Introduction to Homomorphic Encryption

Kyle Yates

Clemson University

June 23, 2022

Cryptography

Suppose there are two people, **Alice** and **Bob**. Alice wants to send Bob a message over an insecure channel, but wants only Bob to read the message she sends.

If Alice just sends the message plainly over an insecure channel, then a third party can intercept the transmission and read the message! How can Alice send Bob this message without anybody else reading it?

Cryptography

Instead of sending the message plainly, Alice is going to **encrypt** the message using some information called a **key**, and then send the message to Bob.

When Bob receives the message, he will **decrypt** the information with a secret key, and can then read the message. However, if a third party without the secret key intercepts the message, they can not decrypt and read the message.

When Alice and Bob share the same secret key, we call this **symmetric key cryptography**. When the key Alice uses is different from Bob and is public information, we call this **public key cryptography**.

Cryptography (More Formally)

Let $\mathtt{Enc} : \mathcal{K}_1 \times \mathcal{M} \times \mathcal{R} \to \mathcal{C}$. For any $m \in \mathcal{M}$ and $k \in \mathcal{K}_1$, one picks a random $r \in \mathcal{R}$ and computes $\mathtt{ct} = \mathtt{Enc}(m, \mathtt{k}, r) \in \mathcal{C}$. We call Enc an **encryption function** or encryption algorithm, m the **message** (or plaintext), k the **key**, and ct the **ciphertext**. When $|\mathcal{R}| = 1$, we call this deterministic encryption. When $|\mathcal{R}| > 1$, we call this probabilistic encryption.

Let $\mathtt{Dec} : \mathcal{C} \times \mathcal{K}_2 \to \mathcal{M}$. For inputs $\mathtt{ct} \in \mathcal{C}$, $\mathtt{sk} \in \mathcal{K}_2$, and corresponding output $m \in \mathcal{M}$, we write $\mathtt{Dec}(\mathtt{ct}, \mathtt{sk}) = m$. We cal Dec a **decryption function** or decryption algorithm. Our goal is to have the following property.

$$\mathtt{Dec}(\mathtt{Enc}(m, \mathtt{k}, r), \mathtt{sk}) = m$$

Homomorphic Encryption

The goal of **homomorphic encryption** is to obtain a homomorphism in the message slot in the encryption function.

For two messages m_0 and m_1 in $\mathcal{M}$, a public or private key $\mathtt{k} \in \mathcal{K}$, and random $r_0, r_1 \in \mathcal{R}$ we have that for some $r_2, r_3 \in \mathcal{R}$,

$$\mathtt{Enc}(m_0, \mathtt{k}, r_0) + \mathtt{Enc}(m_1, \mathtt{k}, r_1) = \mathtt{Enc}(m_0 + m_1, \mathtt{k}, r_2)$$
$$\mathtt{Enc}(m_0, \mathtt{k}, r_0)\mathtt{Enc}(m_1, \mathtt{k}, r_1) = \mathtt{Enc}(m_0 m_1, \mathtt{k}, r_3)$$

This property allows one to perform computations on encrypted data without needing to first decrypt it. Current homomorphic encryption schemes are largely based on error correction.

Notation

Define $\mathbb{Z}_q$ as the ring of centered representatives as $\mathbb{Z}_q := \mathbb{Z} \cap (-q/2, q/2]$. When given an integer x, we denote $[x]_q$ as the reduction of x into the interval $\mathbb{Z}_q$ such that q divides $[x]_q - x$. When x is a polynomial or vector, $[x]_q$ means applying $[\cdot]_q$ to each coefficient.

We define R_n as the ring

$$R_n := \mathbb{Z}[x]/(\Phi(x))$$

where $\Phi(x)$ is an mth cyclotomic polynomial of degree n, a power of two. Namely, $\Phi(x) = x^n + 1$. Similar to R_n, we define $R_{n,q}$ as the ring

$$R_{n,q} := \mathbb{Z}_q[x]/(\Phi(x))$$

Notation

For any $c \in \mathbb{R}$, the *infinity norm of* c is defined as $\|c\|_\infty = |c|$. For any polynomial $f(x) = \sum_{i=0}^{k} a_i x^i$ with $a_i \in \mathbb{R}$, the *infinity norm of* $f(x)$ is defined as

$$\|f(x)\|_\infty = \max\{|a_1|, \dots, |a_k|\}$$

therefore using centered representatives, for any $f(x) \in R_{n,q}$ we have $\|f(x)\|_\infty \leq q/2$.

The symbols $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ will denote floor and ceiling respectively, whereas $\lfloor \cdot \rceil$ will denote rounding to the nearest integer.

For a set S and a given probability distribution χ, we let $\chi(S)$ denote that distribution on S. We will denote $U(S)$ as a uniform distribution on S.

Learning With Errors (LWE)

Homomorphic encryption is based off the **learning with errors (LWE)** problem.

Choose $\mathbf{s} \in \mathbb{Z}_q^n$ secret. Sample $e \leftarrow \chi(\mathbb{Z})$ from some desired distribution such that $|e| \leq \rho$, where ρ is a small parameter. Then, we sample a uniform random $\mathbf{a} \leftarrow U(\mathbb{Z}_q^n)$ and calculate b via

$$b = \langle \mathbf{a}, \mathbf{s} \rangle + e \mod q$$

We call $(\mathbf{a}, b)$ an LWE sample.

LWE Problem: Given many LWE samples, find $\mathbf{s}$.

The LWE problem is at least as hard as many worst-case hard lattice problems.

Ring Learning With Errors (RLWE)

We can define an analogous problem with rings known as the **ring learning with errors (RLWE)** problem.

Choose secret $s \in R_{n,q}$ and sample an error $e \leftarrow \chi(R_n)$ such that $\|e\|_\infty \leq \rho$. Sample $a \leftarrow U(R_{n,q})$ and compute $b \in R_{n,q}$ via

$$b \equiv as + e \mod (\Phi(x), q)$$

The ordered pair $(a, b) \in R^2_{n,q}$ is called an *RLWE sample*. The RLWE problem can be defined in the same way as the LWE problem.

The security of cryptosystems we will use rely on the hardness of RLWE.

Basic Homomorphic Encryption

Three main schemes: BFV and BGV (exact), and CKKS (approximate).

BFV Public Key Generation

First the **BFV** scheme.

Choose $s \in R_{n,3}$ secret. Sample random $a \leftarrow U(R_{n,q})$ and $e \leftarrow \chi(R_n)$ such that $\|e\|_\infty \leq \rho$. Now, compute $b \in R_{n,q}$ via

$$b = -(as + e) \mod (\Phi(x), q)$$

The secret key is $\texttt{sk} = s \in R_{n,3}$. The public key is $\texttt{pk} = (a, b) \in R_{n,q}^2$.

BFV Public Key Encryption

Consider a message $m_0 \in R_{n,t}$ for some $t \in \mathbb{Z}^+$. We then encrypt the message using $\texttt{pk}$, a constant $D = \lfloor q/t \rfloor \in \mathbb{Z}^+$, and a chosen parameter $\rho \in \mathbb{Z}^+$ as follows:

1. Generate a random $u \in R_{n,3}$.
2. Sample $e_0', e_0'' \leftarrow \chi(R_n)$ such that $\|e_0'\|_\infty, \|e_0''\|_\infty \leq \rho$.
3. Compute $a_0 \in R_{n,q}$ with $a_0 \equiv au + e_0' \mod (\Phi(x), q)$ and $b_0 \in R_{n,q}$ with $b_0 \equiv bu + Dm_0 + e_0'' \mod (\Phi(x), q)$.
4. Return $\texttt{ct}_0 = (a_0, b_0)$.

The ordered pair $\texttt{ct}_0 = (a_0, b_0) \in R_{n,q}^2$ is our BFV ciphertext.

Notice that

$$\begin{aligned} b_0 + a_0 s &\equiv asu - asu + Dm_0 + eu + e_0' s + e_0'' \mod (\Phi(x), q) \\ &\equiv Dm_0 + e_0 \mod (\Phi(x), q) \end{aligned}$$

for $e_0 = eu + e_0' + e_0''$. Or equivalently,

$$b_0 \equiv -a_0 s + Dm_0 + e_0 \mod (\Phi(x), q)$$

The ciphertext $\mathtt{ct}_0 = (a_0, b_0) \in R_{n,q}^2$ is essentially an RLWE sample!

BFV Decryption

If we possess the secret key $\mathtt{sk} = s$, can we retrieve the message? The following gives the decryption algorithm.

1. Compute $m_0 = \lfloor \frac{b_0 + a_0 s \mod (\Phi(x), q)}{D} \rceil$.
2. Return m_0.

Is this actually our m_0? Note that a BFV ciphertext has the relationship $b_0 + a_0 s \equiv Dm_0 + e_0 \mod (\Phi(x), q)$, so

$$\left\lfloor \frac{b_0 + a_0 s \mod (\Phi(x), q)}{D} \right\rceil = \left\lfloor m_0 + \frac{e_0}{D} \right\rceil = m_0 + \left\lfloor \frac{e_0}{D} \right\rceil$$

As long as $\|e_0\|_\infty \leq D/2$, we are good!

BGV Public Key Generation

Now for the **BGV** scheme.

Choose a secret $s \in R_{n,q}$ with coefficients in $\{-1, 0, 1\}$. The secret key is $\mathtt{sk} = s \in R_{n,3}$. Sample a uniform random $a \leftarrow U(R_{n,q})$ and a random error polynomial $e \leftarrow \chi(R_n)$ such that $\|e\|_\infty \le \rho$. In BGV, we define the public key $\mathtt{pk} = (a, b) \in R_{n,q}^2$ where

$$b \equiv -(as + te) \mod (\Phi(x), q) \tag{1}$$

BGV Publc Key Encryption

Consider a message $m_0 \in R_{n,t}$ for some $t \in \mathbb{Z}^+$. We then encrypt the message using $\texttt{pk}$ and a chosen parameter $\rho \in \mathbb{Z}^+$ as follows:

1. Generate a random $u \in R_{n,3}$.
2. Sample $e_0', e_0'' \leftarrow \chi(R_n)$ such that $\|e_0'\|_\infty, \|e_0''\|_\infty \leq \rho$.
3. Compute $a_0 \in R_{n,q}$ with $a_0 \equiv au + te_0' \mod (\Phi(x), q)$ and $b_0 \in R_{n,q}$ with $b_0 \equiv bu + m_0 + te_0'' \mod (\Phi(x), q)$.
4. Return $\texttt{ct}_0 = (a_0, b_0)$.

The ordered pair $\texttt{ct}_0 = (a_0, b_0) \in R_{n,q}^2$ is our BGV ciphertext.

Observe that

$$\begin{aligned} b_0 + a_0 s &\equiv bu + m_0 + te_0'' + (au + te_0')s \mod (\Phi(x), q) \\ &\equiv -(as + te)u + m_0 + te_0'' + (au + te_0')s \mod (\Phi(x), q) \\ &\equiv m_0 + t(e_0'' + e_0's - eu) \mod (\Phi(x), q) \end{aligned}$$

Let $e_0 = e_0'' + e_0's - eu$. Then, we simply have

$$b_0 + a_0 s \equiv m_0 + te_0 \mod (\Phi(x), q) \tag{2}$$

BGV Decryption

Decryption is given by the following

1. Compute $m_0 = (b_0 + a_0 s \mod (\Phi(x), q)) \mod t$.
2. Return m_0.

Again, is this actually m_0?

$$\begin{aligned}(b_0 + a_0 s \mod (\Phi(x), q)) \mod t &\equiv (te_0 + m_0 \mod (\Phi(x), q)) \mod t \\ &\equiv te_0 + m_0 \mod t \\ &\equiv m_0 \mod t\end{aligned}$$

As long as $te_0 + m_0 \in R_{n,q}$, this works! We require $\|te_0 + m_0\|_\infty \le q/2$, or $\|e_0\|_\infty \le q/2t$ (approximately).

The differences in the homomorphic encryption schemes mostly lies in the relationship of a_0 and b_0.

$$\begin{aligned}
\text{BFV}: &\ b_0 + a_0 s \equiv Dm_0 + e_0 \mod (\Phi(x), q) \\
\text{BGV}: &\ b_0 + a_0 s \equiv m_0 + te_0 \mod (\Phi(x), q) \\
\text{CKKS}: &\ b_0 + a_0 s \equiv m_0 + e_0 \mod (\Phi(x), q)
\end{aligned}$$

BGV Addition

Given two BGV ciphertexts $\texttt{ct}_0 = (a_0, b_0)$ and $\texttt{ct}_1 = (a_1, b_1)$, recall they satisfy

$$b_0 + a_0 s \equiv m_0 + te_0 \mod (\Phi(x), q) \tag{3}$$
$$b_1 + a_1 s \equiv m_1 + te_1 \mod (\Phi(x), q) \tag{4}$$

Then, addition can be done via

1. Compute $\texttt{ct}_2 = \texttt{ct}_0 + \texttt{ct}_1 \mod q$.
2. Return $\texttt{ct}_2$.

There are two questions we should be thinking about: Does this actually work? What happens to the error?

Does it work? Let r be the integer such that $m_0 + m_1 = [m_0 + m_1]_t + tr$. Now, notice that

$$\texttt{ct}_2 = \texttt{ct}_1 + \texttt{ct}_2 \mod q = (a_0 + a_1, b_0 + b_1) \mod q$$

Then,

$$\begin{aligned}(b_0 + b_1) + (a_0 + a_1)s &\equiv m_0 + m_1 + t(e_0 + e_1) \mod (\Phi(x), q)\\ &\equiv [m_0 + m_1]_t + t(e_0 + e_1) + tr \mod (\Phi(x), q)\\ &\equiv [m_0 + m_1]_t + te_{\text{add}} \mod (\Phi(x), q)\end{aligned}$$

What happens to the error? We have that $\|r\|_\infty \leq 1$, so

$$\|e_{\text{add}}\|_\infty = \|e_0 + e_1 + r\|_\infty \leq \|e_0\|_\infty + \|e_1\|_\infty + 1$$

BFV Addition

BFV addition is very similar to BGV. Take two BFV ciphertexts, (a_0, b_0) and $(a_1, b_1) \in R_{n,q}^2$ where

$$\begin{aligned} b_0 + a_0 s &\equiv Dm_0 + e_0 \mod (\Phi(x), q) \\ b_1 + a_1 s &\equiv Dm_1 + e_1 \mod (\Phi(x), q) \end{aligned}$$

Let $r \in R_{n,q}$ such that $m_0 + m_1 = [m_0 + m_1]_t + tr$ and $\epsilon = q/t - D$. We have that

$$\begin{aligned} (b_0 + a_0 s) + (b_1 + a_1 s) &\equiv D(m_0 + m_1) + e_0 + e_1 \mod q \\ &\equiv D[m_0 + m_1]_t + e_0 + e_1 + Dtr \mod q \\ &\equiv D[m_0 + m_1]_t + e_0 + e_1 - \epsilon tr \mod q \\ &\equiv D[m_0 + m_1]_t + e_{\text{add}} \mod q \end{aligned}$$

BGV Multiplication

Multiplication is a little bit more involved. We'll start with BGV again.

Given two BGV ciphertexts $\mathtt{ct}_0 = (a_0, b_0)$ and $\mathtt{ct}_1 = (a_1, b_1)$, recall they satisfy

$$b_0 + a_0 s \equiv m_0 + te_0 \mod (\Phi(x), q) \tag{5}$$

$$b_1 + a_1 s \equiv m_1 + te_1 \mod (\Phi(x), q) \tag{6}$$

Step 1. Compute the following:

1. $c'_0 = b_0 b_1 \mod (\Phi(x), q)$
2. $c'_1 = b_1 a_0 + b_0 a_1 \mod (\Phi(x), q)$
3. $c'_2 = a_0 a_1 \mod (\Phi(x), q)$

... but why?

Let $r_m \in R_{n,q}$ such that $m_0 m_1 \equiv [m_0 m_1]_t + t r_m \mod \Phi(x)$. Now, we have

$$\begin{aligned} c_0' + c_1' s + c_2' s^2 &\equiv b_0 b_1 + (b_1 a_0 + b_0 a_1)s + a_0 a_1 s^2 \mod (\Phi(x), q) \\ &\equiv (b_0 + a_0 s)(b_1 + a_1 s) \mod (\Phi(x), q) \\ &\equiv (m_0 + t e_0)(m_1 + t e_1) \mod (\Phi(x), q) \\ &\equiv m_0 m_1 + t(m_0 e_1 + m_1 e_0 + t e_0 e_1) \mod (\Phi(x), q) \\ &\equiv [m_0 m_1]_t + t(m_0 e_1 + m_1 e_0 + t e_0 e_1 + r_m) \mod (\Phi(x), q) \\ &\equiv [m_0 m_1]_t + t e^* \mod (\Phi(x), q) \end{aligned}$$

We like most of this. c_0', c_1', c_2' can all be computed only with a_0, a_1, b_0, b_1. And, using the secret key s, we can retrieve a multiplication of messages $m_0 m_1$.

What do we not like? The s^2 term! :'(

Relinearization

We want to compute polynomials $d_0, d_1 \in R_{n,q}$ such that $d_0 + d_1 s$ is approximately $c_2' s^2$ modulo $(\Phi(x), q)$. Or,

$$d_0 + d_1 s \equiv c_2' s^2 + te'' \mod (\Phi(x), q) \tag{7}$$

for some small error $e'' \in R_n$. If we do this, then we will have

$$c_0' + c_1' s + c_2' s^2 \equiv (c_0' + d_0) + (c_1' + d_1)s + te'' \mod (\Phi(x), q)$$

and therefore

$$(c_0' + d_0) + (c_1' + d_1)s \equiv [m_0 m_1]_t + t(e^* - e'') \mod (\Phi(x), q)$$

For any $B \in \mathbb{Z}^+$, with the smallest $\gamma \in \mathbb{Z}$ such that $B^\gamma > q$, we write

$$c_2' = \sum_{j=0}^{\gamma-1} h_j B^j$$

where $h_j \in R_{n,q}$ such that $\|h_j\|_\infty \leq B/2$. Let $\mathbf{h}$ be the $1 \times \gamma$ vector

$$\mathbf{h} = \left(h_0, h_1, \dots, h_{\gamma-1}\right)$$

and $\mathbf{g}$ be the $\gamma \times 1$ vector

$$\mathbf{g} = \left(1, B, B^2, \dots, B^{\gamma-1}\right)^T$$

Then, $c_2' s^2 = (\mathbf{hg})s^2 = (\sum_{j=0}^{\gamma-1} h_j B^j)s^2$. Let $\mathbf{u}$ a $\gamma \times 1$ vector where each entry is drawn uniform randomly from $R_{n,q}$. Sample a random $\gamma \times 1$ vector $\mathbf{w}$, where each entry of $\mathbf{w}$ is in R_n and bounded by ρ. Let $\mathbf{v}$ be the $\gamma \times 1$ vector

$$\mathbf{v} = s^2\mathbf{g} - \mathbf{u}s + t\mathbf{w}$$

where each entry is computed modulo $(\Phi(x), q)$. We call the vector pair $(\mathbf{u}, \mathbf{v})$ the *evaluation key*, which we denote $\texttt{evk}_{\text{flat}} = (\mathbf{u}, \mathbf{v})$.

Notice that for the jth coordinate in the vector pair $(\mathbf{u}, \mathbf{v})$, we have the relationship

$$v_j + u_j s = s^2 B^j + t w_j \mod (\Phi(x), q)$$

As the equation above is a simple RLWE encryption of $s^2 B^j$, we can publish the evaluation key without compromising security of the secret key s.

Notice,

$$\begin{aligned}
c_2' s^2 &= (\mathbf{hg}) s^2 \\
&= \Big(\sum_{j=0}^{\gamma-1} h_j B^j \Big) s^2 \\
&\equiv \sum_{j=0}^{\gamma-1} h_j (v_j + u_j s - t w_j) \mod (\Phi(x), q)
\end{aligned}$$

Then, we have

$$\begin{aligned}
c_2' s^2 + t \sum_{j=0}^{\gamma-1} h_j w_j &\equiv \sum_{j=0}^{\gamma-1} h_j v_j + \Big(\sum_{j=0}^{\gamma-1} h_j u_j \Big) s \mod (\Phi(x), q) \\
&\equiv \mathbf{hv} + (\mathbf{hu}) s \mod (\Phi(x), q)
\end{aligned}$$

Or equivalently, $c_2' s^2 + t\mathbf{hw} \equiv d_0 + d_1 s \mod (\Phi(x), q)$.

Finally... we have that

$$(c_0' + \mathbf{hv}) + (c_1' + \mathbf{hu})s \equiv [m_0 m_1]_t + te_{\text{mult}} \mod (\Phi(x), q)$$

where $e_{\text{mult}} = e^* - \mathbf{hw}$. This is our final result from multiplication. Better yet, everything on the left hand side we can compute with public information.

We also know bounds for the error term on the right.

BFV Multiplication

BFV multiplication is even more of a nightmare. Now, we are working with ciphertexts such that

$$
\begin{aligned}
b_0 + a_0 s &\equiv Dm_0 + e_0 \mod (\Phi(x), q) \\
b_1 + a_1 s &\equiv Dm_1 + e_1 \mod (\Phi(x), q)
\end{aligned}
$$

The issue is now, if we try and multiply these together, we get

$$(b_0 + a_0 s)(b_1 + a_1 s) \equiv (Dm_0 + e_0)(Dm_1 + e_1) \mod (\Phi(x), q)$$

$$\equiv D^2 m_0 m_1 + Dm_0 e_1 + Dm_1 e_0 + e_0 e_1 \mod (\Phi(x), q)$$

Loosely speaking, we have something of the form $D^2 m_0 m_1$, when we need to have $Dm_0 m_1$.

How do we deal with this? The idea is we convert our relationships to integer equations, and then divide by (approximately) D and round our entries to integers. First, we write

$$(b_0 + a_0 s)(b_1 + a_1 s) \equiv (Dm_0 + e_0 + qr_0)(Dm_1 + e_1 + qr_1) \mod \Phi(x)$$

for some $r_0, r_1 \in R_n$. Then, we multiply everything by t/q.

The algebra to do this is not nice :'(

$$(t/q)(c_0 + c_1 s + c_2 s^2)$$

$$\begin{aligned}
&\equiv (tD^2/q)[m_0 m_1]_t + (tD^2/q)2tr_m + (tD/q)(m_0 e_1 + m_1 e_0) \\
&+ (tq/q)(e_0 r_1 + r_0 e_1) + (t/q)[e_0 e_1]_D + (tD/q)r_e \\
&+ (tqD/q)(m_0 r_1 + r_0 m_1) + (tq^2/q)r_0 r_1 \mod \Phi(x)
\end{aligned}$$

$$\begin{aligned}
&\equiv ((q - r_t(q))D/q)[m_0 m_1]_t + ((q - r_t(q))D/q)2tr_m \\
&+ ((q - r_t(q))/q)(m_0 e_1 + m_1 e_0) + (t)(e_0 r_1 + r_0 e_1) \\
&+ (t/q)[e_0 e_1]_D + ((q - r_t(q))/q)r_e + (q - r_t(q))(m_0 r_1 + r_0 m_1) \\
&+ (tq^2/q)r_0 r_1 \mod \Phi(x)
\end{aligned}$$

Believe it or not, this does clean up pretty nicely. The final result is

$$c_0' + c_1's + c_2's^2 \equiv D[m_0m_1]_t + e^* \mod (\Phi(x), q)$$

where $c_0' = \lfloor tc_0/q \rceil, c_1' = \lfloor tc_1/q \rceil$, and $c_2' = \lfloor tc_2/q \rceil$ and the error term is

$$e^* = (m_0e_1+m_1e_0)+t(e_0r_1+r_0e_1)+r_e+(-r_t(q))(r_m+m_0r_1+r_0m_1)+r_r-r_a$$

From here, we can do a similar type of relinearization process.

Error Control

Recall that to decrypt a ciphertext, we need to have error $< D/2$. When we do operations on ciphertexts, the error grows. How do we manage the ciphertext error?

We will introduce the technique known as **modulus reduction** in order to do this.

Let $q < Q$ integers. Given an integer modulus Q in the ciphertext, we will reduce the ciphertext to a modulus q ciphertext while also reducing the error.

For the following, let $D_Q = \lfloor Q/t \rfloor$ and $D_q = \lfloor q/t \rfloor$.

Input: $Q \in \mathbb{Z}^+$ an integer, $q \in \mathbb{Z}^+$ an integer, and $\texttt{ct}_0 = (a_0, b_0) \in R_{n,Q}^2$ BFV ciphertext such that $b_0 + a_0 s \equiv D_Q m_0 + e_0 \mod (\Phi(x), Q)$.

1. Compute $a_0' = \lfloor \frac{q a_0}{Q} \rceil$ and $b_0' = \lfloor \frac{q b_0}{Q} \rceil$.
2. Return $\texttt{ct}_0' = (a_0', b_0') \in R_{n,q}^2$ such that $b_0' + a_0' s \equiv D_q + e_{\mathrm{MR}}$ $\mod (\Phi(x), q)$ for some e_{MR}.

We have the same two questions as before: Does this actually work? What happens to the error?

Let $\epsilon_Q = Q/t - D_Q$, $\epsilon_q = q/t - D_q$, $\epsilon_{a_0} = qa_0/Q - a_0'$, and $\epsilon_{b_0} = qb_0/Q - b_0'$. By assumption, we first note that $(a_0, b_0) \in R_{n,Q}$ is a BFV ciphertext. That is,

$$b_0 \equiv -a_0 s + D_Q m_0 + e_0 \mod (\Phi(x), Q)$$

Therefore, there is some integer $r_Q \in \mathbb{Z}$ such that $b_0 + a_0 s \equiv D_Q m_0 + e_0 + Qr_Q \mod \Phi(x)$. Then,

$$\begin{aligned} b_0' &= \frac{qb_0}{Q} - \epsilon_{b_0} \\ &\equiv -\frac{qa_0 s}{Q} + \frac{qD_Q}{Q} m_0 + \frac{qe_0}{Q} - \epsilon_{b_0} + qr_Q \mod \Phi(x) \end{aligned}$$

Note that as $D_Q = Q/t - \epsilon_Q$, we have that $qD_Q/Q = q/t - q\epsilon_Q/Q$. Since $q/t = D_q + \epsilon_q$, we have $qD_Q/Q = D_q + \epsilon_q - q\epsilon_Q/Q$.

Therefore,

$$\begin{aligned} b_0' &\equiv -\frac{qa_0 s}{Q} + \frac{qD_Q}{Q} m_0 + \frac{qe_0}{Q} - \epsilon_{b_0} + qr_Q \mod \Phi(x) \\ &\equiv -a_0' s + \epsilon_{a_0} s + D_q m_0 + (\epsilon_q - \frac{q\epsilon_Q}{Q}) m_0 + \frac{qe_0}{Q} - \epsilon_{b_0} + qr_Q \mod \Phi(x) \\ &\equiv -a_0' s + D_q m_0 + e_{\mathrm{MR}} \mod (\Phi(x), q) \end{aligned}$$

where

$$e_{\mathrm{MR}} = \frac{qe_0}{Q} + (\epsilon_q - q\epsilon_Q/Q) m_0 - \epsilon_{b_0} + \epsilon_{a_0} s$$

Therefore, $b_0' + a_0' s \equiv D_q m_0 + e_{\mathrm{MR}} \mod (\Phi(x), q)$.

What about the error?

Since $Q > q$, then we have that $0 < q/Q < 1$. Noting that as $Q/t = D_Q + \epsilon_Q$ and $Q/t \geq D_Q$, then $0 \leq \epsilon_Q \leq 1$. Similarly, $0 \leq \epsilon_q \leq 1$. Then $|\epsilon_q - q\epsilon_Q/Q| \leq 1$. So,

$$\begin{aligned}
\|e_{\mathrm{MR}}\|_\infty &\leq \left\| \frac{qe_0}{Q} + (\epsilon_q - q\epsilon_Q/Q)m_0 - \epsilon_{b_0} + \epsilon_{a_0} s \right\|_\infty \\
&\leq \frac{q}{Q}\|e_0\|_\infty + |\epsilon_q - q\epsilon_Q/Q|\frac{t}{2} + \|\epsilon_{b_0}\|_\infty + \|\epsilon_{a_0} s\|_\infty \\
&\leq \frac{q}{Q}E + \frac{t+1}{2} + \frac{\delta_R \|s\|_\infty}{2}
\end{aligned}$$

Modulus Leveling

Let $q_{\ell+1} > q_\ell > \cdots > q_0$ be distinct primes, and define $Q_0, \ldots, Q_{\ell+1}$ as

$$Q_i = \prod_{j=0}^{i} q_j$$

We call Q_i the modulus at level i. It is easy to see that $Q_i / Q_{i-1} = q_i$ for any $i \in \{1, \ldots, \ell + 1\}$. The idea is that we will periodically reduce the modulus from Q_i to Q_{i-1} to reduce the error.

We know that performing modulus reduction produces a ciphertext with e_{MR} such that

$$\begin{aligned}\|e_{\mathrm{MR}}\|_\infty &< \frac{Q_{i-1}}{Q_i}E + \frac{t+1}{2} + \frac{\delta_R \|s\|_\infty}{2} \\ &= \frac{1}{q_i}E + \frac{t+1}{2} + \frac{\delta_R \|s\|_\infty}{2}\end{aligned}$$

or equivalently,

$$q_i > \frac{2E}{2\|e_{\mathrm{MR}}\|_\infty - t - 1 - \delta_R \|s\|_\infty}$$

If we choose some constant $C > \|e_{\mathrm{MR}}\|_\infty$ in the equation above to replace $\|e_{\mathrm{MR}}\|_\infty$, then we can guarantee that modulus reduction will always return an error bounded by C so long as this bound on q_i above is met.

Techniques in Efficiency (Thesis)

-Chinese Remainder Theorem. Break down Q into individual coprime pieces and do operations component-wise. Tricky part here is doing computations that we did in $\mathbb{Q}$ or $\mathbb{R}$.

-Fast Fourier Transforms for the actual polynomial-polynomial multiplication

-Improving error bounds to allow for more computations.

References (1/3)

[1] Matthew Baker. Algebraic number theory. Course Notes Math 8803, Georgia Tech, 2006. http://www.math.toronto.edu/~ila/ANTBook.pdf.

[2] Dan Boneh and Victor Shoup. A graduate course in applied cryptography. Textbook, 2020. http://toc.cryptobook.us/book.pdf.

[3] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Report 2011/277, 2011. https://ia.cr/2011/277.

[4] Benjamin Case. Homomorphic encryption and cryptanalysis of lattice cryptography. All Dissertations. 2635., 2020. https://tigerprints.clemson.edu/all_dissertations/2635.

[5] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. A full rns variant of approximate homomorphic encryption. Cryptology ePrint Archive, Report 2018/931, 2018. https://ia.cr/2018/931.

[6] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. Cryptology ePrint Archive, Report 2016/421, 2016. https://ia.cr/2016/421.

References (2/3)

[7] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[8] Anamaria Costache and Nigel P. Smart. Which ring based somewhat homomorphic encryption scheme is best? Cryptology ePrint Archive, Report 2015/889, 2015. https://ia.cr/2015/889.

[9] Xinyue Deng. An introduction to lenstra-lenstra-lovasz lattice basis reduction algorithm. Massachusetts Institute of Technology, 2016. https://math.mit.edu/~apost/courses/18.204-2016/18.204_Xinyue_Deng_final_paper.pdf.

[10] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. https://ia.cr/2012/144.

[11] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.

[12] Shai Halevi, Yuriy Polyakov, and Victor Shoup. An improved rns variant of the bfv homomorphic encryption scheme. Cryptology ePrint Archive, Report 2018/117, 2018. https://ia.cr/2018/117.

[13] M.T. Heath. *Scientific Computing: An Introductory Survey*. McGraw-Hill Education, 2005.

[14] Kenneth F. Ireland. *A classical introduction to modern number theory / Kenneth Ireland, Michael, Michael Rosen*. Graduate texts in mathematics ; 84. Springer, New York, 1982.

References (3/3)

[15] Eunsang Lee, Joon-Woo Lee, Young-Sik Kim, and Jong-Seon No. Optimization of homomorphic comparison algorithm on rns-ckks scheme. Cryptology ePrint Archive, Report 2021/1215, 2021. `https://ia.cr/2021/1215`.

[16] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), sep 2009.

[17] S. Roberts. Lecture 7-the discrete fourier transform. Textbook, 2020. `http://www.robots.ox.ac.uk/~sjrob/Teaching/SP/l7.pdf`.

[18] Voyko Flores Rocha and Julio López. An overview on homomorphic encryption algorithms. 2019.