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 predetermined 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. □

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 $\lfloor \frac{a}{s} \rceil = \lfloor \frac{b}{s} \rceil$. 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.

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

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*

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

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 ρ 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, λ, the security parameter is set. Then, the following parameters are set

γ is the bit-length of the integers in the public key,

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

ρ 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

τ 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 ν-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 $i = 0, 1, \ldots, \tau$. Relabel to ensure x_0 is the largest. Restart this process until x_0 is odd and $x_0 \pmod s$ is even. Then, $p =< x_0, \ldots, x_\tau >$.

4.2.2 `Encrypt`

Given a bit $m \in \{0, 1\}$, we first choose a random subset $S \subseteq \{1, 2, \ldots, \tau\}$ 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.

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, $< x_0, x_1, \ldots, x_\tau >$. x_0 is the largest of the publik 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 $m \in \{0,1\}$, $\texttt{Decrypt}(\texttt{Encrypt}(m)) = m$.

So, we claim the following,

Theorem 4.2. *With sufficiently small noise, the above scheme perfectly decrypts any arbitrary $m \in \{0,1\}$.*

We begin with a technical lemma.

Lemma 4.3. *Consider a scheme generated according to the conditions of 4.2 with secret key s and length of public key τ and $\alpha_1, \ldots, \alpha_n$ 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, ρ, satisfies*

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

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

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

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

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

Disregarding all but the last term,

$$\begin{aligned}
(4\tau+5)2^{2\rho+1} &= (1+2\tau+2+2\tau+1)2^{2\rho} \\
&\geq \left(\frac{1}{c_0 s} + \left(\frac{2\tau+2}{c_0 s}\right)2^{\rho} + (2\tau+1)\right)2^{\rho} \\
&= \left(\frac{1+(2)2^{\rho}+(2\tau)2^{\rho}}{c_0 s} + 2\tau+1\right)2^{\rho} \\
&\geq \left(\frac{m+(2)2r_e+2\sum_{i=1}^{n} c_i s + r_{\alpha_i}}{c_0 s + r_0} + 1\right)2^{\rho} \\
&= \left(\frac{m+(2)2r_e+2\sum_{i=1}^{n} x_{\alpha_i}}{c_0 s + r_0} + 1\right)2^{\rho} \\
&= \left|\left\lfloor \frac{m+(2)2r_e+2\sum_{i=1}^{n} x_{\alpha_i}}{c_0 s + r_0} \right\rceil 2^{\rho}\right| \\
&\geq \left\lfloor \frac{m+(2)2r_e+2\sum_{i=1}^{n} x_{\alpha_i}}{c_0 s + r_0} \right\rceil 2^{\rho} \\
&= \left\lfloor \frac{m+(2)2r_e+2\sum_{i=1}^{n} x_{\alpha_i}}{x_0} \right\rceil 2^{\rho} \\
&= q_{x_0}(c)
\end{aligned}$$

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 4.2.

Proof. Suppose $m \in \{0,1\}$ has been encrypted with `Evaluate` using s and $< x_0, x_1, \ldots, x_\tau >$ 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

$$\begin{aligned}
x_0 &= c_0 s + r_0 \\
x_1 &= c_1 s + r_1 \\
&\vdots \\
x_\tau &= c_\tau s + r_\tau.
\end{aligned}$$

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 $0 \le i \le \tau$.

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

$$c = m + 2r_e + 2\sum_{i=0}^{n} x_{\alpha_i} \,(\text{mod } x_0)$$

Then, to decrypt,

$$m' = c\,(\text{mod } s)\,(\text{mod } 2)\,.$$

$$c = m + 2r_e + 2\sum_{i=0}^{n} x_{\alpha_i} \,(\text{mod } x_0)$$

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

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

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

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

and since any multiple of s reduces to $0\,(\text{mod } s)$, this simplies to

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

It is assumed sufficiently small noise parameter, ρ and non-trivial public key size τ. By lemma 4.3,

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

so

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

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

5.1 Least Significant Bit Guessing

5.2 Solving Approximate GCD