PAPER - REPORT
COMPUTING FUNCTIONS ON ENCRPYTED DATA

(Bhuvnesh Jain, 2014A7PS028P)

Abstract

This research paper describes the schemes to perform computation on encrypted data, namely Fully Homomorphic Encryption (FHE). A cryptosystem that supports arbitrary computation on ciphertexts is known as fully homomorphic encryption (FHE). The goal of the paper was to equip the reader with enough insights to pursue research in the field further.

Introduction

In recent times, there has been an increase in the amount of data that is being available on cloud. Ranging from medical data, legal documents and other sophisticated data, everything is moving towards this fast growing technology. The questions which arises here is about security. With help with subjects like Cryptography and network security and the advancements that have taken place in this area, there is no doubt that data can be efficiently encrypted and stored on cloud safely. But the question which arises is whether we can compute some arbitrary functions over the encrypted data without decrypting it at any stage and giving the output also in encrypted form only.

An encryption algorithm is **malleable** if it is possible for an adversary to transform a ciphertext into another ciphertext which decrypts to a related plaintext i.e. given an encryption of a plaintext ‘m’, it is possible to generate another ciphertext ‘c’ which decrypts to $f(m)$, for a known function ‘f’ without necessarily knowing or learning about ‘m’.

The simplest example of malleable encryption algorithm is OTP, or in general stream ciphers. Here the ciphertext is given by the relation $c = m \oplus k$, for given message ‘m’ and key ‘k’. This ciphertext can be easily transformed to another cipher text c' when the function $f(m) = m \oplus t$. The new ciphertext is given by $c' = c \oplus t$. In most cases, the property of malleability is undesirable.

But looking this from the point of the topic of paper, this may be a desirable only when notion of sematic security of system is not compromised. Also, a cryptographic system can be semantically secure against chosen plaintext attacks or even non- adaptive ciphertext attacks, although it being malleable. But security against adaptive ciphertext attacks leads to non-malleability in the cryptographic system.

Homomorphic encryption

It is a form of encryption that allows computations to be carried out on ciphertext, thus generating an encrypted result which, when decrypted, matches the result of operations performed on the plaintext. . Homomorphic encryption would allow the chaining together of different services without exposing the data to each of those services. Homomorphic encryption schemes are malleable by design. Although a crypto-system which is unintentionally malleable can be subject to attacks on this basis, if treated carefully homomorphism can also be used to perform computations securely. In mathematical terms, homomorphic encryption can be stated as follows:

Given messages m_1, m_2, …… m_k and their corresponding ciphertexts as c_1, c_2, ……. c_k. We need to compute the ciphertext 'c' for the function 'f' defined for f(m_1, m_2, ….. m_k). While carrying out this process, we should at no stage decrypt back any of the ciphertext , c_i, to be able to evaluate our final ciphertext.

Now, we know that every function can be decomposed to a set of simple machine level instructions consisting of add, multiply, subtract only. Even the basic gates like and, or and not can be represented in terms of add, multiply and subtract.
For example: $x \,\&\, y = x\,.y,\quad x \mid y = 1-(1-x)(1-y),\quad \sim x = 1-x$

Thus, we discuss below only the process of carrying out these functions only on the encrypted data. In the whole report we consider the following representations and notions. Also, we will discuss the asymmetric cryptographic systems only, the symmetric case can be dealt similarly.

We have an encryption scheme E, having the following basic algorithms:

1. **KeyGen(E)**
   It generates the key used for the cryptographic system. In case of symmetric encryption, only one key is generated which is used both for encryption and decryption. In case of asymmetric encryption, 2 keys, namely one public key, p_k, and another private key, s_k, is generated.
2. **Encrypt(pk, m)**
   It encrypts the given message 'm' to ciphertext 'c' based on the proposed algorithm using the key 'k'.
3. **Decrypt(sk, c)**
   It decrypts the given ciphertext 'c' into the corresponding message 'm' using the key 'k' with the guarantee that
   $$Decrypt\big(sk, Encrypt(pk, m)\big) = m$$
4. **Evaluate(E)**
   It is associated with a set of permitted functions $F(E)$. For any function f in $F(E)$, and ciphertexts c_1, c_2, ……. c_k and corresponding messages m_1, m_2, …… m_k, it outputs the ciphertext 'c' which is basically encryption of the function f on the above messages.
   $$c = Encrypt(K, f(m_1, m_2 \dots\dots\, m_k))$$
   For a function f which is not present in $F(E)$, the output of Evaluate(E) can be considered as undefined.

All the above algorithms should be efficient i.e. they run in polynomial time on the number of bits in the input.

Points to keep in mind

Now while carrying out the process of performing functions on encrypted data the following points must be kept in mind –

1. **Functionality**

   The Evaluate functions should have execution time which is independent of the complexity of function f from the set of permitted functions. Also, it should be atleast linear in the number of ciphertexts given in the input. If it is not the case, then there is some security issue in the sense that the function doesn't depend on all of its messages. Also, the output of the evaluate function should be a ciphertext 'c' which is of the same length as the other ciphertexts provided.

2. **Security**

   The weakest requirement for an encryption scheme is one-wayness, i.e. given the public key p_k and a ciphertext 'c' that encrypts unknown message 'm' under p_k, it should be "Hard" to output 'm'. "Hard" means that any algorithm 'A' used by the adversary that runs in polynomial time has a negligible probability of success over the choices of p_k and m. For this we require that the encryption algorithm be not deterministic i.e. there is only one ciphertext that encrypts a given message. But since, we are allowing computation of functions over encrypted data, Evaluate algorithm allows the process of breaking E easier as it gives the attacker more power. Although Homomorphic encryption is malleable, researches have shown that fully deterministic homomorphic encryption scheme E can be broken in sub-exponential time on normal computer and polynomial time only on quantum computers.

3. **Bootstrappability**

   If E has a self-referential property of being able to handle its own decryption function(augmented by a single gate), it is said to be bootstrappable. It will be shown below in later section that when E is bootstrappable, we can use it construct a fully homomorphic encryption E'.

Simplest Homomorphic Encryption Scheme

Below are the algorithms for a simple homomorphic encryption scheme E, with security parameter λ. Let $N = \lambda, P = \lambda^2, Q = \lambda^5$.

1. **KeyGen(E)**

   Select a random P bit random odd number p.

2. **Encrypt(p, m)**

   To encrypt a single bit $m \in \{0,1\}$, first select a random N bit number m' such that $m' \equiv m \ (mod\ 2)$. Now, select a random Q bit random number. The output of encrypt is given by $c \ \rightarrow m' + p.q$

3. **Decrypt(p, c)**

   Output the required message bit m as $(c \ mod \ p) mod\ 2$, where $(c \ mod \ p)$ return a value between $\left(-\frac{p}{2}, \frac{p}{2}\right)$, such that $(c - c')$ is divisible by p for some c'.

   Ciphertexts from E are near multiples of p. (As m' is a N bit number and $p.q$ is $P * Q$ bit number which is much larger than N.) Thus the value of $(c \ mod \ p)$ returns the distance of c

from the nearest multiple of p. This distance is called the **noise** associated with ciphertext c. Decryption works because the noise, m' $(= c\, mod\, p)$ has the same parity as the message bit m. Also, if the m' was allowed to be larger the large than the value of $p.q$, the noise may not be calculated correctly and thus decryption algorithm will not work.

4. **Evaluate(f, c$_1$, c$_2$, ……. c$_k$)**

Add(c$_1$, c$_2$) = $c_1 + c_2$

Sub(c$_1$, c$_2$) = $c_1 - c_2$

Mul(c$_1$, c$_2$) = $c_1 * c_2$, where the above operations are done in modulo 2 field.

But there are problems here. Let us consider the multiplication operation.

$$c = c_1 * c_2 = (m_1 + p.q_1) * (m_2 + p.q_2) = m_1 m_2 + p.q'$$

for some integer q'. But as discussed above the decryption algorithm works fine only when the noise associated is less than p. This means that $|m_1 m_2| < \frac{p}{2}$

The above same principle applies to add and sub operations also. Roughly speaking, performing add and sub operation on ciphertext c_1 and c_2 with noises k_1 and k_2 respectively, the noise in new ciphertext c is atmost max(k_1, k_2) + 1. Performing mul operations increase the nose more rapidly as the new noise in ciphertext c is given by (k_1 + k_2).

Now, as discussed earlier, any function f from the set of permitted functions can be computed from add, sub and mul operations. For this, consider the function f as a circuit C with XOR and AND gates. Let C' be the same circuit as C, but with XOR and AND gates replaced with add, sub and mul gates. Let f' be the multivariate polynomial that corresponds to C'. Output the ciphertext as

$$c \leftarrow f'(c_1, \ldots\ldots, c_k).$$

This corresponds to

$$f'(c_1, \ldots\ldots, c_k) = f'(m_1, \ldots\ldots, m_k) + p.q'$$

But as pointed out earlier, the add and mul operations can performed only when the noise of resulting function doesn't exceed p. Thus we require that $|f'(m_1, \ldots\ldots, m_k)| < \frac{p}{2}$. Thus the encryption scheme E can handle elementary symmetric polynomial of degree d in t variables, as long as $2^{Nd}.\binom{t}{d} < \frac{p}{2}$, which is roughly when $d < P(N\,.\log t)$. For the suggested parameters, $d < \frac{\lambda}{\log t}$.

It also has an advantage when the degree of function f is low, for example when doing a basic keyword search, the function can be parallelised as well leading to better performances.

Notion of Sematic Security for above Encryption scheme E

The gcd of 2 numbers can be 2 numbers can be calculated efficiently using extended-euclid algorithm. Given the 2 ciphertexts, c_1 (= m_1 + p*q_1) and c_2 (= m_2 + p*q_2), their gcd can be approximated as p. But calculated the exact value from p from the approximated gcd value is considered a "HARD" problem as no efficient attack has been found to this. Also, the approximate gcd problem has been noticed to be hard even when multiple ciphertexts are also available.

Making the Homomorphic Encryption Bootstrappable

For making the above encryption, we would require our scheme E to handle the following function, namely the decryption function and the decryption circuits augmented by one more gate, add, sub and mul. It will also require us to create a series of public and secret keys. Thus, the KeyGen algorithm has to modified too.

But while making the scheme bootstrappable it is observed that it doesn't meet our definition of fully homomorphic function as the complexity of the KeyGen will grow linearly with the maximum depth of the function we want to evaluate. This may be overcomed by the notion of **circular security** where it is considered safe to reveal the encryption of private key under its own associated public key. This allows the same private key encryption and public key to be used at all levels, meaning freedom from the depth of function of being evaluated.

Making the Decryption process faster

The decryption process is considered slow when we want to make our scheme E bootstrappable. It can be noted that decryption can be also written as following:

$$m \leftarrow (c \bmod p) \bmod 2$$

$$m \leftarrow LSB(c) \oplus LSB\left(\left\lfloor \frac{c}{p} \right\rceil\right)$$

But again calculating $\left\lfloor \frac{c}{p} \right\rceil$ is slow and the circuit is also complicated. Thus, we should modify our scheme for all the parts, KeyGen, Encrypt, Decrypt and Evaluate in order to be able to make the computation of overall decryption process faster. This will require making a sparse subset consisting of rational numbers from $[0, 2)$ such that their sum approximately the value of $1/p$ and thus the multiplication in decrypt step can be replaced by a series of additions, making it efficient. Though we would be revealing some information about the key p we chose randomly, but it can shown that it doesn't improve the situation of the adversary much as he would be required to guess a sparse subset whose sum is known, which is same as the subset-sum problem. We know that subset-sum problem is "HARD", meaning that although revealing some information about the key, the situation for adversary doesn't get better but we are able to speed up our decryption process.

Conclusion

Being able to compute on encrypted data brings about new applications that have never been imagined before. Email filtering, secure cloud computing, and software obfuscation are some of the applications that are currently in development. The field of computing on encrypted data still has many challenges.

References

- https://google.com
- https://wikipedia.com
- https://www.ripublication.com/irph/ijict_spl/ijictv4n15spl_05.pdf
- https://eprint.iacr.org/2015/1192.pdf