

# Analysis of Somewhat Homomorphic Encryption Over the Integer Ring

Bryan Kaperick

January 16, 2017

# 1 Preliminaries

## 1.1 Symmetric Modulus

Traditionally, the modulus operator can be defined as follows

**Definition 1.1.** Define  $q_a(b) = \lfloor \frac{b}{a} \rfloor$ . Then, define  $a \pmod b = a - q_a(b)b$ , which is equivalent to setting  $a \pmod b$  to be the representative in  $[0, b)$  for the residue class containing  $a$  for the congruence relation of congruence modulo  $b$ .

However, for the purposes of this paper, it will be seen that a slightly altered definition is much more convenient.

**Definition 1.2.** Define  $q_a(b) = \lfloor \frac{a}{b} \rceil$ , where  $\lfloor \cdot \rceil$  returns the nearest integer to the input value (rounding up for multiples of one-half). Again, define  $a \pmod b = b - q_a(b)a$ .

While notationally annoying, this approach makes much more sense once the *idea* of this scheme is understood. In general, the scheme relies on recovering a noisy approximation of a multiple of the secret key, so in this respect, it is more natural to allow a symmetric distribution of noisy approximations to all be in the same *class*. More on this later.

## 1.2 Rounding Operator

In these notes it is often necessary to round a number to the nearest integer. The following notation is used,

**Definition 1.3.** Let  $x \in \mathbb{R}$ . Then,  $\lfloor x \rceil$  is equal to the integer closest to  $x$  (rounding down if equidistant).

# 2 Goals of Scheme

This scheme is intended to be a homomorphic encryption scheme equipped to allow evaluation of the encrypted data on arbitrary binary addition and multiplication circuits (up to a pre-determined depth) such that the evaluated data almost surely decrypts correctly.

# 3 Motivation for Approach

The main idea is to map a bit to an arbitrary integer multiple of the secret key — also an integer — with some additional noise added. Let  $\mathcal{S}$  be the space of integer multiples of the secret key,  $s$ . Let  $x, y \in \mathcal{S}$ . Observe that with integer addition and multiplication,  $\mathcal{S}$  forms a ring.

*Proof.*  $\mathcal{S} = \{x | \exists n \in \mathbb{Z}, x = n \cdot s\}$ . Let  $x, y \in \mathcal{S}$ . If  $x = n \cdot s$  and  $y = m \cdot s$  for some  $n, m \in \mathbb{Z}$ , then clearly  $x + y = n \cdot s + m \cdot s = (n + m) \cdot s$ , so the operation is closed. Integer addition is commutative. Every integer  $n \in \mathbb{Z}$  has additive inverse  $-n$ , and both  $n \cdot s$  and  $-n \cdot s$  are in  $\mathcal{S}$ . Clearly  $0 \cdot s$  is in  $\mathcal{S}$ , satisfying conditions for the identity. Thus,  $\mathcal{S}$  is a group under addition.

Multiplication is also closed with respect to the integers, is associative and distributes over addition. 1 satisfies as the identity element. Thus, multiplication acts as the second binary operation, and  $(\mathcal{S}, +, \cdot)$  is a ring.  $\square$

This fact is the foundational motivation behind this scheme. Since adding and multiplying elements of  $\mathcal{S}$  will also be elements of  $\mathcal{S}$ , so the goal is to develop a scheme which maps these operations of  $\mathcal{S}$  to the equivalent operations on the unencrypted bits corresponding to those elements of  $\mathcal{S}$ . The security of the scheme comes from adding noise to the elements of  $\mathcal{S}$  to make the act of retrieving  $s$  difficult.

### 3.1 Noisy Ring $\mathcal{S}_n$

To formalize the notion of noise in this ring, we will discuss a new ring,  $\mathcal{S}_n$ . First, we begin with the set of integers,  $\mathbb{Z}$ . We define a congruence relation on  $\mathbb{Z}$ ,

**Definition 3.1.** Fix  $s \in \mathbb{Z}^+$ . Let  $a, b \in \mathbb{Z}$ . We will say  $a$  is equivalent to  $b$ , or  $a \equiv b$ , if  $q_s(a) = q_s(b)$ . That is, if  $\lfloor \frac{a}{s} \rfloor = \lfloor \frac{b}{s} \rfloor$ . This is equivalent to defining the relation as the following: Decompose  $a$  and  $b$  into  $a = xs + n$  and  $b = ys + m$  for some  $x, y \in \mathbb{Z}$  and  $m, n \in (-s/2, s/2]$ . Then,  $a \equiv b$  if and only if  $x = y$ .

This relation clearly satisfies symmetry, reflexivity and transitivity. The equivalency classes of this relation partition  $\mathbb{Z}$  into neighborhoods around each multiple of  $s$ . This can be enumerated by denoting  $\mathcal{C}_i$  to be the equivalency class around  $i \cdot s$ , so

$$\mathbb{Z} = \bigcup_{i \in \mathbb{Z}} \mathcal{C}_i.$$

Now, let  $\mathcal{S}_n$  be the set of these equivalency classes.

$$\mathcal{S}_n = \{\dots, \mathcal{C}_{-2}, \mathcal{C}_{-1}, \mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, \dots\}.$$

Now, define the following binary operations,  $\oplus$  and  $\odot$ .

**Definition 3.2.** Let  $\mathcal{C}_i, \mathcal{C}_j \in \mathcal{S}_n$  be equivalency classes as described above. Then, define this operation as  $\mathcal{C}_i \oplus \mathcal{C}_j = \mathcal{C}_{i+j}$ .

**Definition 3.3.** Let  $\mathcal{C}_i, \mathcal{C}_j \in \mathcal{S}_n$  be equivalency classes as described above. Then, define this operation as  $\mathcal{C}_i \odot \mathcal{C}_j = \mathcal{C}_{i \cdot j}$ .

Since both operations return elements of  $\mathcal{S}_n$ , they are both closed. It is simple to show that these satisfy the necessary conditions to make  $(\mathcal{S}_n, \oplus, \odot)$  a ring.

This structure will serve as a stronger model for discussing the encryption scheme. The  $\oplus$  and  $\odot$  operators mimic the interaction of two integers near a multiple of  $s$ .

## 4 Implementation

### 4.1 Special Distribution, $\mathcal{D}_{\gamma, \rho}(p)$

We define  $\mathcal{D}_{\gamma, \rho}(p)$  and analyze it prior to discussing the encryption scheme. We define  $\mathcal{D}_{\gamma, \rho}(p)$ ,

**Definition 4.1.** Let  $s \in \mathbb{Z}$  be odd and positive. Now define the distribution of interest as

$$\mathcal{D}_{\gamma, \rho}(p) = \{\text{choose } q \leftarrow \mathbb{Z} \cap [0, 2^\gamma/s), \quad r \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho), \quad \text{output } x = sq + r\}.$$

Random variables drawn from  $\mathcal{D}_{\gamma, \rho}(p)$  are simply noisy multiples of  $s$  with certain size restrictions.  $r$  is the *noise parameter*, with  $\rho$  dictating the size, in bits of  $r$ . Notice it is evenly distributed over  $sq$ . Since for  $x \leftarrow \mathcal{D}_{\gamma, \rho}(p)$ ,  $x = sq + r$ , if  $\rho = 0$  then  $r = 0$  so  $x \in \mathcal{S}$ . However, with nonzero noise, we see that if  $x = sq + r$ , then  $x \in \mathcal{C}_q \in \mathcal{S}_n$ . So, this distribution can be seen as choosing a random element of  $\mathcal{S}_n$  and then a random element within a subset of that equivalency class.

The noise level determines how far from the nearest multiple of  $s$  an element from  $\mathcal{D}_{\gamma, \rho}(p)$  can be.

## 4.2 Overview of Scheme

First,  $\lambda$ , the security parameter is set. Then, the following parameters are set

$\gamma$  is the bit-length of the integers in the public key,

$\nu$  is the bit-length of the secret key (which is the hidden approximate-gcd of all the public-key integers),

$\rho$  is the bit-length of the noise (i.e., the distance between the public key elements and the nearest multiples of the secret key), and

$\tau$  is the number of integers in the public key.

Then, the `KeyGen`, `Encrypt`, `Decrypt`, and `Evaluate` functions can be described in terms of these, and the input bit,  $m \in \{0, 1\}$ .

### 4.2.1 KeyGen

The first step is to create the public key,  $p$  and the secret key,  $s$ . We define  $s$  to be an odd  $\nu$ -bit integer, so

$$s \leftarrow (2\mathbb{Z} + 1) \cap [2^{\nu-1}, 2^\nu).$$

To create the public key, we start by sampling  $\mathcal{D}_{\gamma, \rho}(p)$  with  $x_i \leftarrow \mathcal{D}_{\gamma, \rho}(p)$  for all  $i = 0, 1, \dots, \tau$ . Relabel to ensure  $x_0$  is the largest. Restart this process until  $x_0$  is odd and  $x_0 \pmod s$  is even. Then,  $p = \langle x_0, \dots, x_\tau \rangle$ .

### 4.2.2 Encrypt

Given a bit  $m \in \{0, 1\}$ , we first choose a random subset  $S \subseteq \{1, 2, \dots, \tau\}$  and random realization  $r \leftarrow (-2^{\rho'}, 2^{\rho'})$ . The encrypted integer,  $c$  is defined

$$c = \left( m + 2r + \sum_{i \in S} x_i \right) \pmod{x_0}.$$

A discussion of why this works is in 4.3.

### 4.2.3 Decrypt

Given an integer  $c$  which has been encrypted by this scheme, it can be unencrypted by setting

$$m = c \pmod s \pmod 2.$$

And  $m$  is the unencrypted bit.

It will become relevant later to state an alternate (equivalent) decryption.

**Lemma 4.2.** *Let  $m \in \{0, 1\}$  be an arbitrary bit. Let  $c = \text{Encrypt}(m)$  under a scheme with secret key  $s$ . Then,*

$$c \pmod s \pmod 2 = c \pmod 2 \wedge q_s(c) \pmod 2.$$

*Proof.* Recall that by construction with KeyGen,  $s$  is odd. Also, by definition,  $c \pmod s = c - \lfloor \frac{c}{s} \rfloor s$ . In decryption of  $c$ , we are only concerned with the parity of  $c \pmod s$ . With  $s$  odd, the parity of the  $\lfloor \cdot \rfloor$  term is unchanged. Thus,

$$c \pmod s \pmod 2 = c - \lfloor \frac{c}{s} \rfloor s \pmod 2 = c - \lfloor \frac{c}{s} \rfloor \pmod 2.$$

The parity of the addition of two integers is the binary  $\wedge$  (XOR) of their least significant bit. Thus,

$$c - \lfloor \frac{c}{s} \rfloor \pmod 2 = c \pmod 2 \wedge \lfloor \frac{c}{s} \rfloor \pmod 2.$$

This completes the result.  $\square$

#### 4.2.4 Evaluate

Performing integer addition and multiplication on the encrypted values and then decrypting returns the equivalent binary addition and multiplication on the original bits.

### 4.3 Proof of Validity

The KeyGen produces a large odd integer,  $s$  and a public key as a tuple of near-multiples of  $s$ ,  $\langle x_0, x_1, \dots, x_\tau \rangle$ .  $x_0$  is the largest of the public key elements, and  $x_0$  is odd and satisfies  $x_0 \pmod s \pmod 2 = 0$ .

The scheme perfectly decrypts if Decrypt is the left inverse of Encrypt. That is, for an arbitrary  $m \in \{0, 1\}$ ,  $\text{Decrypt}(\text{Encrypt}(m)) = m$ .

So, we claim the following,

**Theorem 4.3.** *With sufficiently small noise, the above scheme perfectly decrypts any arbitrary  $m \in \{0, 1\}$ .*

We begin with a technical lemma.

**Lemma 4.4.** *Consider a scheme generated according to the conditions of 4.2 with secret key  $s$  and length of public key  $\tau$  and  $\alpha_1, \dots, \alpha_n$  the indices of the random public key subset from KeyGen. Let  $c$  be the encryption of an arbitrary bit  $m \in \{0, 1\}$ . (So  $c = m + 2r_e + \sum_{i=1}^n x_{\alpha_i}$ ). If the noise parameter,  $\rho$ , satisfies*

$$\rho \leq \frac{1}{2} (\log_2(s-1) - \log_2(2\tau+3))$$

then

$$\left( m + 2r_e + 2 \sum_{i=1}^n r_{\alpha_i} \right) - q_{x_0}(c) r_s(x_0) \in \left( -\frac{s-1}{2}, \frac{s-1}{2} \right].$$

*Proof.* Assume  $\rho \leq \frac{1}{2} (\log_2(s-1) - \log_2(2\tau+3))$  is given. The following are all equivalent,

$$\begin{aligned} \rho &\leq \frac{1}{2} (\log_2(s-1) - \log_2(2\tau+3)) \\ 2\rho + 1 &\leq \log_2 \left( \frac{s-1}{2(2\tau+3)} \right) \\ (2\tau+3)2^{2\rho+1} &< \frac{s-1}{2}. \end{aligned}$$

Decompose  $x_0$  into  $x_0 = c_0s + r_0$ ,  $c_0, r_0 \in \mathbb{Z}$  with  $r_0 \in (-2^\rho, 2^\rho)$ .

$$\begin{aligned}
(2\tau + 3)2^{2\rho+1} &= (4\tau + 6)2^{2\rho} \\
&\geq (4\tau + 5)2^{2\rho} + 2(\tau + 1)2^\rho \\
&\geq m + 2r_e + 2 \sum_{i=1}^n r_{\alpha_i} - (4\tau + 5)2^{2\rho}
\end{aligned}$$

Focusing on just the last term,

$$\begin{aligned}
(4\tau + 5)2^{2\rho} &= (1 + 2\tau + 2 + 2\tau + 1)2^{2\rho} \\
&\geq \left( \frac{1}{c_0s} + \left( \frac{2\tau + 2}{c_0s} \right) 2^\rho + (2\tau + 1) \right) 2^\rho \\
&= \left( \frac{1 + (2)2^\rho + (2\tau)2^\rho}{c_0s} + 2\tau + 1 \right) 2^\rho \\
&\geq \left( \frac{m + (2)2r_e + 2 \sum_{i=1}^n c_i s + r_{\alpha_i}}{c_0s + r_0} + 1 \right) 2^\rho \\
&= \left( \frac{m + (2)2r_e + 2 \sum_{i=1}^n x_{\alpha_i}}{c_0s + r_0} + 1 \right) 2^\rho \\
&= \left\lfloor \frac{m + (2)2r_e + 2 \sum_{i=1}^n x_{\alpha_i}}{c_0s + r_0} \right\rfloor 2^\rho \\
&\geq \left\lfloor \frac{m + (2)2r_e + 2 \sum_{i=1}^n x_{\alpha_i}}{c_0s + r_0} \right\rfloor 2^\rho \\
&= \left\lfloor \frac{m + (2)2r_e + 2 \sum_{i=1}^n x_{\alpha_i}}{x_0} \right\rfloor 2^\rho \\
&= q_{x_0}(c)
\end{aligned}$$

So, putting this into the earlier inequality, if the hypothesis is true,

$$\begin{aligned}
\left| m + 2r_e + 2 \sum_{i=1}^n r_{\alpha_i} - q_{x_0}(c)r_0 \right| &\leq \left| m + 2r_e + 2 \sum_{i=1}^n r_{\alpha_i} \right| + |q_{x_0}(c)r_0| \\
&\leq (2\tau + 3)2^{2\rho+1} \\
&< \frac{s-1}{2}
\end{aligned}$$

so the bound is achieved. □

Now, we are prepared to address the proof of theorem 4.3.

*Proof.* Suppose  $m \in \{0, 1\}$  has been encrypted with Evaluate using  $s$  and  $\langle x_0, x_1, \dots, x_\tau \rangle$  is the private and public keys, respectively. Since each element of the public key is a noisy multiple of  $s$ , we can rewrite the public key elements as

$$\begin{aligned}
x_0 &= c_0s + r_0 \\
x_1 &= c_1s + r_1 \\
&\vdots \\
x_\tau &= c_\tau s + r_\tau.
\end{aligned}$$

Subject to the constraints

- $c_i \in \mathbb{Z}$
- $r_i \in (-2^\rho, 2^\rho) \subseteq \left(-\frac{s-1}{2}, \frac{s-1}{2}\right)$
- $c_0 s + r_0 \geq c_i s + r_i$
- $r_0$  is odd
- $c_0$  is even

for all  $0 \leq i \leq \tau$ .

To encrypt  $m$ , a subset of the public key indices is chosen randomly. Let  $\alpha_1, \alpha_2, \dots, \alpha_n$ ,  $0 < \alpha_i \leq \tau$  for all  $0 < i \leq n$  be the indices of the subset chosen. Finally, more noise is added as  $r_e \leftarrow (-2^{\rho'}, 2^{\rho'})$ .

$$c = m + 2r_e + 2 \sum_{i=0}^n x_{\alpha_i} \pmod{x_0}$$

Then, to decrypt,

$$m' = c \pmod{s} \pmod{2}.$$

$$c = m + 2r_e + 2 \sum_{i=0}^n x_{\alpha_i} \pmod{x_0}$$

Let  $r = r_e + \sum_{i=1}^n r_{\alpha_i}$ . Now, with the definition of  $\pmod{\cdot}$ ,

$$\begin{aligned} c &= m + 2r_e + 2 \sum_{i=0}^n x_{\alpha_i} \pmod{x_0} \\ &= m + 2r + 2 \sum_{i=0}^n c_{\alpha_i} s \pmod{x_0} \\ &= m + 2 \left( r + s \sum_{i=0}^n c_{\alpha_i} \right) \pmod{x_0} \\ &= m + 2 \left( r + s \sum_{i=0}^n c_{\alpha_i} \right) - \left\lfloor \frac{m + 2r + 2 \sum_{i=0}^n c_{\alpha_i} s}{x_0} \right\rfloor (x_0) \end{aligned}$$

Now, applying the first step of decryption to  $c$ , we have

$$\begin{aligned} c \pmod{s} &= m + 2 \left( r + s \sum_{i=0}^n c_{\alpha_i} \right) - \left\lfloor \frac{m + 2r + 2 \sum_{i=0}^n c_{\alpha_i} s}{x_0} \right\rfloor (x_0) \pmod{s} \\ &= m + 2 \left( r + s \sum_{i=0}^n c_{\alpha_i} \right) - \left\lfloor \frac{m + 2r + 2 \sum_{i=0}^n c_{\alpha_i} s}{x_0} \right\rfloor (c_0 s + r_0) \pmod{s} \end{aligned}$$

and since any multiple of  $s$  reduces to 0  $\pmod{s}$ , this simplifies to

$$\begin{aligned} c \pmod{s} &= m + 2r - \left\lfloor \frac{m + 2r + 2 \sum_{i=0}^n c_{\alpha_i} s}{x_0} \right\rfloor r_0 \pmod{s} \\ &= m + 2r - q_{x_0}(c) r_0 \pmod{s}. \end{aligned}$$

It is assumed sufficiently small noise parameter,  $\rho$  and non-trivial public key size  $\tau$ . By lemma 4.4,

$$m + 2r - q_{x_0}(c)r_0 \in \left(-\frac{s-1}{2}, \frac{s-1}{2}\right],$$

so

$$c \pmod{s} = m + 2r - q_{x_0}(c)r_0.$$

Now, since by constraint,  $r_0$  is even, so the final result is simply given

$$c \pmod{s} \pmod{2} = m + 2r - q_{x_0}(c)r_0 \pmod{2} = m.$$

Thus, the decryption is correct. □

## 5 Attacks

The secret key,  $s$ , must be kept private in order to prevent unwanted parties from decrypting data under this scheme. Clearly, if an attacker possesses  $s$  and an encrypted bit,  $c$ , it is trivial to compute  $m = c \pmod{s} \pmod{2}$  to uncover the data. Since the public key elements are noisy multiples of  $s$ , the process for uncovering  $s$  is at most as difficult as solving the Approximate GCD problem.

Let  $\mathcal{A}$  be an attacker with advantage  $\varepsilon$  if for a given ciphertext and public key, it can output the plaintext bit with probability  $\frac{1}{2} + \varepsilon$ . With this, it will be demonstrated how  $\mathcal{A}$  can uncover the secret key.

Before discussing the details of this attack, we give an overview of the Least Significant Bit estimation problem.

### 5.1 Least Significant Bit Guessing

Given an arbitrary  $z \in [0, 2^\gamma)$  with  $|z \pmod{s}| < 2^\rho$  and public key  $p = \langle x_0, x_1, \dots, x_\tau \rangle$ , the output is the least significant bit of  $q_s(z)$ , which is equivalent to  $\text{Decrypt}(z)$ .

The method proposed for estimating this value, is to simply perform the following procedure  $\text{poly}(\lambda)/\varepsilon$  times, and take the majority result.

Choose a random bit  $m$  and perform  $c = \text{Encrypt}(z + m)$ . Then use  $\mathcal{A}$  with  $c$  and  $p$  to predict  $a = \text{Decrypt}(c)$ . Finally, set  $b = a \wedge z \pmod{2} \wedge m$ .

**Theorem 5.1.** *This routine will return the least significant bit of  $q_s(z)$  with probability proportional to  $\varepsilon$ .*

*Proof.* Consider a single iteration of the above-described method.  $\text{Decrypt}(\text{Encrypt}(z + m))$  is equivalent to

$$\text{Decrypt}(z) \wedge m = z \pmod{s} \pmod{2} \wedge m.$$

By lemma 4.2, this becomes

$$z \pmod{s} \pmod{2} \wedge m = (z \pmod{2} \wedge q_s(z) \pmod{2}) \wedge m.$$

So if  $a$  is the correct bit, then  $a = \text{Decrypt}(\text{Encrypt}(z + m))$ , so the output is

$$\begin{aligned} b &= a \wedge z \pmod{2} \wedge m \\ &= (z \pmod{2} \wedge q_s(z) \pmod{2} \wedge m) \wedge z \pmod{2} \wedge m \\ &= (z \pmod{2} \wedge z \pmod{2}) \wedge (m \wedge m) \wedge q_s(z) \pmod{2} \\ &= q_s(z) \pmod{2} \end{aligned}$$



which is the least significant bit of  $q_s(z)$ , as desired.  $\square$

## 5.2 Binary GCD Algorithm

To give motivation for the Approximate GCD algorithm, a brief explanation of the Binary (exact) GCD algorithm is helpful.

The Euclidean algorithm for solving the GCD problem is given as,

Given  $x, y \in \mathbb{Z}$ , use the following procedure.

1. If  $x < y$ , swap  $x, y$
2. iteratively form

$$\begin{aligned} x &= q_0 y + r_0 \\ y &= q_1 r_0 + r_1 \\ r_0 &= q_2 r_1 + r_2 \\ r_1 &= q_3 r_2 + r_3 \\ &\vdots \\ r_{k-2} &= q_k r_{k-1} + r_k \end{aligned}$$

3. Return  $r_{k-1}$ , the gcd of  $x$  and  $y$ .

The Binary GCD algorithm is similar, but uses simpler bit-wise arithmetic operations. It relies on the following identities for arbitrary  $u, v \in \mathbb{Z}$ .

- $\gcd(u, 0) = u$
- $\gcd(0, v) = v$
- $u$  and  $v$  even, then  $\gcd(u, v) = 2 \cdot \gcd(\frac{u}{2}, \frac{v}{2})$
- $u$  odd and  $v$  even, then  $\gcd(u, v) = \gcd(u, \frac{v}{2})$  (same is true swapping  $u, v$ )
- $u$  and  $v$  odd and  $u \geq v$ , then  $\gcd(u, v) = \gcd(\frac{u-v}{2}, v)$

## 5.3 Solving Approximate GCD