# Homomorphic Encryption

Ben LeVeque

April 5, 2013

# Contents

## 0.1 Introduction and background

In an age of ubiquitous computing, digital security is an important aspect of everyday life. As computations become more expensive and time-consuming, and as the need for data storage increases, cloud computation is rising in popularity. With this rise comes the need for a new class of security protocols that can allow data to be stored in encrypted form yet manipulated in a meaningful way by third parties.

### 0.1.1 Introduction to cryptographic concepts

Hello

Introduce idea of FHE. Define terms (plaintext, ciphertext, message, message space, ciphertext space, cryptosystem, encryption, key generation, decryption $\mathbb{N} = \mathbb{Z}_{\geq 0}$).

Imagine you have a large collection of data points and want to compute the mean or standard deviation. You have have two options: you can either do this computation yourself at a potentially high computational cost, or you can delegate this work to another party and risk losing privacy. The goal of fully homomorphic encryption is to minimize both potential costs by allowing secure delegation. In practice, this might mean being able to store data on The Cloud in an encrypted form such that a third party could still manipulate the data in a meaningful way. By meaningful, we mean that any combination of sums and products of encryptions decrypt to the corresponding sums and products of the original, unencrypted data. We call such a combination an *arithmetic circuit*, or just a *circuit*. If this were possible, data could be encrypted, operated upon, and decrypted in a completely well-defined manner. What we are looking for, then, is an encryption function that is a *ring homomorphism* from the message space $\mathcal{M}$ to the ciphertext space $\mathcal{C}$. Recall that a ring homomorphism is a map $\sigma : \mathcal{M} \to \mathcal{C}$ such that for any $x, y \in \mathcal{M}$,

$$\sigma(x) + \sigma(y) = \sigma(x + y)$$

$$\sigma(x)\sigma(y) = \sigma(xy)$$

If our encryption function $e$, then, is a ring homomorphism, we could add or multiply two encryptions and the result would be the encryption of the sum or product of the corresponding plaintexts. A cryptosystem employing such an encryption function is said to give *fully homomorphic encryption* (FHE). If the encryption function in question is a homomorphism in most circumstances, but is not guaranteed to give well-defined decryption after the application of every circuit, the scheme is said to provide *somewhat homomorphic encryption*. Such a scheme might arise if the decryption function requires its input to be of a certain size, which large circuits might exceed. For motivation, consider the following toy examples.

First, we will construct a somewhat homomorphic encryption scheme that provides very little security but illustrates the concepts presented above. Consider the message space $\mathcal{M} = \mathbb{N}$ and choose as a secret key some prime number $p$. We will encrypt a message $m \in \mathcal{M}$ by setting $e(m) = m + ap$ for some random integer $a$. Decryption is then performed by reducing modulo $p$. As long as the message $m$ is less than $p$, this scheme is perfectly well defined, since

$$d(e(m)) = m + ap \pmod{p} = m$$

However, if $m \geq p$, then $d(e(m))$ will no longer equal $m$, but will be the reduction of $m$ modulo $p$. One way to attempt to avoid this problem is to encrypt values only less than $p$. However, once we start considering circuits applied to encryptions, this idea shows its flaws; the sum of two encryptions of $p-1$, for example, would really be an encryption of $2p-2$, but it would encrypt to $-2 \equiv p-2 \pmod{p}$. Another solution is to recognize the limitations of the system and include as a part of the system's protocol a bound on the number of additions and multiplications that can be performed on encryptions before the result should be decrypted.

To make this scheme fully homomorphic, we could set $\mathcal{M} = \mathbb{Z}/p\mathbb{Z}$, since in this case, reducing modulo $p$ during decryption is consistent with the structure of the message space. However, it is perhaps unnatural to for messages to live in $\mathbb{Z}/p\mathbb{Z}$, since when we compute, for example, standard deviations, we don't want to reduce modulo $p$ at any point during the computation. In this case, it may be best to accept the limitations of the somewhat homomorphic scheme above and work within its framework.

It should be mentioned, as well, that the scheme above is just a "toy" example because it is very insecure. Suppose an eavesdropper asked us to encrypt the message $m = 0$ using this system. The result would be a multiple of $p$. Suppose we were asked to perform many such encryptions. We would end up with a collection of multiples of $p$. With this information, it would be very fast to compute the greatest common divisor of these values using the Euclidean Algorithm, and the result would likely be $p$ itself. Using $p$, the eavesdropper could then decrypt any message he pleased by simply reducing it modulo $p$. This type of attack—using encryptions of the message 0 to reveal an essential feature of the structure of the scheme itself—will appear again, and is a necessary threat to consider when formulating encryption schemes. This attack and others is discussed in the appendix.

In his 2009 dissertation CITATION, Craig Gentry introduced a fully homomorphic encryption scheme based on the theory of ideal lattices...

This thesis will present two encryption schemes that explore their security and efficiency.

## 0.2 The schemes under investigation

In this project, we formulate two schemes that show promise as either somewhat or fully homomorphic encryption schemes. The first, which is formulated entirely over the integers, relies on the hardness of making $N$ consecutive correct choices, where each choice is between two values. The second, which is formulated over multivariate polynomial rings, relies on the hardness of the ideal membership problem (IMP) in this setting. Schemes such as this are sometimes referred to as "Polly Cracker schemes," and have

### 0.2.1 Choice-dependent encryption

One way to formulate a system is entirely over the integers, where the hardness comes in requiring Eve to correctly pick $N$ numbers in a row, where there are two options for each choice. With sufficiently large $N$, trying all $2^N$ combinations of choices will be computationally infeasible.

### Scheme

The scheme goes as follows:

- Key generation
  1. Pick a prime $P$; the message space is $\mathcal{M} := \mathbb{Z}/p\mathbb{Z}$
  2. Pick an integer $K$; this is the number of ways we can mask a given message
  3. Pick $N \in \mathbb{Z}$ so that $2^N$ is sufficiently large
  4. Pick primes $\{p_i\}_{i=1}^N$ such that $(K+1)P < \prod p_i$
  5. Pick primes $\{q_i\}_{i=1}^N$
  6. Return private key $(K, P, \{p_i\})$ and public key $(N, \{p_i q_i\})$

- Encryption $(e)$
  1. Pick a message $m \in \mathcal{M}$
  2. Pick random integers $\{a_i\}_{i=1}^N$
  3. Pick a random integer $k < K$
  4. Return $e(m) := \{m + a_i p_i + kP \pmod{p_i q_i}\}_{i=1}^N$

- Decryption $(d)$
  1. For all $i = 1, \ldots, N$, reduce the $i$th component of the encryption modulo $p_i$
  2. Run the Chinese Remainder Theorem on the resulting components, which gives $m + kP$ $(\text{mod } \prod p_i)$
  3. Reduce the result modulo $P$ to retrieve $m$

### Explanation

Before proving the correctness and analyzing the characteristics of this scheme, it is helpful to have some motivation for and explanation of the different components, beginning with the public and private keys. The prime $P$ is the size of our message space . If The value $N$ is a security parameter; changing its value affects the hardness of the underlying problem of choosing $N$ consecutive values correctly. Increasing $N$ makes the scheme more secure, but it also increases the size of ciphertexts, so we pay a price in efficiency. The integer $K$ whose value affects the message space size and the hardness of , the . Increasing $N$ "increases" the product $\prod p_i$, since there are more primes involved, which therefore increases the potential value of $P$

The hardness is in the fact that each pair $\{p_i, q_i\}$ is known from $p_i q_i$, but correctly deciding which of the two is $p_i$ requires a choice. Choosing the wrong value at one component will give an incorrect decryption, since reducing modulo $q_i$ will not eliminate the multiple of $p_i$ in the encryption. In the worst case for Eve, then, it will take her (naively, at least) $2^N$ tries to decrypt the message. It can be noted, too, that an incorrect choice at any point will result in a completely incorrect decryption, since reducing modulo $q_i$ at any point could give a drastically different reduction in that component, which will cause the Chinese Remainder Theorem to return an entirely incorrect value.

The size of the primes $p_i$ and $q_i$ is up to the user; larger primes allow for a larger message space, but they also lead to larger ciphertexts. This is a trade-off between efficiency and security that will be analyzed in more depth later on, but for now, we note that it is also possible to create a larger message space by increasing the size of $N$. This will also affect the size of a cipher text, but it has the potential benefit of keeping the values in each component relatively smaller.

The integer $K$ represents the number of ways we can mask or add noise a message with multiples of $P$. If this masking was not performed, then it would be easy for an eavesdropper to determine the primes $p_i$ from encryptions of zero. Consider a collection of $r$ encryptions of zero:

$$\begin{aligned} e(0) &= (a_{1,1}p_1 \pmod{p_1q_1}, \ldots, a_{1,N}p_N \pmod{p_Nq_N}) \\ &\vdots \\ e(0) &= (a_{r,1}p_1 \pmod{p_1q_1}, \ldots, a_{r,N}p_N \pmod{p_Nq_N}) \end{aligned}$$

Since reducing an integer $l$ modulo $M$ is done by adding an appropriate multiple of $M$ to $l$, we can express the above as:

$$\begin{aligned} e(0) &= (a_{1,1}p_1 + b_{1,1}p_1q_1, \ldots, a_{1,N}p_N + b_{1,N}p_Nq_N) \\ &\vdots \\ e(0) &= (a_{r,1}p_1br, 1p_1q_1, \ldots, a_{r,N}p_Nb_{r,N}p_Nq_N) \end{aligned}$$

where the $b_{i,j}$ are the appropriate factors necessary to perform reduction in each component. Now, the $i$th component of each $j$th encryption is a multiple of $p_i$, namely $(a_{j,i} + b_{j,i}q_i)p_i$, so the component-wise GCD of the $r$ encryptions above will likely give us all of the $p_i$.

By setting the message space to be $\mathbb{Z}/P\mathbb{Z}$, we ensure that even if a sum or product of messages is greater than $P$, reducing modulo $P$ will give the well-defined sum or product in the message space.

### Correctness of encryption/decryption

Claim: If $m < P$, then $d(e(m)) = m$

Proof of Claim:

Let $(N, \{p_i\}, \{q_i\}, K, P)$ be our secret key. Then

$$e(m) = \{m + a_ip_i + kP \pmod{p_iq_i}\} \tag{1}$$

for some random $a_i$ and a random $k < K$. Then reducing each component modulo the corresponding $p_i$ gives

$$\{m + kP \pmod{p_i}\} \tag{2}$$

Now, using the Chinese Remainder Theorem gives

$$m + kP \pmod{\prod p_i} \tag{3}$$

Since we chose $m < P$ and $k < K$, and $P$ was defined such that $KP < \prod p_i$, we have that $m + kP < \prod p_i$, so

$$m + kP \pmod{\prod p_i} = m + kP \tag{4}$$

Now, reducing modulo $P$ gives $m$, since $m \in \mathbb{Z}/P\mathbb{Z}$.

### Correctness of homomorphic operations

Furthermore, the scheme is somewhat homomorphic (using component-wise addition and multiplication), since if $m_1, m_2 \in \mathcal{M}$, then

$$e(m_1) + e(m_2) = \{m_1 + m_2 + (a_i + b_i)p_i + (k_1 + k_2)P \pmod{p_i q_i}\} \tag{5}$$

This is of the form of a regular encryption of the message $m_1 + m_2$. The danger is that it is possible for $m_1 + m_2 + (k_1 + k_2)P > \prod p_i$, in which case the Chinese Remainder Theorem would give an incorrect result. Similarly,

$$e(m_1)e(m_2) = \{m_1m_2 + ((m_1+k_1)b_i + (m_2+k_2)a_i + a_ib_i)p_i + (m_1k_2 + m_2k_1 + k_1k_2P)P \pmod{p_i q_i}\} \tag{6}$$

Again, this is of the form of an encryption of the message $m_1m_2$, but we run into a similar as above. The danger is even greater now, since the numbers are growing multiplicatively instead of merely additively, now. Therefore, we will measure the homomorphicity in terms of the number of multiplications allowed by this system. As we see above, this depends on the size of the product of messages as well as the size of the product of messages and the values of $k$ that we pick.

### Analysis of homomorphicity

In terms of the dangers mentioned above, the worst case is $m \approx P$ and $k \approx K$. If all of our messages were on this order and were encrypted with random $k$ around $K$, then we would be allowed only $\log_{(K+1)P} \prod p_i$ multiplications, since each multiplication would roughly raise the message $m+kP \approx (K+1)P$ to a power. If we want a guarantee, then, that we can perform $M$ multiplications on encryptions, we should set $\{p_i\}$ such that $((K+1)P)^M < \prod p_i$. There are two ways to accomplish this: increase the size of each prime $p_i$, or increase $N$, the number of components in each encryption. This decision can be left to the user.

An additional concern is that our message space may not realistically resemble $\mathbb{Z}/p\mathbb{Z}$, for example if the information we want to encode is a bunch of integer values, and we want to compute the product of these integers, we expect the result of performing the encrypted operation to be the true product, even if it exceeds $P$. As soon as the results of computations exceed $P$, the scheme fails to return the correct value to the user, so practically speaking, we want our messages to be much smaller than $P$, which is in turn much smaller than $\prod p_i$. Precisely, this means that if $E := \log_{(K+1)P} \prod p_i$, then $|\mathcal{M}| < \sqrt[E]{P}$ guarantees that products of messages will always decrypt properly, since $E$ is the number of allowed multiplications from above.

### 0.2.2 Using multivariate polynomial rings

Another approach to homomorphic encryption is to use the properties of arithmetic in polynomial rings. The ring structure essentially guarantees that if our encryption involves only arithmetic operations, it will be homomorphic. Perhaps the most natural thing to try in this setting is the following algorithm: choose a principal ideal $(f) \subset \mathbb{Z}[x]$ and encrypt an integer message $m$ by adding a random element $af \in (f)$, so $e(m) = m + af$. Then to decrypt, we simply reduce modulo the polynomial $f$. This process is homomorphic because ideals are closed under addition and multiplication:

$$e(m_1) + e(m_2) = m_1 + m_2 + (a_1 + a_2)f = e(m_1 + m_2)$$

and

$$e(m_1)e(m_2) = m_1m_2 + (m_2a_1 + m_1a_2 + a_1a_2)f = e(m_1m_2)$$

Note that our definition of equality above is somewhat loose; since $e(m_1+m_2)$ encrypts by choosing a *random* multiple of $f$, it may not be exactly $(a_1+a_2)f$ (and likewise for the case of multiplication). The notion of equality we adopt, then, is a notion of coset equality: $e(m_1) + e(m_2)$ is in the same coset of $I$ as $e(m_1 + m_2)$, so they will decrypt identically, and the same is true for $e(m_1)e(m_2)$ and $e(m_1m_2)$.

While this scheme illustrates the ideas we will use later, it is not very secure on its own. For example, suppose an eavesdropper, Eve, asks us to encrypt $m = 0$ several times. The resulting ciphertexts would be a collection

$$\{a_1f_1, a_2f_2, \ldots, a_nf_n\}$$

Now, if Eve takes the greatest common divisor of the elements in this collection, there is a very high chance that the result will be $f$ itself or a small multiple of $f$. With $f$ in hand, Eve could then decrypt any message she pleased, and the scheme would be compromised.

The scheme above was easily broken because it relied on a fairly easy problem: given a collection of polynomials in a principal ideal, find a generator for the ideal. This is easily found because we can compute fairly efficiently the greatest common divisor of two polynomials in $\mathbb{Z}[x]$. This is a basic case of a more general problem:

**Ideal Membership Problem:** Given a ring $R$, a set of elements $\{r_1, \ldots, r_n\} \in R$, and an element $f \in R$, determine whether $f \in (r_1, \ldots, r_n) \subset R$.

If the structure of $R$ is simple, then this problem is correspondingly easy, as we saw above in the case of our set of generators being multiples of a single polynomial $f \in \mathbb{Z}[x]$. The ease of solution here comes from the fact that greatest common divisors are fairly easy to compute over $\mathbb{Z}[x]$; with the greatest common divisor of our generators available, we can easily test whether the gcd divides any other encryption to see if it is an encryption of zero. However, if we look instead at multivariate polynomial rings such as $\mathbb{Z}[x, y]$ or $\mathbb{Z}/p\mathbb{Z}[x, y]$, the problem might not be so simple. It is in this spirit that we formulate the following scheme:

**Scheme**

The scheme goes as follows:

- Key generation
  1. Pick a general degree bound $D$ and a coefficient bound $B$
  2. Pick a random number $z_0 < B$
  3. Pick a random polynomial $f$ with total degree less than or equal to $D$ and coefficients less than $B$
  4. Pick a random polynomial $g'$ of total degree less than or equal to $D - 1$
  5. Set $g := (y - z_0)g'$
  6. Return private key $(f, g, z_0)$

- Encryption ($e$)
  1. Pick a message $m \in \mathcal{M} := \mathbb{Z}$
  2. Pick random polynomials $a$ and $b$ which respect the degree and coefficient bounds
  3. return $e(m) := m + af + bg$

- Decryption ($d$)
  1. Evaluate an encryption at $z_0$
  2. Reduce the result modulo $f(x, z_0)$, a single-variable polynomial, and return the result

**Explanation**

We can use the results below to choose our degree and coefficient bounds appropriately to assure us of security. The idea behind our scheme is to give an encryption which is a multivariate polynomial (to increase the complexity of the Groebner basis computation), but which can be easily reduced during the decryption stage. This is accomplished by constructing our secret key $g$ such that it vanishes at the line $y = z_0$. We can then plug in $(x, z_0)$ to eliminate $g$, and we are left with $m + (af)(x, z_0)$. Reduction modulo single variable polynomials is easy to accomplish, so we can then reduce modulo $f(x, z_0)$ to retrieve $m$. Since reducing modulo a polynomial does not put any restrictions on the size of $m$ (as reducing modulo an integer would), $m$ can be arbitrarily large and still go through encryption and decryption in a well-defined manner.

### Correctness of encryption/decryption

We have broached this subject above, and now we will prove it

Claim: If $m \in \mathbb{Z}$, then $d(e(m)) = m$

Proof of Claim:

First, encryption gives us

$$e(m) = m + af + bg$$

for some $a$ and $b$, where we know that $g(x, z_0) = 0$ by construction. In order to decrypt, we first evaluate at $(x, z_0)$, which gives

$$e(m)(x, z_0) = m + a(x, z_0)f(x, z_0)$$

This is a single variable polynomial which can be reduced modulo the polynomial $f(x, z_0)$ (which is known to the decryptor, since $f$ and $z_0$ are both components of the secret key) to give $m$. This proves correctness.

### Correctness of homomorphic operations

Now we prove that sums and products of encryptions are valid encryptions of the sums and products of the corresponding plaintexts.

Claim: If $m_1, m_2 \in \mathbb{Z}$, then $e(m_1) + e(m_2) = e(m_1 + m_2)$

Proof of Claim:

Let us recall first that the equality above is a coset equality, since encryptions are equivalent if they ar

### Practical analysis of homomorphicity

### Generalizing

... One benefit to this scheme is that it can be easily generalized to $n$ variables – i.e. we can consider an analogous scheme over $\mathbb{Z}[x_1, \ldots, x_n]$. This is done by choosing $g$ such that $g(x_1, \ldots, x_n) = g_1(x_1)g_2(x_2)\cdots g_n(x_n)$ and $z_2, \ldots, z_n$ such that $z_i$ is a root of $g_i$. Then encryption proceeds as in the two-variable case:

$$e(m) \quad = \quad m + af + bg$$

and to decrypt, we first evaluate at $(x_1, z_2, \ldots, z_n)$ and then reduce modulo $f(x_1, z_2, \ldots, z_n) \in \mathbb{Z}[x_1]$. Further tests would be necessary to see if increasing the number of variables is an effective way to increase the computational complexity of Groebner basis computation.

## 0.3 Applications

Beyond theoretical formulation, it is important to see how these schemes might perform in practice. As mentioned in the introduction, the efficiency of homomorphic schemes has been a major obstacle to their dissemination, and

### 0.3.1 Application to medical needs

### 0.3.2 Application to finance

EXAMPLE OF STD-DEV

A significant drawback of using our choice-based encryption scheme for the manipulation of statistics is made apparent when we consider taking the standard deviation of many numbers. This requires being able to make a large number of well-defined multiplications, which may be impossible, especially if the numbers in question are large. Problems like this may be more suited to our multivariate scheme, which can make arbitrary numbers of well-defined multiplications. We may lose efficiency, but in computations that require correctness over

## 0.4 Implementation

## 0.5 Appendix A: Common Attacks

This section is a compilation of and reference for some of the types of attacks considered in the duration of this project.

- gcd attacks
- G.B. attacks (type of gcd)
- using known encryptions of zero to identify encryptions of the same value
-

## 0.6 Appendix B: An intro to Gröbner bases

## 0.7 Appendix C: Failed Attempts

## 0.8 Notes

Cite Donald Knuth, Art of Computer Programming, Addison-Wesley Publishing Company, Reading, Massachusetts, second addition, pp 273-4 for representation of multivariate polynomials using linked lists.