Applying Homomorphic Encryption in the Cloud

Jesús Antonio Soto Velázquez

Universidad Autónoma de Nuevo León

23 de septiembre de 2015

Overview

1. Introduction
   - Problem Definition
   - Motivation
   - Hyphotesis
   - Objectives
2. Background
   - Relevant Terminology
   - Abstract Algebra
   - Homomorphic Encryption
   - Services in the Cloud
3. Methodology
   - Case Study
   - Proposed Solution
4. Related Works
5. Experiments and Results
   - Setup
   - Results
6. Conclusions

Introduction

These days, it has become more common to exchange information with our peers remotely. An issue arises when the information is considered to be *sensitive*, and thus, the confidentiality of it must be protected.

Cryptography is the study of techniques that enable secret communication, such as ciphers, that is, encryption and decryption algorithms to be used on sensitive data.

The issue at hand becomes even more interesting when the safe containing the private information does not go directly to the intended party, but is rather stored somewhere in the cloud, i.e. the Internet.

Problem Definition

An approach to secure data on the cloud consists in encrypting the data before storing it; therefore, assuring its confidentiality. However, it is inconvenient to modify the data if this approach is followed.

Homomorphic encryption is an encryption technique used to modify the already encrypted data without compromising its confidentiality.

Even though there are many homomorphic encryption schemes that have been created to allow for computations on encrypted data, so far, they are not considered fast enough to build efficient secure cloud computing solutions.

Motivation

There are many areas in which homomorphic encryption could be used, such as the medical, marketing, and financial fields. Until now, there was no way to make a concrete implementation of a solution related to these areas, since available schemes were either too limited or too slow.

By showing a compelling example of how homomorphic encryption can be applied in a simple scenario, cloud service providers and developers might start considering how to apply homomorphic encryption in other ways and mediums, and thus, expanding on software that makes use of it.

Hyphotesis

Building a client-server based solution using the homomorphic encryption functionalities provided by HElib is feasible in terms of processing time.

Objectives

Develop a client-server based solution using HElib to perform homomorphic evaluations on encrypted data.

Specific Objectives:

1. Establish a client-server architecture where homomorphic encryption can be applied.
2. Identify which factors pose a challenge to deem applications of homomorphic encryption as inefficient.
3. Show the use of HElib to setup, encrypt, and decrypt in simple terms.
4. Collect performance data on the use of homomorphic encryption.

Information Security

Computer security [?]

The protection granted to an information system enforced to preserve the *integrity*, *availability*, and *confidentiality* of information system resources.

Data Confidentiality [?]

Refers to the assurance that private or confidential information is not made available or disclosed to unauthorized individuals.

Cryptography [?]

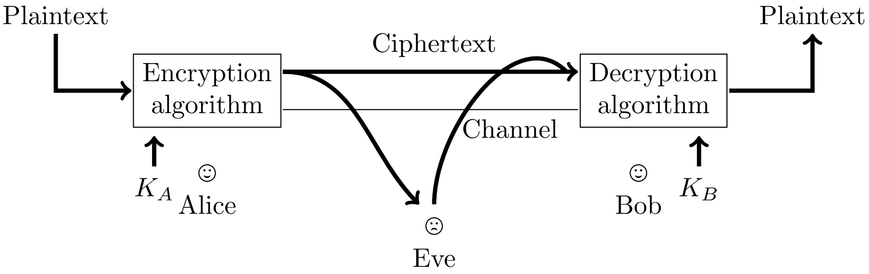
Cryptography is the study of protecting data and communications. It employs ciphers to change the appearance of messages shared between two or more parties, making it difficult for unauthorized parties to learn the content of the messages.

Definition of a Cryptosystem

Stinson [?] formally defines a cryptosystem as a quintuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, where the following conditions are satisfied:

1. $\mathcal{P}$ is a finite set of possible plaintexts;
2. $\mathcal{C}$ is a finite set of possible ciphertexts;
3. $\mathcal{K}$, the keyspace, is a set of possible keys;
4. For each $k \in \mathcal{K}$ there is an *encryption rule* $e_k \in \mathcal{E}$ and a corresponding *decryption rule* $d_k \in \mathcal{D}$. Each $e_k : \mathcal{P} \to \mathcal{C}$ and $d_k : \mathcal{C} \to \mathcal{P}$ are functions such that $d_k(e_k(x))$ for every plaintext element $x \in \mathcal{P}$.

Definition of a Cryptosystem



K_A: Alice's Key
K_B: Bob's Key

Figure 1 : Basic cryptosystem diagram. Based on Kurose and Ross [?].

Abstract Algebra

Abstract algebra is a branch of mathematics that focuses on those elements that can be operated algebraically. Groups and rings are examples of such sets of elemnts [**?**].

Definition of Group

A *group* G, denoted by $\{G, \cdot\}$ is a set of elements with a binary operation, which is denoted by the operator $\cdot$.

Definition of Ring

A *ring* R, which is sometimes denoted by $\{R, +, \times\}$, is a set of elements with two binary operations, called *addition* and *multiplication*, such that for all a, b, c in R, there are some axioms that must be obeyed.

Homomorphisms

A *homomorphism* consists of the construction of a function that *translates* elements from one group to another group with the same properties.

Definition

Let G_1 and G_2 be groups, and let $\phi : G_1 \to G_2$ be a function. Then ϕ is said to be a **group homomorphism** if

$$\phi(a * b) = \phi(a) *' \phi(b) \tag{1}$$

for all a, b in G_1.

In the case for rings, a homomorphism considers two operations instead.

Homomorphic Encryption

Refers to the ability to make computations, i.e. operations such as addition and multiplication, on encrypted data without sharing the secret key to decrypt the data prior to the computations.

When the encryption function of a cryptosystem is a homomorphism, meaning that it preserves group operations performed on ciphertexts, it is called a *homomorphic cryptosystem*.

Lange [**?**] mentions that several real life applications of homomorphic encryption fall in the domain of: e-cash, e-voting, private information retrieval, and cloud computing.

Additive and Multiplicative Homomorphic Encryption

A homomorphic encryption is additive if:

$$\mathcal{E}(x + y) = \mathcal{E}(x) \otimes \mathcal{E}(y), \tag{2}$$

where $\mathcal{E}$ denotes an encryption function, $\otimes$ denotes an operation depending on the used cipher and x and y are plaintext messages.

A homomorphic encryption is multiplicative if:

$$\mathcal{E}(x \cdot y) = \mathcal{E}(x) \otimes \mathcal{E}(y), \tag{3}$$

where again $\mathcal{E}$ denotes an encryption function, $\otimes$ denotes an operation depending on the used cipher and x and y are plaintext messages.

Classification of Homomorphic Encryption Schemes

- *Partially homomorphic encryption* schemes are defined over a group, and they support one operation at most: either addition or multiplication.
- *Fully homomorphic encryption* schemes are defined over a ring, and can support up to two operations: addition and multiplication.
- *Somewhat homomorphic encryption* schemes support up to two operations; however, the number of operations that can be performed is limited.

Jewelry Store Analogy

- Alice has a jewelry store and wants her workers to assemble raw materials (diamonds, gold, silver, etc.) into finished products (necklaces, rings, etc.).
- She does not trust her workers.
- Alice places the materials inside transparent glove boxes, which are then promptly locked with a key that only Alice has.
- Workers can put their hands inside the gloves that are connected to the box, and through them, work with the raw materials.
- The workers are able to put in things inside the boxes, such as soldering iron to use on the raw materials.
- Once the products are finished, Alice can then recover the products from the box using her key.

Services in the Cloud

The cloud, a trendy term used to describe a network of servers usually accessible through the Internet, currently provides two important services: storage and computation.

One of the main advantages of employing a service in the cloud is that the owners are relieved from the burden of data storage and maintenance at all times.

By relying on a cloud service, the owner is relieved of the control and protection of the data, raising security concerns.

Attempts to tackle these concerns include *secure multi-party computation* and *homomorphic encryption*.

Case Study

Consider a household that has a pattern of activity, i.e. empty during the day and non-empty at night, so that the resident seeks to ascertain the number of people inside at any point in time during the day.

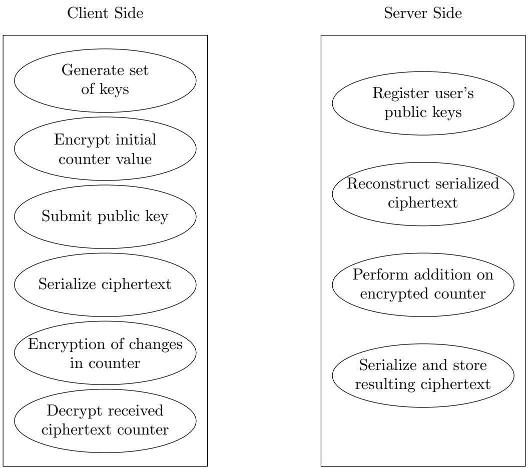
Assuming the resident has put in place certain sensors around the building so that it detects and counts who comes in and out, he would like to learn the value of the counter remotely. As the resident chooses to store the value in the cloud, he quickly realizes he does not want others to learn of this value, not even the cloud service itself, as to prevent potential burglars to break in when the household is empty.

Proposed Solution

Implementation of a client-server software that aims to store and modify securely a counter by making use of homomorphic encryption.

The solution utilizes a *HElib*, a library in C++ that enables the use of a leveled fully homomorphic encryption scheme. It is used to perform operations on the encrypted data, such as addition, subtraction, and multiplication.

Architecture Breakdown


Client Side

Generate set of keys

Encrypt initial counter value

Submit public key

Serialize ciphertext

Encryption of changes in counter

Decrypt received ciphertext counter

Server Side

Register user's public keys

Reconstruct serialized ciphertext

Perform addition on encrypted counter

Serialize and store resulting ciphertext

Figure 2 : Architecture breakdown into client and server.

Operation Flow

1. Parameters to use with the HElib are arbitrarily chosen or calculated.
2. Public and private keys are created.
3. The public key is serialized and sent to the server.
4. The structures to store plaintext and ciphertext are declared.
5. The initial counter value is set into the plaintext structure.
6. The plaintext is encrypted using the public key and stored into the ciphertext structure.
7. The encrypted counter is sent to the server.
8. Values are added or substracted from the ciphertext.
9. Ciphertext is decrypted using the private key, and its value is stored in another plaintext structure.

Sequence Diagram

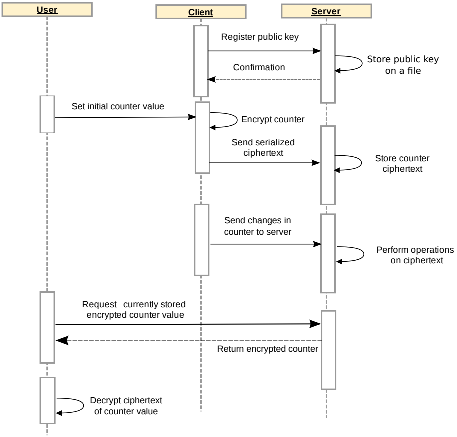

User
Client
Server
Register public key
Store public key on a file
Confirmation
Set initial counter value
Encrypt counter
Send serialized ciphertext
Store counter ciphertext
Send changes in counter to server
Perform operations on ciphertext
Request currently stored encrypted counter value
Return encrypted counter
Decrypt ciphertext of counter value

Figure 3 : Sequence diagram showing operation under normal conditions

Implementation Details

- A seed is used to pseudorandomly generate the set of keys.
- Among the parameters chosen, the security parameter k is chosen at the start.
- Serialization is done by using the *istringstream* and *ostringstream* C++ classes.
- Communication between client and server is done by using sockets on TCP.

Related Works

Table 1 : Comparison between related works

Characteristic / Work	Yasuda et al. [?]	Yasuda et al. [?]	Adida et al. [?]	Cybernetica [?]	Tetali et al. [?]	Schroepfer et al. [?]	Thesis work
Client-server architecture	×	×	✓	∼	✓	×	✓
Cloud computation	✓	×	✓	✓	✓	×	✓
Real-world application	✓	×	✓	✓	×	✓	✓
Secure computation	✓	✓	✓	✓	✓	✓	✓
Homomorphic addition support	✓	×	✓	×	×	×	✓
Homomorphic multiplication support	✓	×	×	×	×	×	✓
No 3rd party trust	×	✓	✓	✓	✓	×	✓
Key exchange support	✓	✓	✓	×	×	✓	×
Web-based	×	×	✓	✓	×	✓	×

✓ *has the characteristic*	∼ *partially has the characteristic*	× *does not have the characteristic*

Setup

The experiment is performed by running several iterations of the program with different values of the security parameter k, as to evaluate how it affects the time required for certain activities in the software, such as key generation, encryption, decryption, and homomorphic addition of ciphertext. Resulting key size and ciphertext size is also considered.

The experiment considers several arbitrarily chosen values, namely: 20, 40, 80, 100; where 80 is the default value found in the examples of the HElib.

It is important to consider that the experimentation was done on a notebook which had an AMD Elite A4-5150M processor. The processor is dual core and it runs at 2.7 GHz.

Results

Table 2 : Experimental results

k	Key Gen.	Key Size	Encryption	Decryption	Addition	Ctext Size
40	13.130 s	165.922 MB	0.355 s	0.048 s	0.074 s	30.977 kB
60	10.710 s	91.635 MB	0.414 s	0.056 s	0.077 s	36.053 kB
80	8.380 s	57.080 MB	0.642 s	0.057 s	0.082 s	37.745 kB
100	9.529 s	35.453 MB	0.579 s	0.064 s	0.082 s	43.676 kB

Results: Key Generation

Figure 4 depicts a box plot that shows the time required to generate the set of keys depending on the size of k.

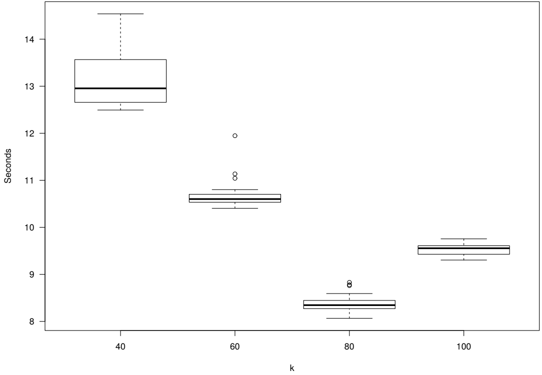


Figure 4 : Key generation time

Conclusions

In essense, a client-server archictecture was established and it made use of homomorphic encryption successfully. It was found that the factors that pose a challenge to deem applications of homomorphic encryption as inefficient were mainly: key generation time, key size, ciphertext size, and encryption and decryption times.

It was chosen to experiment on the value of the security parameter k. Data regarding the execution times and size of the key and ciphertext were collected.

The results showed that key generation took the most amount of time, ranging between 9 and 13 seconds. The size of the public key ranged between 35 and 165 megabytes. Despite the long time and large size, these were not considered to be obstacles to the feasibility of the system.

Future Work

- Design and implement the appropriate mechanisms for key management.
- Design a database scheme to keep track of the ciphertexts stored in the server.
- Look into the possibility of implementing the client-architecture system as a Software-as-a-Service model.
- Provide a secure communication channel between the client and server, relying on the SSL/TLS protocol.
- Experiment on the number of homomorphic operations correctly evaluated before the accumulated noise causes an incorrect decryption.
- Implement solution over the HTTPS protocol, as to allow its use via web browsers.