Homomorphic Encryption

Ben LeVeque

March 27, 2013

Contents

0.1 Introduction and background

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

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 $N \in \mathbb{Z}$ so that 2^N is sufficiently large
  2. Pick an integer K, which represents the number of ways we can mask a message
  3. Pick a prime P – this will be the message space size, so the message space is $\mathcal{M} := \mathbb{Z}/P\mathbb{Z}$
  4. Pick large 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 ith component of the encryption modulo p_i
  2. Run the Chinese Remainder Theorem on the resulting components, which gives $m + kP \pmod{\prod p_i}$
  3. Reduce the result modulo P to retrieve m

Explanation

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 a message with multiples of P. Note that if this masking were impossible, then it would be easy to determine the primes p_i from encryptions of zero by taking the component-wise GCD of several such encryptions:

$$\begin{aligned}
e(0) &= (a_{1,1}p_1 \pmod{p_1 q_1}, \ldots, a_{1,N}p_N \pmod{p_N q_N}) \\
&\vdots \\
e(0) &= (a_{r,1}p_1 \pmod{p_1 q_1}, \ldots, a_{r,N}p_N \pmod{p_N q_N})
\end{aligned}$$

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

<u>Claim</u>: If $m < M$, then $d(e(m)) = m$

<u>Proof of Claim</u>:

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

$$e(m) = \{m + a_i p_i + kP \pmod{p_i q_i}\} \tag{1}$$

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,

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

Again, this is of the form of an encryption of the message $m_1 m_2$, 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.

This scheme 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 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 $f \in \mathbb{Z}[x]$. 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. We start by looking at the former

Scheme

Explanation

Correctness of encryption/decryption

Correctness of homomorphic operations

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:

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

and to decrypt, we first evaluate at $(x_1, z_2, \ldots, z_n)$ and then reduce modulo $f(x_1, z_2, \ldots, z_n) \in \mathbb{Z}[x_1]$.

0.3 Applications

0.3.1 Application to medical needs

0.3.2 Application to finance

0.4 Implementation

0.5 Appendix A: Common Attacks

0.6 Appendix B: An intro to Gröbner bases

0.7 Appendix C: Failed Attempts