An Overview of Gentry's Fully Homomorphic Encryption Scheme

Adam Risi Ross Snider

May 12, 2010

Contents

1 Introduction **2**

2 History **2**

3 Background **3**
3.1 Multiplicative Homomorphism in RSA 3
3.2 Partially Homomorphic and Fully Homomorphic Encryption . . . 3
3.3 Constructing a Boolean Circuit 3

4 Gentry's Homomorphic Encryption Scheme **3**
4.1 Partially Homomorphic Encryption 3
4.2 Noise 4
4.3 Fully Homomorphic Encrpytion 4
4.4 Construction 4
4.4.1 Approximate Greatest Common Divisor 4
4.4.2 Sparse Subset Sum 5
4.5 Bootstrapping 5

5 Fully Homomorphic Encryption in the Future **5**

Abstract

In a 1978 paper[2] inspired by the multiplicative homomorphic properties of the RSA cryptosystem, Rivest, Shamir and Dertouzos detailed essential properties of what they called a privacy homomorphism a cryptographic scheme that would allow third party processing of encrypted data without access to any plaintext or private key information. While partial strides toward such a system have been made over the past thirty years, most cryptographers believed that no such scheme could actually exist. Craig Gentry of Stanford University fully realized such a scheme in his 2009 PhD dissertation A Fully Homomorphic Encryption Scheme Over Ideal Lattices.[1] In this paper, we describe a similar (fully homomorphic) scheme over the integers (also invented by Gentry), which remains conceptually identical to the original lattice scheme but with the added benefit that it is much simplier.

1 Introduction

In his 2009 paper "Fully Homomorphic Encryption over the Integers"[**?**], Craig Gentry made two important contributions to the field. First, he defined a partially homomorphic scheme which was interesting in its own right. This scheme was robust enough to allow third party contributions of a nearly arbitrary number of additions and many multiplications, beating out the Boneh-Goh-Nissim cryptosystem, which remained a leader in the field for quite some time.

Unfortunately, Gentry's partially homomorphic scheme suffered from the fact that as computations are performed on the ciphertexts, noise is introduced in the output ciphertexts. This limited the malleability of the scheme after some number of computations, the error vectors would grow too large and decryption would become invalid.

Gentry realized that if he could find some way to eliminate the growing noise implicit in the scheme, he could modify his original scheme to allow an arbitrary number of additions and multiplications. In turn, this would allow him to perform any computable function on encrypted data.

Aware that successfully decrypting and subsequently re-encrypting a given ciphertext would eliminate its noise vector, Gentry asked 'why not allow the third party to decrypt the ciphertexts homomorphically?' The third party would be able to remove the growing noise from ciphertexts. At the same time, asking the third party to do the decryption homomorphically would maintain the privacy of the private key.

If the noise introduced by a homomorphic evaluation of the decryption function is less than what the scheme is capable of handling, then the scheme can chain together homomorphic evaluations of decryption with homomorphic evaluations of the target computable functions, resulting in a scheme that can handle a computable function of any complexity.

2 History

While a Fully Homomorphic Encryption scheme is new, the idea is not. The inspiration for a fully homomorphic encryption system came from an interesting property in RSA.

$$c = m^e \bmod n \tag{1}$$
$$m = c^d \bmod n \tag{2}$$

If we multiply the ciphertext c by y^e, we can see that the decryption of cy^e will yield my, or the plaintext multiplied by the value y. This *partial homomorphism* lead to greater investivation by Rivest, Adleman, and Dertouzos.

Proposed in 1978 by Rivest, Adleman, and Dertuzos, "privacy homomorphisms" are the first known example of what came to be called "fully homomorphic encryption." Rivest, Adleman, and Dertuzos proposed a thought experiment: Imagine a loan company who stored its data at an off site location. Since the data about loans is sensitive, the loan company decided to have all of its data encrypted. Now, a problem becomes apparent: if the loan data is stored off site, then how would the loan company be able to query it? Rivest, Adleman, and Dertuzos proposed that "privacy homomorphisms" could be used, a theoretical encryption system where data could be stored, and queried, in a completely encrypted environment.

Between 1978 and today, a number of other schemes have been shown to be partially homomorphic. Paillier, Boneh-Goh-Nissim, and ElGamal all exhibited partial homomorphism over some operations.

3 Background

3.1 Multiplicative Homomorphism in RSA

In RSA, the encryption function ε and decryption function δ provide a multiplicative homomorphism. First, recall:

$$\varepsilon_{RSA}(m) = m^e \bmod n \tag{3}$$
$$\delta_{RSA}(c) = c^d \bmod n \tag{4}$$
$$de = 1 \bmod \phi(n) \tag{5}$$
$$m^{de} \bmod n = m \bmod n \tag{6}$$
$$\tag{7}$$
$$\varepsilon_{RSA}(x)\varepsilon_{RSA}(y) = \tag{8}$$

3.2 Partially Homomorphic and Fully Homomorphic Encryption

3.3 Constructing a Boolean Circuit

4 Gentry's Homomorphic Encryption Scheme

4.1 Partially Homomorphic Encryption

To begin, Gentry proposes a simple partially homomorphic encryption scheme over the integers:

Key Generation: The key, p, is a randomly generated odd integer.

Encryption: To encrypt a single bit $m \in \{0, 1\}$, pick a ciphertext integer whose residue mod p has the same parity as m. This ciphertext integer can be described as $c = pq + m$, where p is the private key, and q is a λ bit integer.

Decryption: Decryption under this scheme is simple, the given a ciphertext c, $m = (c \bmod p) \bmod 2$.

We can show that this is homomorphic by looking showing the multiplication and addition of two cithpertexts, $\varepsilon(x)$ and $\varepsilon(y)$.

Multiplication

$$\begin{aligned} \varepsilon(x) \times \varepsilon(y) &= (m_1 + pq_1)(m_2 + pq_2) \\ \delta(\varepsilon(x) \times \varepsilon(y)) &= m_1 m_2 + m_1 pq_2 + m_2 pq_1 + p^2 q_1 q_2 \bmod p \\ &= m_1 m_2 \bmod 2 \end{aligned}$$

Addition

$$\begin{aligned} \varepsilon(x) + \varepsilon(y) &= m_1 + pq_1 + m_2 + pq_2 \\ \delta(\varepsilon(x) + \varepsilon(y)) &= m_1 + m_2 + pq_1 + pq_2 \bmod p \\ &= m_1 + m_2 \bmod 2 \end{aligned}$$

To make this a public key system, we simply issue the 3rd party a series of encryptions of 0. The 3rd party can then add one to any encryption of 0 to have it decrypt to 1, and leave it alone to have it decrypt to 0.

4.2 Noise

As sequential operations are preformed on cyphertext, a number of excess "noise" is introduced. For example, by preforming a single addition and multiplication, a large term appears:

$$\begin{aligned} \varepsilon(x) + \varepsilon(y) \times \varepsilon &= (m_1 + m_2 + pq_1 + pq_2)(m_3 + pq_3) \\ &= m_1 m_3 + m_2 m_3 + p^2 q_1 q_3 + p^2 q_2 q_3 \\ &= m_3(m_1 + m_2) + p^2(q_1 q_3 + q_2 q_3) \end{aligned}$$

4.3 Fully Homomorphic Encrpytion

4.4 Construction

Gentry proposes a fully homomorphic encryption scheme that relies on two different "difficult problems" for security, the **Approximate Greatest Common Divisor** problem, and the **Sparse Subset Sum** problem.

4.4.1 Approximate Greatest Common Divisor

The approximate GCD problem is, given a set of $n_i + pq_i$, determine p. The best known algorithm is exponential time in the size of inputs.

4.4.2 Sparse Subset Sum

The sparse subset sum problem is a variant of the common subset sum problem. In the classic Subset Sum problem, we are given a set $\sim$ and a target value t, and are asked for a subset of $\sim$ whose *sum* is equal to t. In the *Sparse Subset Sum* problem, the simple addition that the cardinality of the answer set must be much smaller than the cardinality of $\sim$. This problem, and its parent problem (Subset Sum), have been shown to be intractable reasonably sized $\sim$.

4.5 Bootstrapping

5 Fully Homomorphic Encryption in the Future

References

[1] Craig Gentry. Fully homomorphic encryption using ideal lattices. In STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing, pages 169–178, New York, NY, USA, 2009. ACM.

[2] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. pages 169–177. Academic Press, 1978.