Scale Invariant Fully Homomorphic Encryption Over the Integers

December 5, 2016

PREPARED IN FULFILLMENT OF THE MATHEMATICS COURSE MAT5160 BY

RANA KHALIL

Abstract

In this report we describe what it means for an encryption scheme to be fully homomorphic. This is illustrated through a simple scale invariant leveled fully homomorphic encryption scheme over the integers, that can be bootstrapped into a pure fully homomorphic encryption scheme. The semantic security of this encryption scheme is then analyzed and reduced to a computationally hard problem known as Approximate-GCD problem.

Contents

1 Introduction

The demand for privacy of digital data has increased dramatically over the last decade. Countless measures have been taken to securely store and access data. Methods such as strong encryption have allowed individuals to store massive amounts of encrypted information on the cloud without fear of an adversary recovering any useful information from that data. However, the problem with encrypted data is that sooner or later you need to decrypt it. Any operations, such as querying the data, have to be processed on the plaintext. Fully Homomorphic Encryption (FHE), a still mostly theoretical but very promising advancement, could completely change that!

The idea of homomorphic encryption was first introduced in 1978 by Rivest, Adleman and Dertouzos in their paper "On Data Banks and Privacy Homomorphisms". The paper introduced the idea of privacy homomorphisms which they defined as "encryption functions which permit encrypted data to be operated on without preliminary decryption of the operands, for many sets of interesting operations" [1]. Following that paper, many encryption schemes were developed that are additively homomorphic, multiplicatively homomorphic and Some What Homomorphic Encryption (SWHE) schemes which only allowed a limited number of additions and multiplications. It was not until 2009 that Craig Gentry implemented the first FHE scheme that was able to perform arbitrarily many additive and multiplicative operations on Ideal lattices [2]. Gentry's contribution not only proved the possibility of implementing a fully homomorphic encryption scheme but also laid a solid groundwork for recent developments in this field, which included FHE over the integers, FHE using Learning with Errors (LWE) and FHE using Hidden Ideal Lattices.

This report is organized as follows. Section 2 cover the basic mathematical definitions and concepts that will be required for the proper understanding of the rest of this report. Section 3 covers public key encryption and FHE. Section 4 goes into depth on one of the FHE schemes, namely, Scale Invariant Fully Homomorphic Encryption over the Integers and provides a proof of correctness and semantic security.

2 Preliminaries

This section covers the basic mathematical definitions and concepts that will be required for the proper understanding of the rest of this report.

2.1 Euclidean Inner Product

Definition 2.1.1 (Inner Product)**.** *Suppose that $\vec{u}$ and $\vec{v}$ are vectors in $\mathbb{R}^n$ such that $\vec{u} = (u_1, u_2, ..., u_n)$ and $\vec{v} = (v_1, v_2, ..., v_n)$. The euclidean inner product of $\vec{u}$ and $\vec{v}$ is defined by*

$$\langle \vec{u}, \vec{v} \rangle = \vec{u} \cdot \vec{v} = u_1 v_1 + u_2 v_2 + ... + u_n v_n.$$

Proposition 2.1.2. *Suppose that $\vec{u}$, $\vec{v}$, $\vec{w} \in \mathbb{R}^n$ and $c \in \mathbb{R}$. Then*

1. $\langle \vec{u}, \vec{v} \rangle = \langle \vec{v}, \vec{u} \rangle$
2. $\langle \vec{u}, \vec{v} + \vec{w} \rangle = \langle \vec{u}, \vec{v} \rangle + \langle \vec{u}, \vec{w} \rangle$
3. $c\langle \vec{u}, \vec{v} \rangle = \langle c\vec{u}, \vec{v} \rangle$

2.2 Integer Functions

Definition 2.2.1 (Least Integer Function)**.** *Suppose that $x \in \mathbb{R}$. The least integer function $\lfloor x \rfloor$ is defined as the largest integer not greater than x.*

Definition 2.2.2 (Nearest Integer Function)**.** *Suppose that $x \in \mathbb{R}$. The nearest integer function $\lfloor x \rceil$ is defined as*

$$\lfloor x \rceil = \lfloor x + 1/2 \rfloor.$$

Proposition 2.2.3. *Let $x \in \mathbb{R}$. Then $2\lfloor x/2 \rceil = x + v$ where $|v| \leq 1$.*

Proof. Note that $|v| \leq 1 \iff -1 \leq v \leq 1$. Therefore, it suffices to show that $2\lfloor x/2 \rceil - x \in [-1, 1]$.
Suppose that $x \in \mathbb{R}$ such that $x = n + r$ when $n \in \mathbb{Z}$ and $0 \leq r < 1$. We have two cases to consider.

Case #1 (n is even): Since n is even, $\exists k_1 \in \mathbb{Z}$ such that $n = 2k_1$. Then,

$$\begin{aligned} 2\lfloor x/2 \rceil - x &= 2\lfloor (n + r)/2 \rceil - x \\ &= 2\lfloor (2k_1 + r)/2 \rceil - 2k_1 - r \\ &= 2\lfloor k_1 + r/2 \rceil - 2k_1 - r \\ &= 2k_1 - 2k_1 - r \qquad \textit{(since } 0 \leq r/2 < 1/2\textit{)} \\ &= -r \in (-1, 0] \subseteq [-1, 1]. \end{aligned}$$

Case #2 (n is odd): Since n is odd, $\exists k_2 \in \mathbb{Z}$ such that $n = 2k_2 + 1$. Then,

$$\begin{aligned} 2\lfloor x/2 \rceil - x &= 2\lfloor (n+r)/2 \rceil - x \\ &= 2\lfloor (2k_2 + 1 + r)/2 \rceil - 2k_2 - 1 - r \\ &= 2\lfloor k_2 + 1/2 + r/2 \rceil - 2k_2 - 1 - r \\ &= 2(k_2 + 1) - 2k_2 - 1 - r \\ &= 1 - r \in (0, 1] \subseteq [-1, 1]. \end{aligned}$$

Therefore, $2\lfloor x/2 \rceil = x + v$ *where* $|v| \leq 1$. ■

Proposition 2.2.4. *Let* $x \in \mathbb{R}$. *Then* $\lfloor x \rceil = x + \epsilon$ *when* $|\epsilon| \leq 1/2$.

Proof. Note that $|\epsilon| \leq 1/2 \iff -1/2 \leq \epsilon \leq 1/2$. *Therefore, it suffices to show that* $\lfloor x \rceil - x \in [-1/2, 1/2]$.
Suppose that $x \in \mathbb{R}$ *such that* $x = n + r$ *where* $n \in \mathbb{Z}$ *and* $0 \leq r < 1$. *We have two cases to consider.*

Case #1 ($0 \leq r < 1/2$*):*
$\lfloor x \rceil - x = \lfloor n + r \rceil - n - r = n - n - r = -r \in (-1/2, 0] \subseteq [-1/2, 1/2]$.

Case #2 ($1/2 \leq r < 1$*):*
$\lfloor x \rceil - x = \lfloor n + r \rceil - n - r = n + 1 - n - r = 1 - r \in (0, 1/2] \subseteq [-1/2, 1/2]$.

Therefore, $\lfloor x \rceil = x + \epsilon$ *where* $|\epsilon| \leq 1/2$. ■

2.3 Computational Indistinguishability

We denote the class of probabilistic polynomial time algorithms by PPT.

Definition 2.3.1 (Probabilistic Polynomial Time Algorithm)**.** *A probabilistic polynomial time algorithm is an algorithm that is randomized and runs in polynomial time.* [12]

Definition 2.3.2 (Probability Ensemble)**.** *Let* I *be a countable index set. An ensemble indexed by* I *is a sequence of random variables indexed by* I. *Namely, any* $X = \{X_i\}_{i \in I}$, *where each* X_i *is a random variable, is an ensemble indexed by* I. [9]

Definition 2.3.3 (Computational Indistinguishability)**.** *Two ensembles,* $X \stackrel{\text{def}}{=} \{X_n\}_{n \in \mathbb{N}}$ *and* $Y \stackrel{\text{def}}{=} \{Y_n\}_{n \in \mathbb{N}}$ *are computationally indistinguishable if for all non-uniform PPT algorithms* D *there exists a negligible function* $\epsilon(n)$ *such that* $\forall n \in \mathbb{N}$

$$|Pr[D(X_n) = 1] - Pr[D(Y_n) = 1]| < \epsilon(n)$$

In other words, two (ensembles of) probability distributions are computationally indistinguishable if no efficient distinguisher D *can tell them apart better than with a negligible advantage over random guessing.* [10]

Lemma 2.3.4 (The Hybrid Lemma)**.** *Let $X^1, X^2, ..., X^m$ be a sequence of probability distributions. Assume that there exists $D \in PPT$ which distinguishes X^1 and X^m with probability ϵ. Then there exists some $i \in [1, ..., m-1]$ such that D distinguishes X^i and X^{i+1} with probability ϵ/m.* [10]

Proof. Let $D \in PPT$ such that D is capable of distinguishing X^1 and X^m with probability ϵ, i.e,

$$|Pr[D(X^1) = 1] - Pr[D(X^m) = 1]| > \epsilon$$

Let $w_i = Pr[D(X^i) = 1]$. Then, by assumption $|w_1 - w_m| > \epsilon$. Therefore,

$$\begin{aligned} |w_1 - w_m| &= |w_1 - w_2 + w_2 - w_3 + ... + w_{m-1} - w_m| \\ &\leq |w_1 - w_2| + |w_2 - w_3| + ... + |w_{m-1} - w_m| \end{aligned}$$

using the $\triangle$-inequality. Since

$$|w_1 - w_m| > \epsilon \implies |w_1 - w_2| + |w_2 - w_3| + ... + |w_{m-1} - w_m| > \epsilon$$

Therefore, there exists i such that $|w_i - w_{i+1}| > \epsilon/m$, i.e. there exists i such that D distinguishes X^i and X^{i+1} with probability ϵ/m (otherwise the sum would be less than ϵ). ■

2.4 Leftover Hash Lemma

Definition 2.4.1 (Pairwise Independent)**.** *Let H represent a family of hash functions from X to Y where X and Y are both finite sets. If for every distinct $x, x' \in X$ we have*

$$Pr_{h \leftarrow H}[h(x) = h(x')] = 1/|Y|$$

then H is said to be pairwise independent. [7]

Definition 2.4.2 (Statistical Distance)**.** *Let D_1 and D_2 be two distributions over a finite domain X. Then,*

$$\triangle(D_1, D_2) = 1/2 \sum_{x \in X} |D_1(x) - D_2(x)|$$

where $\triangle(D_1, D_2)$ represents the statistical distance. [7]

Definition 2.4.3 (ϵ-uniform)**.** *If the statistical distance of a distribution D from the uniform distribution is at most ϵ, the distribution is said to be ϵ-uniform.*[7]

Lemma 2.4.4 (Leftover Hash Lemma)**.** *Let H be a family of pairwise independent hash functions from X to Y. Suppose that $h \leftarrow H$ and $x \leftarrow X$ are chosen uniformly and independently. Then, $(h, h(x))$ is $1/2\sqrt{|X|/|Y|}$-uniform over $H \times X$.* [7]

Lemma 2.4.5. *Set $x_1, ..., x_m \leftarrow \mathbb{Z}_M$ uniformly and independently, set $s_1, .., s_m \leftarrow \{0,1\}$, and set $y = \sum_{i=1}^{m} s_i \cdot x_i \mod M$. Then $(x_1, ..., x_m, y)$ is $1/2\sqrt{M/2^m}$-uniform over $\mathbb{Z}_M^{m+1}$.* [7]

3 Encryption

We first recall the basic definition of a public key encryption scheme and what it means for an encryption scheme to be correct and semantically secure. Then we explain homomorphic encryption through a "real world" problem to give the reader more intuition on how it works and provide the corresponding mathematical definitions as we go along.

3.1 Public Key Encryption

Encryption schemes are usually composed of a three tuple of PPT algorithms, namely, key generation, encryption and decryption denoted by (KeyGen, Encrypt, Decrypt).

Definition 3.1.1. *Let M represent the message space, and let ℓ be the security parameter. A public key encryption scheme ε on M is a three tuple $(KeyGen, Encrypt, Decrypt) \in PPT$ satisfying the following functionalities:*

- *$KeyGen_\varepsilon(1^\ell)$: The key generation algorithm takes as input the security parameter 1^ℓ and outputs a public key, secret key pair (pk, sk).*
- *$Encrypt_\varepsilon(pk, m)$: The encryption algorithm takes as input the public key pk and a message $m \in M$ and outputs the corresponding ciphertext $c \in C$ where C is the ciphertext space.*
- *$Decrypt_\varepsilon(sk, c)$: The decryption algorithm takes as input the ciphertext $c \in C$ and secret key sk and outputs the corresponding message $m \in M$.*

A public key encryption scheme must satisfy two requirements:

- **Correctness**: $\forall m \in M$, the $Pr[(pk, sk) \leftarrow KeyGen(1^\ell) : Dec(sk, Enc(pk, m)) = m] = 1$. In other words, a scheme is said to be correct if the decryption of an encrypted message under the corresponding keys returns the original message.
- **Semantic Security**: We define the notion of semantic security in terms of a game between a challenger C and an Adversary A:
  Step 1: The challenger generates a random key pair $(pk, sk) \leftarrow KeyGen_\epsilon(1^\ell)$.
  Step 2: The adversary A sends m_0, m_1 to the challenger.
  Step 3: The challenger generates a random bit, $b \leftarrow \{0, 1\}$, and generates the ciphertext $c \leftarrow Enc(pk, m_b)$.
  Step 4: The adversary outputs a $b' \in \{0, 1\}$.
  The advantage of the adversary A is defined as follows:

$$Adv_A[\varepsilon] \stackrel{\text{def}}{=} |Pr[b' = b] - 1/2|$$

  The encryption scheme ε is said to be semantically secure if for all probabilistic polynomial time adversaries A, the advantage $Adv_A[\varepsilon]$ is negligible.

3.2 Homomorphic Encryption

We will use Gentry's analogy [5] to illustrate how Fully Homomorphic Encryption works.

3.2.1 Alice's Jewelry Store

Alice owns an expensive jewelry store where she produces diamonds from raw material. Alice was doing great with her jewelery business until one day one of her employees took off with a quarter of her precious raw material. After intensive police work the culprit was brought to justice and the material was all restored to Alice; however this incident left Alice with trust issues when it comes to her employees. This got her thinking - is there a way of giving her employees the ability to process the raw material into diamonds without having access to the material itself?

After many sleepless nights, Alice came up with a plausible plan. She decides to use a transparent impenetrable glove box that has a lock on it. Alice keeps the key on her at all times. She uses the key to open the box whenever she needs to insert the raw material and locks the box after she is done. This way the workers can use the gloves to assemble the raw material into diamonds without having direct access to the raw material. Once the final product is produced, Alice opens the box with her key and takes out the assembled diamonds.

In the above analogy, the raw material represents our plaintext $m_1, ..., m_t$. The transparent impenetrable glove box with the raw material inside of it represents the encryption of the plaintext. Opening (decrypting) the box can only be done using Alice's key. The gloves represent the homomorphism of the encryption scheme by allowing workers to perform manipulations on the raw data when it's in encrypted form (inside the box).

Of course, the above given analogy, as you might have noticed, is flawed. Nevertheless, it does give the reader an intuition for how homomorphic encryption works.

3.2.2 Homomorphic Encryption

A homomorphic encryption scheme can be either symmetric or asymmetric. For the purpose of this report, we'll focus only on public key (asymmetric) homomorphic encryption.

A FHE scheme is made up of the conventional algorithms required in a public key encryption scheme (Refer to Section 3.1 for review): key generation, encryption and decryption. In addition to these algorithms, a homomorphic encryption scheme requires an *Evaluate* algorithm. This algorithm takes as input the public key pk, a circuit C (combinations of AND and XOR gates) and a tuple of ciphertext $(c_1, ..., c_t)$ which are used as the inputs to C. The output of the *Evaluate* algorithm is a ciphertext c such that the decryption of c under the secret key outputs $m = (m_1, ..., m_t)$ where c_i is the corresponding ciphertext of the plaintext m_i.

As mentioned before, a FHE scheme allows arbitrarily many computations on the ciphertext. A relation of that is called a leveled fully homomorphic encryption scheme, where circuits can be evaluated only up to a fixed depth.

3.2.3 How to Construct a FHE Scheme

Going back to Alice's jewellery store, Alice orders a bulk of glove boxes from Acme Glovebox Company to test out her idea. Unfortunately for Alice, all the glove boxes turn out to be defective! The glove boxes can only be used for a few minutes before the gloves stiffen and become unusable. Alice files a complaint and is promised a replacement of her order to arrive in a few weeks. While waiting for her order, Alice decides to make use of what she has. Alice realizes that the glove boxes have a one way opening which can be used by her workers to insert material into the box. Being the smart person she is, it only takes her a few minutes to figure out how she can use that property to come up with a possible solution to her problem.

Alice gives her worker a couple of glove boxes. The first glove box, labelled box #1, contains the raw material; the second glove box, labelled box #2, contains the key to box #1; the third glove box, labelled box #3, contains the key to box #2 and so on. The worker first uses the gloves of box #1 to work on the raw material. When the gloves of box #1 stiffen and become unusable, he takes box #1 and inserts it into box #2. Then he uses the gloves of box #2 to use the the key inside the box to open box #1, extract the somewhat assembled raw material and continue to work on assembling it. When the gloves of box #2 stiffen, the worker moves on to box #3 in a similar fashion. When the worker finishes assembling the raw material (in box #n), he hands box #n to Alice and Alice uses her secret key to open it and retrieve the diamond. Alice observes that with the right amount of glove boxes and the ability to unlock the box and do a bit of assembly work within the useful time frame of the box, Alice can assemble any precious diamond she wants!

The first step of Gentry's scheme [2] is to describe a Somewhat Homomorphic Encryption (SWHE) scheme which supports a limited number of additions and multiplications. Each ciphertext carries a noise component which is increased with every homomorphic operation. When the noise becomes too large, the ciphertexts no longer decrypt properly. In the above analogy our SWHE scheme is represented by the defective boxes, where Alice can assemble the raw material only for a limited amount of time before the gloves stiffen.

The second step is to convert the SWHE scheme to a FHE scheme. This is exactly what Alice was trying to do when she stored the raw material in box #1 and for every other box #$(i+1)$, she stored the key to open the previous box #i. In order for this to work, the SWHE scheme must be able to handle the decryption function. If the SWHE scheme is capable of handling its own decryption function, we say that the scheme is bootstrappable. If the encryption scheme is bootstrappable then we can convert it to a fully homomorphic encryption scheme.

This report does not cover how to convert a bootstrappable scheme into a

FHE scheme. For more information on this topic refer to [5].

4 Scale Invariant DGHV Scheme

Fully Homomorphic Encryption over the integers was first introduced by Dijk, Gentry, Halevi and Vaikuntanathan (DGHV) [6]. As the name suggests, addition and multiplication operations are done over the integers with the semantic security of this scheme depending on the hardness of the Approximate GCD problem. This was the first scheme to implement FHE using simple mathematical operations. Following this scheme, several variant FHE implementations over the integers have been presented.

In this section we'll describe one of the variant schemes introduced by Coron, Lepoint and Tibouchi known as Scale Invariant Fully Homomorphic Encryption over the Integers. This scheme provides the key feature that the single secret modulus grows linearly (as opposed to exponentially) with the multiplicative depth of the circuit when homomorphically evaluated [7]. Section 4.1 describes the parameters used in this scheme and lists the constraints necessary on these parameters. Section 4.2 describes the scale invariant FHE scheme. Section 4.3 provides a proof of correctness, and lastly, Section 4.4 reduces the security of this scheme to solving the Approximate GCD problem and thereby presents a proof of semantic security.

4.1 Constraints on the Parameters

The parameters used in this scheme are as follows.

Table 1: List of Parameters [3]

Symbol	**Quantity**
λ	Security parameter
γ	Bit-length of integers in the public key
η	Bit-length of private key
ρ	Bit-length of the noise
τ	Number of integers in the public key
ρ'	Secondary noise parameter used for encryption
L	Multiplicative depth of the circuit
Θ	Asymptotic tight bound
Ω	Asymptotic lower bound
ω	Weaker Asymptotic lower bound

The parameters of this scheme must meet the following constraints [7]:

1. $\rho = \Omega(\lambda)$, protects against brute force attacks on the noise,
2. $\eta \geq \rho + \mathcal{O}(L \cdot log\lambda)$,
3. $\gamma = \omega(\eta^2 \cdot log\lambda)$, thwarts lattice-based attacks,

4. $\Theta^2 = \gamma \cdot \omega(log\lambda)$, avoids lattice attacks on the subset sum,

5. $\tau \geq \gamma + 2\lambda$, allows use of Leftover Hash Lemma.

4.2 Public-Key Leveled Fully Homomorphic Encryption

The scale invariant FHE scheme was introduced by [7] and described as follows.

For a specific η-bit odd integer p and an integer $q_0 \in [0, 2^\gamma/p^2)$, we define the set:

$$D^\rho_{p,q_0} = \{q \cdot p^2 + r : q \in \mathbb{Z} \cap [0, q_0), r \in \mathbb{Z} \cap (-2^\rho, 2^\rho)\}$$

- $KeyGen(1^\lambda)$: Generate an odd η-bit integer p and a γ-bit integer $x_0 = q_0 \cdot p^2 + r_0$ with $r_0 \leftarrow (-2^\rho, 2^\rho) \cap \mathbb{Z}$ and $q_0 \leftarrow [0, 2^\gamma/p^2) \cap \mathbb{Z}$. Let $x_i \leftarrow D^\rho_{p,q_0}$ for $1 \leq i \leq \tau$. Let also $y' \leftarrow D^\rho_{p,q_0}$ and $y = y' + (p-1)/2$. Let $\vec{z}$ be a vector of θ numbers with $\kappa = 2\gamma + 2$ bits of precision after the binary point, and let $\vec{s}$ be a vector of θ bits such that

  $$2^\eta/p^2 = < \vec{s}, \vec{z} > + \epsilon \mod 2^\eta$$

  with $|\epsilon| \leq 2^{-\kappa}$. Now define

  $$\vec{\sigma} = \vec{q} \cdot p^2 + \vec{r} + \lfloor PowersofTwo_\eta(\vec{s}) \cdot p/2^{\eta+1} \rceil,$$

  where the components of $\vec{q}$ (resp. $\vec{r}$) are randomly chosen from $\mathbb{Z} \cap [0, q_0)$ (resp. $\mathbb{Z} \cap (-2^\rho, 2^\rho)$).
  The secret key is $sk = \{p\}$ and the public key is $pk = \{x_0, x_1, ..., x_\tau, y, \vec{\sigma}, \vec{z}\}$.

- $Encrypt(pk, m \in \{0,1\})$: Choose a random subset $S \subset \{1, ..., \tau\}$ and output

  $$c \leftarrow \left[m \cdot y + \sum_{i \in S} x_i \right]_{x_0}.$$

  ($[z]_{x_0}$ represents $z \mod x_0$).

- $Add(pk, c_1, c_2)$: Output $c \leftarrow c_1 + c_2 \mod x_0$.

- $Convert(pk, c)$: Output $c' \leftarrow 2 \cdot \langle \vec{\sigma}, BitDecomp_\eta(\vec{c}) \rangle$ where $\vec{c} = (\lfloor c \cdot z_i \rceil \mod 2^\eta)_{1 \leq i \leq \theta}$.

- $Mult(pk, c_1, c_2)$: Output $c' \leftarrow Convert(pk, 2 \cdot c_1 \cdot c_2) \mod x_0$.

- $Decrypt(sk, c)$: Output $m \leftarrow ((2c) \mod p) \mod 2$.

Since the noise grows linearly with the number of levels, the above described scheme is a leveled fully homomorphic encryption scheme. In order to obtain a pure fully homomorphic encryption scheme, Gentry's bootstrapping technique can be applied.

4.3 Proof of Correctness

Let $Encrypt(pk, m \in \{0,1\})$ be defined as in Section 4.2. Then, for a random subset $S \subset \{1, ..., \tau\}$, the ciphertext is encrypted as follows.

$$c = \left[m \cdot y + \sum_{i \in S} x_i \right]_{x_0}$$

$$\begin{aligned}
\implies c &= \left[m \cdot (y' + (p-1)/2) + \left(q_1 \cdot p^2 + r_1 \right) \right]_{x_0} \text{ for some } q_1, r_1 \in \mathbb{Z} \\
&= \left[m \cdot \left(q_2 \cdot p^2 + r_2 + (p-1)/2 \right) + \left(q_1 \cdot p^2 + r_1 \right) \right]_{x_0} \text{ for some } q_1 \in \mathbb{Z} \cap [0, q_0), \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad r_1 \in \mathbb{Z} \cap (-2^\rho, 2^\rho) \\
&= \left[m \cdot q_2 \cdot p^2 + m \cdot r_2 + m \cdot (p-1)/2 + (q_1 \cdot p^2 + r_1) \right]_{x_0} \\
&= \left[(m \cdot r_2) + (m + (2r_1/(p-1))) \cdot (p-1)/2 + (m \cdot q_2 + q_1) \cdot p^2 \right]_{x_0} \\
&= \left[r + (m + 2r^*) \cdot (p-1)/2 + q \cdot p^2 \right]_{x_0}
\end{aligned}$$

where r and r^* are the two noises of the ciphertext. Coron, et al., call a ciphertext of this following form a Type I ciphertext.
Note that homomorphic addition and multiplication is done modulo x_0; however, we remove the modulus for simplicity without fear of it affecting our calculations.

4.3.1 Homomorphic Addition

Let c_1 and c_2 be two Type-I ciphertexts with noise length (ρ, ρ^*). Then,

$$\begin{aligned}
c_1 &= r_1 + (m_1 + 2r_1^*) \cdot (p-1)/2 + q_1 \cdot p^2 \\
c_2 &= r_2 + (m_2 + 2r_2^*) \cdot (p-1)/2 + q2 \cdot p^2
\end{aligned}$$

with noises r_1 and r_2 of respective bit length ρ, and r_1^* and r_2^* of respective bit length ρ^*.
Homomorphic addition is done as follows.

$$c_1 + c_2 = r_3 + (m_1 + m_2 + 2r_3^*) \cdot (p-1)/2 + q_3 \cdot p^2$$

for integers $r_3 = r_1 + r_2$, $r_3^* = r_1^* + r_2^*$ and $q_3 = q_1 + q_2$, with $log_2|r_3| \leq \rho + 1$ and $log_2|r_3^*| \leq \rho^* + 1$.

4.3.2 Homomorphic Multiplication

Let c_1 and c_2 be two Type-I ciphertexts with noise length (ρ, ρ^*). Then,

$$c_1 = r_1 + (m_1 + 2r_1^*) \cdot (p-1)/2 + q_1 \cdot p^2$$

$$c_2 = r_2 + (m_2 + 2r_2^*) \cdot (p-1)/2 + q_2 \cdot p^2$$

with noises r_1 and r_2 of respective bit length ρ and, r_1^* and r_2^* of respective bit length ρ^*.

Homomorphic multiplication is done as follows.

$$\begin{aligned}
c_3 &= 2 \cdot c_1 \cdot c_2 \\
&= 2\left(r_1 + (m_1 + 2r_1^*) \cdot (p-1)/2 + q_1 \cdot p^2\right) \cdot \left(r_2 + (m_2 + 2r_2^*) \cdot (p-1)/2 + q_2 \cdot p^2\right) \\
&= 2r_1 \cdot r_2 + (r_1(m_2 + 2r_2^*) + r_2(m_1 + 2r_1^*)) \cdot (p-1) \\
&\quad + (m_1 + 2r_1^*) \cdot (m_2 + 2r_2^*) \cdot (p-1)^2/2 \\
&\quad + (2r_1q_2 + q_2(m_1 + 2r_1^*) \cdot (p-1) + 2r_2q_1 + q_1(m_2 + 2r_2^*) \cdot (p-1) + 2q_1q_2p^2) \cdot p^2 \\
&= (2r_1r_2 + (r_1(m_2 + 2r_2^*) + r_2(m_1 + 2r_1^*))(p-1)) \\
&\quad + (m_1 + 2r_1^*)(m_2 + 2r_2^*) \cdot (p-1)^2/2 + q_3'p^2 \\
&= r_3' + (m_1 + 2r_1^*)(m_2 + 2r_2^*) \cdot (p-1)^2/2 + q_3'p^2
\end{aligned}$$

for some $r_3', q_3' \in \mathbb{Z}$ with $log_2|r_3'| \leq \eta + \rho + \rho^* + 3$. Then we have

$$\begin{aligned}
c_3 &= r_3' + (m_1m_2 + (2m_1r_2^* + 2m_2r_1^* + 4r_1^*r_2^*)) \cdot ((p^2-1)/2 - (p-1)) + q_3'p^2 \\
&= r_3' + m_1m_2 \cdot (p^2-1)/2 - m_1m_2 \cdot (p-1) + (2m_1r_2^* + 2m_2r_1^* + 4r_1^*r_2^*) \cdot (p^2-1)/2 \\
&\quad - (2m_1r_2^* + 2m_2r_1^* + 4r_1^*r_2^*) \cdot (p-1) + q_3'p^2 \\
&= r_3' + m_1m_2 \cdot (p^2-1)/2 - m_1m_2(p-1) \\
&\quad + ((m_1r_2^* + m_2r_1^* + 2r_1^*r_2^*) \cdot p^2 - (m_1r_2^* + m_2r_1^* + 2r_1^*r_2^*)) \\
&\quad - (2m_1r_2^* + 2m_2r_1^* + 4r_1^*r_2^*) \cdot (p-1) + q_3'p^2 \\
&= ((r_3' - m_1m_2 \cdot (p-1) - (m_1r_2^* + m_2r_1^* + 2r_1^*r_2^*)) \\
&\quad - (2m_1r_2^* + 2m_2r_1^* + 4r_1^*r_2^*) \cdot (p-1)) \\
&\quad + m_1m_2 \cdot (p^2-1)/2 + ((m_1r_2^* + m_2r_1^* + 2r_1^*r_2^*) + q_3') \cdot p^2 \\
&= r_3 + m_3 \cdot (p^2-1)/2 + q_3 \cdot^2
\end{aligned}$$

For some integers $r_3, q_3, m_3 = m_1m_2$ where $log_2|r_3| \leq \eta + \rho + \rho^* + 4$ assuming $\rho^* < \rho$. Coron, et al., call a ciphertext of this form a Type II ciphertext. In order to convert a Type II ciphertext back to a Type I ciphertext we use the following Lemma.

Lemma 4.3.1. *Let ρ' be such that $\rho' \geq \eta + \rho + log_2(\eta\theta)$. The procedure Convert converts a Type-II ciphertext with noise size ρ' into a Type-I ciphertext with noise $(\rho' - \eta + 5, log_2\theta)$.* [7]

Proof. The proof is presented in the next section by evaluating the *Convert* procedure. ■

4.3.3 Convert Procedure

For this section of the report, assume that initially the two ciphertexts c_1 and c_2 are Type-I ciphertexts with noise $(\rho_1, log_2\theta)$.

We start with a Type-II ciphertext

$$r + m \cdot (p^2 - 1)/2 + q \cdot p^2$$

with $|r| \leq 2^{\rho'}$ where the noise ρ' is at most $\rho' = \eta + \rho_1 + log_2\theta + 4$ (follows from the result of the previous section). Let $\vec{s}' = PowersofTwo_\eta(\vec{s})$ and $\vec{c}' = BitDecomp_\eta(\vec{c})$. Using

$$\vec{\sigma} = p^2 \cdot \vec{q} + \vec{r} + \lfloor \vec{s}' \cdot p/2^{\eta+1} \rceil$$

we have:

$$\begin{aligned} c' &= 2\langle \vec{\sigma}, \vec{c}' \rangle \\ &= 2\left\langle p^2 \cdot \vec{q} + \vec{r} + \lfloor \vec{s}' \cdot p/2^{\eta+1} \rceil, \vec{c}' \right\rangle \\ &= 2p^2 \cdot \langle \vec{q}, \vec{c}' \rangle + 2 < \vec{r}, \vec{c}' > + 2\left\langle \lfloor \vec{s}' \cdot p/2^{\eta+1} \rceil, \vec{c}' \right\rangle \qquad (*) \end{aligned}$$

Using Proposition 2.2.3 in Section 2.3 we evaluate,

$$\begin{aligned} 2\left\langle \lfloor \vec{s}' \cdot p/2^{\eta+1} \rceil, \vec{c}' \right\rangle &= \langle \vec{s}' \cdot p/2^\eta + \vec{v}, \vec{c}' \rangle \\ &= \langle \vec{s}' \cdot p/2^\eta, \vec{c}' \rangle + \langle \vec{v}, \vec{c}' \rangle \\ &= p/2^\eta \cdot < \vec{s}', \vec{c}' > + v_2 \qquad (**) \end{aligned}$$

where $|v_2| \leq \theta \cdot \eta$. Note that,

$$\begin{aligned} < \vec{s}', \vec{c}' > &= \langle PowersofTwo_\eta(\vec{s}), BitDecomp_\eta(\vec{c}) \rangle \\ &= \left\langle \left(\vec{s}, 2 \cdot \vec{s}, ..., 2^{\eta-1} \cdot \vec{s}\right), (\vec{c}_0, ..., \vec{c}_{\eta-1}) \right\rangle \end{aligned}$$

where $\vec{c} \mod 2^\eta = \sum_{i=0}^{\eta-1} \vec{c}_i \cdot 2^i$. Therefore, $< \vec{s}', \vec{c}' > = < \vec{s}, \vec{c} > \mod 2^\eta = < \vec{s}, \vec{c} > + q_2 \cdot 2^\eta$ for some $q_2 \in \mathbb{Z}$. Furthermore,

$$< \vec{s}, \vec{c} > = \langle (s_1, s_2, ..., s_\theta), (\lfloor c \cdot z_i \rceil \mod 2^\eta)_{1 \leq i \leq \theta} \rangle = \sum_{i=1}^{\theta} (s_i \cdot \lfloor c \cdot z_i \rceil) + \triangle \cdot 2^\eta$$

for some $\triangle \in \mathbb{Z}$. Using Proposition 2.2.4, we evaluate

$$\begin{aligned} < \vec{s}, \vec{c} > &= \sum_{i=1}^{\theta} s_i \cdot (c \cdot z_i + \pi_i) + \triangle \cdot 2^\eta \\ &= \sum_{i=1}^{\theta} (s_i \cdot c \cdot z_i) + \sum_{i=1}^{\theta} (s_i \cdot \pi_i) + \triangle \cdot 2^\eta \\ &= \sum_{i=1}^{\theta} (s_i \cdot c \cdot z_i) + \delta_1 + \triangle \cdot 2^\eta \\ &= c < \vec{s}, \vec{z} > + \delta_1 + \triangle \cdot 2^\eta \end{aligned}$$

where $|\delta_1| \leq \theta/2$. Since $2^\eta/p^2 =< \vec{s}, \vec{z} > +\epsilon \mod 2^\eta$, we have $< \vec{s}, \vec{z} >= 2^\eta/p^2 - \epsilon - \mu \cdot 2^\eta$ for some $\mu \in \mathbb{Z}$. Also, using $c = q \cdot p^2 + m \cdot (p^2 - 1)/2 + r$ as our Type II ciphertext we have:

$$\begin{aligned}
< \vec{s}, \vec{c} > &= c \cdot (2^\eta/p^2 - \epsilon - \mu \cdot 2^\eta) + \delta_1 + \triangle \cdot 2^\eta \\
&= c \cdot (2^\eta/p^2) - c \cdot \epsilon - c \cdot \mu \cdot 2^\eta + \delta_1 + \triangle \cdot 2^\eta \\
&= c \cdot (2^\eta/p^2) - c \cdot \epsilon + \delta_1 + (\triangle - c \cdot \mu) \cdot 2^\eta \\
&= (q \cdot p^2 + m \cdot (p^2 - 1)/2 + r) \cdot (2^\eta/p^2) - c \cdot \epsilon + \delta_1 + (\triangle - c \cdot \mu) \cdot 2^\eta \\
&= q \cdot 2^\eta + m \cdot 2^{\eta-1} - m \cdot (2^\eta/2p^2) + r \cdot (2^\eta/p^2) - c \cdot \epsilon + \delta_1 + (\triangle - c \cdot \mu) \cdot 2^\eta \\
&= (q + (\triangle - c \cdot \mu)) \cdot 2^\eta + m \cdot 2^{\eta-1} + (r \cdot (2^\eta/p^2) - c \cdot \epsilon + \delta_1 - m \cdot (2^\eta/2p^2)) \\
&= q_1 \cdot 2^\eta + m \cdot 2^{\eta-1} + r^*
\end{aligned}$$

for some $r^* \in Z$, with $|r^*| \leq 2^{\rho' - \eta + 3}$ (See Appendix A for bound calculation). We input this result into $(**)$ to get:

$$\begin{aligned}
2\left\langle \lfloor \vec{s}' \cdot p/2^{\eta+1} \rceil, \vec{c}' \right\rangle &= \frac{p}{2^\eta} \cdot ((q_1 \cdot 2^\eta + m \cdot 2^{\eta-1} + r^*) + q_2 \cdot 2^\eta) + v_2 \\
&= \frac{p}{2^\eta} \cdot ((q_1 + q_2) \cdot 2^\eta + m \cdot 2^{\eta-1} + r^*) + v_2 \\
&= (q_1 + q_2) \cdot p + m \cdot \frac{p}{2} + \frac{p}{2^\eta} \cdot r^* + v_2 \\
&= q_3 p + m \cdot \frac{p}{2} + \frac{p}{2^\eta} \cdot r^* + v_2
\end{aligned}$$

where $|q_3| \leq \theta$. This can be written as,

$$\begin{aligned}
2\left\langle \lfloor \vec{s}' \cdot p/2^{\eta+1} \rceil, \vec{c}' \right\rangle &= q_3 p - q_3 + q_3 + m \cdot \frac{p}{2} - \frac{m}{2} + \frac{m}{2} + \frac{p}{2^\eta} \cdot r^* + v_2 \\
&= (2q_3 + m) \cdot \frac{p-1}{2} + \left(q_3 + \frac{m}{2} + \frac{p}{2^\eta} \cdot r^* + v_2 \right) \\
&= (2q_3 + m) \cdot \frac{p-1}{2} + r_2^*
\end{aligned}$$

where $|r_2^*| \leq 2^{\rho' - \eta + 4}$. Therefore, we input this result into $(*)$ to get:

$$\begin{aligned}
c' &= 2p^2 \cdot < \vec{q}, \vec{c}' > +2 < \vec{r}, \vec{c}' > +(2q_3 + m) \cdot \frac{p-1}{2} + r_2^* \\
&= 2 < \vec{q}, \vec{c}' > \cdot p^2 + (2q_3 + m) \cdot \frac{p-1}{2} + (2 < \vec{r}, \vec{c}' > + r_2^*) \\
&= 2q'' \cdot p^2 + (2q_3 + m) \cdot \frac{p-1}{2} + r' \\
&= r' + (m + 2q_3) \cdot \frac{p-1}{2} + (2q'') \cdot p^2
\end{aligned}$$

where $|r_1^*| \leq |r_2^*| + \eta\theta 2^{\rho+1} \leq 2^{\rho' - \eta + 4} + \eta\theta 2^{\rho+1} \leq 2^{\rho' - \eta + 5}$.

Therefore, we converted the Type II ciphertext back to a Type I ciphertext with noises r' and q_3 with noise size $(\rho' - \eta + 5, log_2\theta)$.

4.3.4 Decrypt

In order to complete our proof of correctness we need to ensure that the decryption of a ciphertext returns its corresponding plaintext. This is shown as follows.

In previous sections we have shown that the output ciphertext of encryption, homomorphic addition and homomorphic multiplication is of Type I:

$$c = r + (m + 2r^*) \cdot (p-1)/2 + q \cdot p^2$$

where m is our original plaintext. Applying the decryption procedure, we get:

$$\begin{aligned} m &\leftarrow ((2c) \mod p) \mod 2 \\ &\leftarrow (2(r + (m + 2r^*) \cdot (p-1)/2 + q \cdot p^2) \mod p) \mod 2 \\ &\leftarrow ((2r + m \cdot p - m + 2r^* p - 2r^* + 2q \cdot p^2) \mod p) \mod 2 \\ &\leftarrow (-m \mod p) \mod 2 \\ &\leftarrow (m \mod p) \mod 2 \end{aligned}$$

This concludes our proof of correctness.

4.4 Proof of Semantic Security

We prove the semantic security of this scheme by reducing it to the hardness of the Decisional-Approximate-GCD problem.

Recall that $D^{\rho}_{p,q_0} = \{q \cdot p^2 + r : q \in \mathbb{Z} \cap [0, q_0), r \in \mathbb{Z} \cap (-2^{\rho}, 2^{\rho})\}$. The variant of the Decisional-Approximate-GCD problem is defined as follows:

Definition 4.4.1. *((ρ, η, γ)–Decisional-Approximate-GCD) Let p be a random odd integer of η bits, q_0 an integer uniformly distributed in $[0, 2^{\gamma}/p^2)$, r_0 an integer uniformly distributed in $(-2^{\rho}, 2^{\rho})$. Given $x_0 = q_0 \cdot p^2 + r_0$, polynomially many samples from D^{ρ}_{p,q_0} and $y \leftarrow D^{\rho}_{p,q_0} + (p-1)/2$, determine $b \in \{0,1\}$ from $c = x + b \cdot r \mod x_0$ where $x \leftarrow D^{\rho}_{p,q_0}$ and $r \leftarrow [0, x_0) \cap \mathbb{Z}$.* [7]

As described in [7] we use the following theorem to provide a proof of semantic security for this scheme. This theorem only considers a subset of the scheme without the procedure *Convert* (without parameters z and σ). In order to prove the semantic security of the full scheme, simply add the parameters z and σ to the above definition.

Theorem 4.4.2. *The above scale invariant DGHV scheme without the parameters z and σ is semantically secure under the (ρ, η, γ)-Decisional-Approximate-GCD assumption.* [7]

Before we proceed with the proof of Theorem 4.4.2, we list the following lemma which will be needed in our proof.

Lemma 4.4.3. *For the parameters (ρ, η, γ), let $pk = (x_0, \{x_i\}_i, y)$ and $sk = p$ be chosen as in the KeyGen procedure. Define $pk' = (x_0, \{x'_i\}_i, y)$ for x'_i uniformly generated in $[0, x_0)$. Then pk and pk' are indistinguishable under the Decisional-Approximate-GCD assumption.* [7]

In other words, this lemma shows that if you are capable of distinguishing public key elements from elements randomly sampled in $[0, x_0)$, then you are capable of solving the Approximate-GCD problem. Therefore, it must be that the public key elements and elements randomly sampled in $[0, x_0)$ are indistinguishable. The proof of this Lemma is provided in Appendix B.

Proof. (Theorem 4.4.2)

Game 0. This is the original attack game described in Section 3.1. The challenger generates a random key pair $(pk, sk) \leftarrow KeyGen(1^\lambda)$ and sends the public key pk to the attacker A. The attacker sends m_0, m_1 to the challenger. The challenger then generates a random bit, $b \leftarrow 0, 1$ and the ciphertext $c \leftarrow Encrypt(pk, m_b)$. The attacker then outputs $b' \in \{0, 1\}$. We define S_0 to be the event that $b = b'$ in Game 0.

Game 1. This is a transition based on indistinguishability. We uniformly draw elements in $[0, x_0)$ and replace the x_i's in the public key with these elements. We apply the same game scenario except with the uniformly drawn elements and set S_1 to be the event that, $b = b'$ in Game 1. Using Lemma 4.4.3, we get

$$|Pr[S_1] - Pr[S_0]| \leq \tau \cdot \varepsilon_{dagcd} \tag{1}$$

where $\varepsilon_{dagcd} = \epsilon/\tau$ is the advantage we obtained in the proof of Lemma 4.4.3. Note that ε_{dagcd} is negligible by the Approximate-GCD assumption.

Game 2. This is also a transition based on indistinguishability. In this game we apply the same game scenario except that we replace the ciphertext given to the attacker with a uniform integer modulo x_0. This completely removes any information given to the attacker on b. Let S_2 be the event that $b = b'$ in Game 2. It is clear that $Pr[S_2] = 1/2$, since the attacker has no information on b and therefore only has a 50% chance of choosing the correct value. Now, using the Leftover Hash Lemma (Lemma 2.4.5) we have $\sum_{i \in S} x_i \mod x_0$ is ε-statistically indistinguishable from the uniform distribution modulo x_0 with $\varepsilon = 2^{(\gamma - \tau)/2}$.

Therefore, we have

$$|Pr[S_2] - Pr[S_1]| \leq \varepsilon. \tag{2}$$

Note that ε is made negligible if the constraints in Section 4.1 are followed. Combining (1) and (2) we get:

$$
\begin{aligned}
|Pr[S_2] - Pr[S_0]| &= |Pr[S_2] - Pr[S_1] + Pr[S_1] - Pr[S_0]| \\
&= |Pr[S_2] - Pr[S_1]| + |Pr[S_1] - Pr[S_0]| \qquad (\triangle - inequality) \\
&= \varepsilon + \tau \cdot \varepsilon_{dagcd}
\end{aligned}
$$

Since $Pr[S_2] = 1/2$, we have $|Pr[S_0]| \leq 1/2 + \varepsilon + \tau\varepsilon_{dagcd}$. Thus, the attacker has negligible advantage in winning Game 0 (i.e. in breaking this encryption scheme) and therefore, the scheme is semantically secure. ■

5 References

[1] Rivest, R. L., Adleman, L. & Dertouzos, M. L. (1978). On Data Banks and Privacy Homomorphisms. *Foundations of Secure Computation*, Academia Press, 169-179.

[2] C. Gentry, *A Fully Homomorphic Encryption Scheme*, Stanford University PhD thesis, 2009, http://crypto.stanford.edu/craig/craig-thesis.pdf

[3] Dasari, V., Palaparthy, A., & Peddinti, K. (2014). *Analysis of Different Fully Homomorphic Encryption Schemes.* Retrieved from http://ece.gmu.edu/coursewebpages/ECE/ECE646/F15/project/F15 _Project_Resources/F14_Homomorphic_Encryption_report.pdf

[4] Shoup, V. & Puniya, P. (2007). *Advanced Cryptography: Lecture 8.* Lecture, https://cs.nyu.edu/courses/spring07/G22.3220-001/lec8.pdf.

[5] Gentry, Craig. (2010) *Computing arbitrary functions of encrypted data.* Communications of the ACM 53.3: 97-105.

[6] Van Dijk, Marten, et al. (2010). *Fully homomorphic encryption over the integers.* Advances in cryptology–EUROCRYPT 2010. Springer Berlin Heidelberg. 24-43.

[7] Coron, Jean-Sébastien; Lepoint, Tancrède; Tibouchi, Mehdi (2014). *Scale-Invariant Fully Homomorphic Encryption over the Integers.* PKC 2014. Springer.

[8] CHEN, W. (1997). *Linear Algebra* (pp. 1-16). Retrieved from https://rutherglen.science.mq.edu.au/wchen/lnlafolder/la09.pdf

[9] Goldreich, O. (2003). *Foundations of cryptography.* Cambridge, UK: Cambridge University Press.

[10] Shelat, A. (2007). *Indistinguishability and Pseudo-Randomness.* Retrieved from https://www.cs.virginia.edu/ shelat/651/www/chap3.pdf

[11] Merca, M. (2011). Inequalities and Identities Involving Sums of Integer Functions. *Journal Of Integer Sequences.* Retrieved from https://cs.uwaterloo.ca/journals/JIS/VOL14/Merca/merca3.html.

[12] Saxena, N. (2009). *Computational Security; Symmetric Key Encryption.* Lecture. Retrieved from http://isis.poly.edu/courses/cs6903-s09/Lectures/lecture3.pdf.

6 Appendix A: Bound Calculation

We want to prove that $|r^*| \leq 2^{\rho'-\eta+3}$, i.e. we want to prove: $-2^{\rho'-\eta+3} \leq r^* \leq 2^{\rho'-\eta+3}$. We have:

$$r^* = r \cdot (2^\eta/p^2) - c \cdot \epsilon + \delta_1 - m \cdot (2^\eta/2p^2)$$

$\underline{r^* \leq 2^{\rho'-\eta+3}:}$

$$\begin{aligned}
r^* &= r \cdot (2^\eta/p^2) - c \cdot \epsilon + \delta_1 - m \cdot (2^\eta/2p^2) \\
&\leq r \cdot (2^\eta/p^2) + \delta_1 \\
&\leq 2^{\rho'} \cdot 2^\eta/2^{2\eta-2} + \Theta/2 && \text{(since } p \in [2^{\eta-1}, 2^\eta - 1]\text{)} \\
&\leq 2^{\rho'-\eta+2} + 2^{\rho'-\eta-\rho_1-5} && \text{(since } \rho' = \eta + \rho_1 + log_2\Theta + 4\text{)} \\
&\leq 2^{\rho'-\eta+2}(1 + 2^{-\rho_1-7}) \\
&\leq 2^{\rho'-\eta+2}(2) \\
&= 2^{\rho'-\eta+3}
\end{aligned}$$

$\underline{-2^{\rho'-\eta+3} \leq r^*:}$

$$\begin{aligned}
r^* &= r \cdot (2^\eta/p^2) - c \cdot \epsilon + \delta_1 - m \cdot (2^\eta/2p^2) \\
&\geq -m \cdot (2^\eta/2p^2) - c \cdot \epsilon \\
&\geq -m \cdot (2^\eta/2p^2) - 2^\kappa \cdot 2^{-\kappa} \\
&\geq -1 \cdot 2^{-\eta-1} - 1 \\
&\geq -2^{\rho'-\eta+3}
\end{aligned}$$

Note that the other bounds stated in this paper can be proved in a similar fashion.

7 Appendix B: Proof of Lemma 4.4.3

Lemma 7.0.1. *For the parameters (ρ, η, γ), let $pk = (x_0, \{x_i\}_i, y)$ and $sk = p$ be chosen as in the KeyGen procedure. Define $pk' = (x_0, \{x'_i\}_i, y)$ for x'_i uniformly generated in $[0, x_0)$. Then pk and pk' are indistinguishable under the Decisional-Approximate-GCD assumption.* [7]

Proof. Assume that there exists a polynomial time algorithm β capable of distinguishing pk and pk' with advantage ϵ. Define,

$$pk^{(r)} = (x_0, \{x_1^{(r)}, ..., x_r^{(r)}, x_{r+1}^{(r)}, ..., x_\tau^{(r)}\}, y)$$

where $x_1^{(r)}, ..., x_r^{(r)} \leftarrow [0, x_0)$ and $x_{r+1}^{(r)}, ..., x_\tau^{(r)} \leftarrow D_{p,q_0}^\rho$. Then, by our assumption β can distinguish $pk^{(0)} = pk$ from $pk^{(\tau)} = pk'$ with advantage ϵ. By the hybrid

argument (Lemma 2.1.4), we conclude that there must exist an r such that β distinguishes $pk^{(r)}$ and $pk^{(r+1)}$ with advantage ϵ/τ. However, setting $x_r^{(r)}$ to be the Decisional-Approximate-GCD challenge, we conclude that β can solve the Decisional-Approximate-GCD problem with advantage ϵ/τ. ■