public key scheme

4.1 Overview

In this section we explain show how to build a public key scheme using DA-Encrypt. Such a conversion is reasonably standard and has been adopted in other integer based and lattice-based FHE schemes. The process is essentially to publish a collection of encryptions of 0.

So to create a PKC, we start by publishing a list of encrypions of 0, say

$$g_i(x) = a_i(x) f(x) + b_i p \bmod F(x) \quad \text{for } 1 \leq i \leq t.$$

Thus each $g_i(x)$ is a ciphertext corresponding to an encryption of 0 in the symmetric scheme. The public key is the set of polynomials $\{g_i(x) : 1 \leq i \leq t\}$, and the secret key is $\{\beta, q, p\}$ as before. The security of the symmetric scheme ensures that the attacker does not gain useful information from the public key.

To encrypt a plaintext m, choose small integers r_i and compute the ciphertext as

$$c(x) = \left(\sum_{i=1}^{t} r_i g_i(x) \right) + m \pmod{F}(x).$$

To decrypt it, compute as usual

$$\Big(c(\beta) \bmod q \Big) \bmod p.$$

As in the symmetric setting, a ciphertext is a vector close to the unknown principal ideal lattice, where the distance to the lattice is the plaintext plus some noise. However, due to the introduction of r_i, this distance is greater than in the symmetric setting, so the number of allowable homomorphic operations is reduced to some extent.

4.2 The DA-Encrypt algorithm (asymmetric version)

DAE_ASY is an asymmetric homomorphic encryption scheme that uses four PPT algorithms:

$$\begin{array}{ll} \text{DAE\_ASY.KeyGen}, & \text{DAE\_ASY.Encrypt}, \\ \text{DAE\_ASY.Decrypt}, & \text{DAE\_ASY.Eval}. \end{array}$$

- DAE_ASY.KeyGen: input a security parameter λ, output a public key and a security key;
  - Generate $\{f(x), \beta, q\} \in \mathsf{R} \times \mathbb{Z} \times \mathbb{Z}$ such that $f(\beta) = 0 \bmod q$;
  - Pick a random integer p that is co-prime with q and sufficiently small compared to q;
  - Generate t polynomials

$$g_i(x) = a_i(x)f(x) + b_i p \bmod q,$$

    where $a_i(x)$ and b_i are picked at random such that the coefficients of a_i and the b_i are small;
  - Publish $\{g_i(x)\}$ as the public key;
  - Keep β, p, q as the secret key;
- DAE_ASY.Encrypt: input a message $m \in \mathbb{Z}$, and a public key, output a ciphertext;
  - Randomly choose t small integers $r_i \in \mathbb{Z}$;
  - Compute

$$e(x) = \sum_{i=1}^{t} r_i g_i(x) + m;$$

  - Output $e(x)$ as the ciphertext;
- DAE_ASY.Decrypt: input a ciphertext, and a secret key, output a message;
  - Output

$$m = e(\beta) \bmod q \mod p;$$

- DAE_ASY.Eval: input two ciphertexts e_1, e_2 and an evaluation circuit $Eval(*, *)$, return a new ciphertext that encrypts $Eval(m_1, m_2)$;
  - Return $e(x) = Eval(e_1(x), e_2(x))$.

4.3 Correctness of decryption

For decryption, we have

$$\begin{aligned} e(\beta) &= \sum_{i=1}^{t} r_i g_i(\beta) + m \\ &= \sum_{i=1}^{t} (r_i a_i(\beta) f(\beta) + r_i c_i p) + m \\ &\equiv r_i c_i p + m \pmod{q} \\ &\equiv m \pmod{p} \end{aligned}$$

Therefore, we have

$$m = e(\beta) \bmod p \bmod q.$$

4.4 Correctness of evaluation, homomorphism

For two ciphertexts:

$$\begin{aligned} e_1 &= \sum_{i=1}^{t} (r_{1,i} a_{1,i}(x) f(x) + r_{1,i} c_{1,i} p) + m_1, \\ e_2 &= \sum_{i=1}^{t} (r_{2,i} a_{2,i}(x) f(x) + r_{2,i} c_{2,i} p) + m_2 \end{aligned}$$

The addition of the two is:

$$\begin{aligned} e_+ = e_1 + e_2 = m_1 + m_2 + \sum_{i=1}^{t} (r_{1,i} a_{1,i}(x) + r_{2,i} a_{2,i}(x)) f(x) \\ + \sum_{i=1}^{t} (r_{1,i} c_{1,i} + r_{2,i} c_{2,i}) p \end{aligned}$$

So the evaluation $e_+(\beta) = m_1 + m_2 \bmod q \bmod p$ is correct as long as all noise terms are small. We omit the details as it is similar to DAE_SYM.

4.5 Security analysis

Identifying an encryption of 0. The major difference between DAE_ASY and DAE_SYM is that in the asymmetric setting, a ciphertext is a linear combination of a finite list of encrypted 0's and the plaintext. Hence, it is important that an attacker should not able to determine the plaintext by recovering the linear combination.

Recall that a ciphertext has the form

$$e(x) = \sum_{i=1}^{t} r_i g_i(x) + m.$$

We let $\mathcal{L}(\mathbf{B})$ be the lattice spanned by the rows of the following matrix, where e_i denotes the ith coeffcient of e.

$$\mathbf{B} = \begin{vmatrix} g_{1,0} & g_{1,1} & \dots & g_{1,n-1} & 1 & 0 & \dots & 0 \\ g_{2,0} & g_{2,1} & \dots & g_{2,n-1} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ g_{t,0} & g_{t,1} & \dots & g_{t,n-1} & 0 & 0 & \dots & 1 \\ e_0 & e_1 & \dots & e_{n-1} & 0 & 0 & \dots & 0 \end{vmatrix}.$$

Then one easily sees that the vector

$$\boldsymbol{r} = \langle, 0, \dots, 0, r_1, r_2, \dots r_k \rangle$$

is in $\mathcal{L}(\mathbf{B})$. If $|\boldsymbol{r}|$ is small enough, then it can be found via reduction of the lattice basis $\mathbf{B}$. One may view this problems as a generalized knapsack problem where elements are polynomials rather than integers.

Remark 1. Semantic security can be reduced to a shortest vector problem of this particular lattice $\mathcal{L}(\mathbf{B})$. However, unlike earlier ideal lattice-based cryptosystems, the basis of $\mathcal{L}(\mathbf{B})$ is the basis of a principal ideal lattice perturbed by some random noise. So an ideal lattice SVP solver will not directly break our scheme. However, at this point we do now know how to establish a reduction to a generic lattice, since our lattices have some special structure.

5 Key generation technique

5.1 Generating a single principal ideal lattice

As mentioned before, for simplicity, we use $F(x) = x^n + 1$. For a randomly chosen $f(x)$, we need to compute the resultant of f and F, find a large prime factor of the resultant, which we will call q, and then factor f and F mod q to find a common root β. The approach to this we are currently using, which does not involve factoring, requires the computation of the inverse of f over $\mathbb{Z}_{(2^\ell)^n+1}$. We need to pick suitable ℓ, for instance, to ensure that $(2^\ell)^n$ is greater than the resultant of $f(x)$ and $F(x)$. In our case, we can safely use $\ell = 1$. We chose this ring since it guarantees that $F(x)$ has n integer roots in this field, and locating these roots is almost free,i.e, when n is even, the roots are simply $\{\pm 2, \pm 2^3, \pm 2^5, \ldots, \pm 2^{n-1}\}$.

We wish to find a polynomial $w(x)$ such that $w(x) \cdot f(x) = d \bmod F(x)$. We sample $f(x)$ at all the roots as computing $f(\zeta_i)$ for $0 \leq i < n$, where ζ_i is the $(i+1)$-th root of $F(x)$.

Recall that since $w(x) \cdot f(x) = q \bmod F(x)$, we know that $(\zeta_i, \frac{d}{f(\zeta_i)})$ are valid points on $w(x)$. Thus, one can construct $w(x)$ in a whole as computing

$$\text{Lag}\left((\zeta_1, \frac{q}{f(\zeta_1)}), (\zeta_2, \frac{q}{f(\zeta_2)}), \ldots, (\zeta_n, \frac{q}{f(\zeta_n)})\right),$$

where Lag() is the Lagrange interpolation function.

However, also notice that $w_i = -aw_{i+1} \bmod q$ for $0 \leq i \leq n-2$, therefore we only need to compute two non-zero coefficients of $w(x)$, rather than the whole interpolated polynomial. Now let's take a deep look at the interpolation:

$$w(x) \equiv \frac{q}{f(\zeta_1)} \times (\prod_{j \neq 1} \frac{x - \zeta_j}{\zeta_1 - \zeta_j}) + \cdots + \frac{q}{f(\zeta_n)} \times (\prod_{j \neq n} \frac{x - \zeta_i}{\zeta_n - \zeta_j}) \bmod (2^n + 1)$$

therefore, the leading coefficient is

$$w_{n-1} \equiv \frac{q}{f(\zeta_1)} \times (\prod_{j \neq 1} \frac{1}{\zeta_1 - \zeta_j}) + \cdots + \frac{q}{f(\zeta_n)} \times (\prod_{j \neq n} \frac{1}{\zeta_n - \zeta_j}) \bmod (2^n + 1)$$

and the constant coefficient is

$$w_0 \equiv \frac{q}{f(\zeta_1)} \times (\prod_{i \neq 1} \frac{-\zeta_j}{\zeta_1 - \zeta_j}) + \cdots + \frac{q}{f(\zeta_n)} \times (\prod_{i \neq n} \frac{-\zeta_j}{\zeta_n - \zeta_j}) \bmod (2^n + 1)$$

To ease the notation, denote

$$a_i = \frac{1}{\prod_{j \neq i}(\zeta_i - \zeta_j)} \bmod (2^n + 1),$$
$$b_i = \frac{\prod_{j \neq i} \zeta_i}{\prod_{j \neq i}(\zeta_i - \zeta_j)} \bmod (2^n + 1).$$

Then,

$$w_{n-1} = \frac{q}{f(\zeta_1)} \times a_1 + \frac{q}{f(\zeta_2)} \times a_2 + \cdots + \frac{q}{f(\zeta_n)} \times a_n \mod (2^n + 1),$$
$$w_0 = \frac{q}{f(\zeta_1)} \times b_1 + \frac{q}{f(\zeta_2)} \times b_2 + \cdots + \frac{q}{f(\zeta_n)} \times b_n \mod (2^n + 1).$$

Now, if w_0 is invertible over $\mathbb{Z}_q$, then from w_0 and w_{n-1}, and having the relation $w_i = -\beta w_{i+1} \bmod q$, one is able to recover the whole $w(x)$ as well as β.

5.2 Forming the final lattice

In our scheme, we derive the final principal ideal lattice from many smaller principal ideal lattices. We start with finding k small principal ideal lattices $\{f_i(x), \beta_i, q_i\}$ where $f_i(x)$ are sparse binary (or trinary) polynomials. Then, we set $q = LCM(q_1, \ldots, q_k)$, i.e., q is the least common multiple of $q_1, \ldots, q_k$. We derive β and $f(x)$ by applying the Chinese reminder theorem. For simplification, we assume all q_i are pair wise co-prime, in which case we have,

$$q = \prod_{i=1}^{k} q_i,$$
$$f = \prod_{i=1}^{k} f_i,$$
$$\beta = \mathrm{CRT}(\langle \beta_1, \ldots, \beta_k \rangle, \langle q_1, \ldots, q_k \rangle).$$

This assures us that $F(\beta) \equiv f(\beta) \equiv 0 \bmod q$.

Lemma 1. *Let $\mathcal{L}_1, \ldots, \mathcal{L}_k$ be principal ideal lattices whose determinants are pairwise co-prime, and let $f_1(x), \ldots, f_k(x)$ be generators. Then $\prod_{i=1}^{k} f_i(x) \bmod F(x)$ generates a principal ideal lattice $\mathcal{L}$. In particular $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2 \ldots \mathcal{L}_k$.*

Proof. Elementary.

Remark 2. This is a somewhat new method for generating principal ideal lattices efficiently. In our method, we find many principal ideal sup-lattices first, and then we derive the final lattice by intersecting those sup-lattices. Our method is fully compatible with previous methods, i.e., we can use either [14] or [26] to find those sup-lattices $(\{f_i, \beta_i, q_i\})$.

It is interesting to note that when generating a principal ideal lattice, many operations can be pre-computed, such as computing a_i and b_i. Once computed, they can be used through out the generating of all principal ideal lattices. What remains to be costly is to compute the resultant of two polynomials. To accelerate this computation, we use very sparse polynomials.

5.3 The algorithm

Remark 3. The generator polynomial $f(x)$ in our selection is very sparse. We make this choice for two reasons. Firstly, it makes evaluation of $f(\zeta_i)$ very efficient. Evaluating a binary polynomial at a power of two is merely reordering the position of binaries. Although this makes a minor difference as this evaluation is negligible compare with other costs. Secondly, and most importantly, it makes computing the resultant much faster. This is the main reason our new technique is faster than all previous methods.

5.4 Security consideration of key generation

We wish to compare the lattice generated using our approach and generic principal ideal lattices. Note that in the actual scheme, q is kept secret from the attacker. However, for comparison's sake, we assume that q is known.

Algorithm 1 Generate a principal ideal lattice with a large determinant

Input: parameters t, T and n
Output: a principal ideal lattice $\mathcal{L}$;

{Beginning of pre-processing}
Find all the roots $\{\zeta_1, \ldots, \zeta_n\}$ of $x^n + 1$ over $\mathbb{Z}_{2^n+1}$
for $i = 1$ **to** n **do**
    Compute $a_i = \frac{1}{\prod_{j \neq i}(\zeta_i - \zeta_j)}$ over $\mathbb{Z}_{2^n+1}$
    Compute $b_i = \frac{\prod_{j \neq i} \zeta_i}{\prod_{j \neq i}(\zeta_i - \zeta_j)}$ over $\mathbb{Z}_{2^n+1}$
end for
{End of pre-processing}
Set $q = 1$, $f(x) = 1$, $N = \lfloor \frac{T}{t} \rceil$ and $k = 1$
while $i \leq k$ **do**
    Sample a sparse binary polynomial $f_i(x)$ with $|\boldsymbol{f}_i|_2 \leq 2^t$
    Set $q_k = \text{resultant}(f_i(x), F(x))$
    if $\gcd(q_i, q) = 1$ **then**
        $w_{i,0} = \frac{q}{f_i(\zeta_1)} \times b_1 + \frac{q}{f_i(\zeta_2)} \times b_2 + \cdots + \frac{q}{f_i(\zeta_n)} \times b_n \bmod (2^n + 1)$
        if $\gcd(q_i, w_{i,0}) = 1$ **then**
            $w_{i,n-1} = \frac{q}{f_i(\zeta_1)} \times a_1 + \frac{q}{f_i(\zeta_2)} \times a_2 + \cdots + \frac{q}{f_i(\zeta_n)} \times a_n \bmod (2^n + 1)$
            $i = i + 1$
            $d = d_i \times d$
            $\beta_i = (\frac{w_{i,n-1}}{w_0})^n \bmod q_i$
        end if
    end if
end while
$f(x) = \prod_{i=1}^{N} f_i(x) \bmod (x^n + 1)$
$\beta = \text{CRT}((\beta_1, \beta_2, \ldots, \beta_N), (q_1, q_2, \ldots, q_N))$
return $f(x), \beta, q$

In a principal idea lattice based cryptosystem, where the public key is given in the form of a bad basis (often β and q), it is crucial that the attacker is not able to find vector shorter than certain bound in this lattice. While the ability to recover a short vector implies the ability to solve an (approximate) shortest vector problem for a principal ideal lattice in the worst cases, it is not the case for the principal ideal lattice generated using our technique, when assuming factorization is easy.

With our method, the determinant of the lattice is a composite number that can be factorized into k integers of similar length. An attacker cannot distinguish a principal ideal lattice generated using our approach from a random one if he/she cannot factorize. However, to preserve the quantum-safe feature of lattice based cryptography, we also give the attacker the ability to factorize q. Therefore, we must ensure that the attacker cannot derive $f_i(x)$ from (q_i, β_i). In other words, we must ensure that the attacker cannot solve the approximate shortest vector problem for those small lattices.

Note that the hardness of an approximate shortest vector problem is determined by $\left(\frac{\log q_i}{\log |\boldsymbol{f}_i|}\right)^{1/n}$, and we have $\frac{\log q_i}{\log |\boldsymbol{f}_i|} \sim \frac{\log q}{\log |\boldsymbol{f}|}$. The attacker does not gain significant advantage solving short vector problems for small lattices, since the ratio is approximately constant. In addition, the attacker needs to solve the approximate SVP for most of the k small lattices. So it is safe to assume that the principal ideal lattice generated using our method does not have significant weakness compared with previous methods.

We stress again that these considerations are only relevant if the modulus q is revealed. We will return to this analysis in the future if there appears to be an advantage to doing this.

5.5 Bootstrapping

We would like to say a few words on bootstrapping. Our scheme is a homomorphic system over $\mathbb{Z}_q$ that can be broken into many subsystems over $\mathbb{Z}_{q_i}$. This modification does not reduces the number of homomorphic multiplications that we can do, but it reduces the decryption circuit depth (roughly from $\log q$ to $\log q_i$). It can be seen as a squashing technique without the reliance on sparse subset

sum problem as in Gentry's original construction. Hence, it is very interesting to see how bootstappable our scheme is.

References

1. Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011.
2. Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from Ring-LWE and security for key dependent messages. In *CRYPTO*, pages 505–524, 2011.
3. Y. Chen and P. Q. Nguyen. BKZ 2.0: Better lattice security estimates. In *ASIACRYPT*, pages 1–20, 2011.
4. Y. Chen and P. Q. Nguyen. Faster algorithms for approximate common divisors: Breaking fully-homomorphic-encryption challenges over the integers. In *EUROCRYPT*, pages 502–519, 2012.
5. H. Cohn and N. Heninger. Approximate common divisors via lattices. *IACR Cryptology ePrint Archive*, 2011:437, 2011.
6. J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In *CRYPTO*, pages 487–504, 2011.
7. J.-S. Coron, D. Naccache, and M. Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In *EUROCRYPT*, pages 446–464, 2012.
8. S. Garg. Program obfuscation via multilinear maps. In *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, pages 91–94, 2014.
9. S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices. 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*, pages 1–17, 2013.
10. R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO*, pages 465–482, 2010.
11. C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.
12. C. Gentry. *A Fully Homomorphic Encyrption Scheme*. PhD thesis, Stanford University, September 2009.
13. C. Gentry. Computing arbitrary functions of encrypted data. *Commun. ACM*, 53(3):97–105, 2010.
14. C. Gentry and S. Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In *EUROCRYPT*, pages 129–148, 2011.
15. C. Gentry, S. Halevi, C. Peikert, and N. P. Smart. Ring switching in BGV-style homomorphic encryption. In *SCN*, pages 19–37, 2012.
16. S. Halevi and V. Shoup. Algorithms in helib. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 554–571, 2014.
17. K. E. Lauter. Practical applications of homomorphic encryption. In *Proceedings of the 2012 ACM Workshop on Cloud computing security, CCSW 2012, Raleigh, NC, USA, October 19, 2012.*, pages 57–58, 2012.

18. L. Lovász. *An Algorithmic Theory of Numbers, Graphs and Convexity*, volume 50 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM Publications, 1986.
19. D. Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Computational Complexity*, 16(4):365–411, 2007.
20. D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems, A Cryptographic Perspective*. Kluwer Academic Publishers, 2002.
21. H. Minkowski. *Geometrie der Zahlen*. B. G. Teubner, Leipzig, 1896.
22. M. Naehrig, K. E. Lauter, and V. Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Cloud Computing Security Workshop, CCSW 2011, Chicago, IL, USA, October 21, 2011*, pages 113–124, 2011.
23. P. Q. Nguyen. Breaking fully-homomorphic-encryption challenges. In *CANS*, pages 13–14, 2011.
24. T. Plantard, W. Susilo, and Z. Zhang. LLL for ideal lattices: re-evaluation of the security of GentryHalevi's FHE scheme. *Designs, Codes and Cryptography*, pages 1–20, 2014.
25. R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–177. Academic Press, 1978.
26. P. Scholl and N. P. Smart. Improved key generation for gentry's fully homomorphic encryption scheme. In *IMA Int. Conf.*, pages 10–22, 2011.
27. N. P. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Public Key Cryptography*, pages 420–443, 2010.
28. D. Stehlé and R. Steinfeld. Faster fully homomorphic encryption. In *ASIACRYPT*, pages 377–394, 2010.
29. M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT*, pages 24–43, 2010.