2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing

A New Lightweight Homomorphic Encryption Scheme for Mobile Cloud Computing

Mohd Rizuan Baharon, Qi Shi, and David Llewellyn-Jones
School of Computing and Mathematical Sciences
Liverpool John Moores University
Liverpool, United Kingdom
{M.R.Baharon, Q.Shi, D.Llewellyn-Jones}@2011.ljmu.ac.uk

***Abstract.* The emerging technology of mobile devices allows mobile users to access a wide range of applications through the Internet connection. As such applications demand extensive computational power, it poses a challenge to the devices with limited computation power, memory, storage and energy. However, such a challenge could be overcome by cloud computing as the cloud offers virtually unlimited dynamic resources for computation, storage and service provision. Nevertheless, mobile users are still reluctant to adopt this technology as moving private data to the cloud with no physical and limited digital control by themselves raises security concerns to their data. Encryption using the primitive encryption schemes is unsuitable for use in cloud environments, as data need to be decrypted first before they can be processed, resulting in the data being exposed to the cloud. Moreover, although homomorphic encryption is believed to be one of the potential solutions to allowing arbitrary computation on encrypted data, its efficiency is still an obstacle for its implementation. Thus, this paper will deeply look at efficiency issues and propose a new Lightweight Homomorphic Encryption (LHE) scheme which minimizes the use of computation power at encryption and key generation. The key contribution of this work is to have a lightweight scheme with improved efficiency, while enabling homomorphism under both addition and multiplication.**

Keywords-Cloud Computing; Mobile Cloud Computing; Data Storing and Processing; Homomorphic Encryption

I. INTRODUCTION

Nowadays, mobile devices like smartphones and tablets are increasingly becoming an essential part of human life as they are the most effective and convenient communication tools not bounded by time and place. Such devices are quickly raising popularities due to the support for a wide range of applications like gaming, image processing, video processing and online social network services that allow users to accumulate rich experience. Such applications include iPhone apps and Google apps, which run on the devices and/or on remote servers via wireless networks. The rapid progress of mobile computing becomes a powerful trend in the development of IT technology as well as commerce and industry fields. However, the mobile devices are facing many challenges in their resources like battery life, storage, and bandwidth. Furthermore, they are also facing communication challenges such as mobility and security [1]. Such limitations have significantly effected on the improvement of service qualities.

This work was supported in part by Liverpool John Moores University (LJMU), United Kingdom, the Ministry of Higher Education Malaysia (MOHE), Malaysia and University Technical Malaysia Melaka, Malaysia.

On the other hand, with the emerging technology of cloud computing, more and more services have been offered and delivered through the Internet. Cloud computing offers tremendous advantages by allowing users to use its infrastructure like servers, networks, and storages, platforms such as middleware services and operating systems, and software like application programs. All of those services are provided by Cloud Service Providers (CSPs) like Google, Amazon, and Salesforce at low costs. In addition, cloud computing enables users to elastically utilize resources in an on-demand fashion. As a result, mobile applications can be rapidly provisioned and released with the minimal management efforts or service providers' interactions. With the explosion of mobile applications and the support of cloud computing for a variety of services for mobile users, mobile cloud computing is introduced as an integration of cloud computing into the mobile environment. Mobile cloud computing brings new types of services and facilities for mobile users to take full advantage of cloud computing.

In order to leverage such a technology and services, mobile users need to outsource their data to the CSPs for storing and processing purpose. However, outsourcing such data, which is often private or sensitive, into the clouds with no physical and limited digital control by the users raises serious security concerns to the data [2]. Furthermore, inappropriately handling such data could result in a disaster to the data owner due to data misuse, data leakage, or data stolen by other parties that use the same services. Moreover, the CSPs do not offer proper security guarantees to the data owners [3]. Due to the scale, dynamicity, openness and resource-sharing nature of cloud computing, addressing security issues in such environments is a very challenging problem [4].

To ensure the security and integrity of the data are preserved in clouds, encryption techniques should be implemented. Primitive encryption schemes such as RSA are good for storing purposes [5]. However, such encryption techniques prevent data from being processed by cloud-based applications [6]. Thus, a scheme that allows data to be processed in an encrypted form, like a Fully Homomorphic Encryption (FHE) scheme, is extremely desired. Although a number of existing FHE schemes have been proposed and improved upon, all of them are far from practical as efficiency is still a big challenge for their implementation [7]. For instance, existing FHE schemes based on Lattices are suffering from efficiency issues due to the amount of noise introduced during the processing stage of data [8].

978-1-5090-0154-5/15 $31.00 © 2015 IEEE
DOI 10.1109/CIT/IUCC/DASC/PICOM.2015.88

Additionally, a scheme based on a bilinear map allows arbitrary additions and only one multiplication on encrypted data [9]. Furthermore, existing FHE schemes are computational expensive, which require a lot of computing resources to implement such schemes. This inhibits the mobile devices from computing in an efficient manner. A scheme that enables data to be processed using both addition and multiplication is highly desired by many applications in order to process users' data to something more significant. Thus, such limitations require an improved FHE scheme to be proposed.

To address the above need, in this paper we propose a new lightweight homomorphic encryption scheme that is constructed based on Gentry's scheme. We follow the same parameters setting as in [10], e.g. q_i's with a large size are designed to avoid a generalised version of Howgrave-Graham attack. The main difference between the two schemes is that our choice of plaintext for encryption is an integer, whereas Gentry's scheme is in the form of bits. This novel choice leads to our scheme being more efficient as the encryption on an integer is faster than encryption on every single bit [11]. Our scheme also provides an improved efficiency as the key generation and encryption require low computation. Moreover, our scheme supports homomorphism under both addition and multiplication as long as a plaintext result satisfies some conditions specified. To implement such a scheme, we also provide a protocol that allows three parties to communicate with one another in order to process data in an encrypted form. Such a protocol will ensure that the security and integrity of the outsourced and processed data in the third party environment are preserved. In this work, the performance of LHE is thoroughly analysed and evaluated with detailed simulations.

The rest of this paper is structured as follows. In Section II, we briefly review the background and some related work on secure mobile cloud computing. We then describe the application setting concepts that will be used in constructing our scheme in Section III. In Section IV, we explain our proposed scheme together with its related process. Section V describes the security analysis of the proposed scheme, while the performance analysis of our work is provided in Section VI. Finally, we conclude this paper in Section VII.

II. Background

A. Mobile Cloud Computing

People would like to work and manage their daily life and tasks regardless of locations, times and situations. This requires the people to have mobile devices that can be carried everywhere at any time. Mobile devices have rapidly emerged and become popular with improved capabilities such as computing power, battery resources, security and privacy. As a result, the devices have been extended and improved to suit recent applications provided through the Internet like cloud computing. With such improvements, people can accomplish their daily tasks like Internet banking, GPS, etc. conveniently and efficiently. Even though the capabilities on the mobile devices have improved, some applications demand extensive computing power and battery consumption. Wireless communication devices for instance appear to be highly power-consumptive.

Several techniques in [12] have been introduced to reduce the power consumption and save the energy during the communication. Furthermore, the battery life can be extended by offloading large tasks for remote processing. Work in [12] shows that the portable computers executing their large tasks remotely can save up to 51% of battery power. Dinh, H., T., et al. [1] proposed a computation offloading technique with the objective to migrate large computations and complex processing from resource-limited devices like mobile devices to resourceful machines such as servers in clouds. This avoids taking a long application execution time on mobile devices, which would result in a large amount of power consumption. To apply this offloading technique, several works have been done to evaluate the effectiveness of such a technique through some experiments. The results demonstrate that the remote application execution can save energy significantly.

On the other hand, storage capacity is also a constraint for mobile devices. Mobile cloud computing is developed to enable mobile users to store/access a large amount of data on the cloud through wireless networks. The Amazon Simple Storage is one of the examples that facilitate storage as a service. Another example is Image Exchange which utilizes the large storage space in clouds for mobile users. This mobile photo sharing service enables mobile users to upload images to the clouds immediately after capturing. Users may access all images from any devices. With the cloud, the users can save a considerable amount of energy and storage space on their mobile devices because all images are sent to, stored and processed on the clouds [1].

Moreover, in cloud computing, all services are delivered through web applications and data that has been outsourced is no longer owned by the users. Shifting all the data and computing resources to the cloud can have implications on privacy and security. Since the data is stored and managed on the cloud, security and privacy settings depend on the IT management provided by the cloud. The CSP typically works with many third party vendors. There is no guarantee how these vendors may safeguard the data. Moreover, data on the cloud may be stored at multiple locations across different states and countries. Data that might be secure in one country may not be secure in another: different jurisdictions may apply over accessing the data. All these factors make it evident that all data cannot be stored in the cloud without considering the privacy and security implications. One possible solution to storing data is to encrypt the data before storage. This can prevent unauthorized access even when the storage is breached at the cloud. If the data is encrypted, then it has to be decrypted at the CSP because of the need to perform operations on the data. On the other hand, performing encryption techniques before sending the data to the cloud requires some additional processing on the mobile system and consumes additional energy [14]. Furthermore, as mobile cloud computing is based on cloud computing, all the security issues are inherited in mobile cloud computing with the extra limitation of resource constraint mobile devices. Due to the resource limitation, the security algorithms proposed for the cloud computing environment may not work well directly on a mobile device. There is a need for a lightweight secure framework that provides security with minimum communication and processing overhead on mobile devices [15].

B. Security and Privacy in Mobile Cloud Computing

Providing strong security and high privacy means requiring more computing resources and energy consumption. Furthermore, increasing the security of data will decrease the functionality that can be executed on the data [16]. Therefore, a balance between security, functionality and energy consumption has to be the guideline in providing a scheme that can be implemented to the mobile data in an efficient manner. Due to this reason, a lot of research work has been conducted to provide security and improve privacy of outsourced data with consideration of the energy consumption and storage spaces. There are several approaches to securing the outsourced data using existing methods. The first approach is to ensure the integrity of users' data stored in cloud servers. Itani et al. [17] proposed an energy efficient framework for mobile devices to ensure the integrity of the mobile users' files/data stored on a cloud server using the concept of incremental cryptography and trusted computing. Furthermore, Jia et al. [18] introduced a secure data service that outsources data and security management to cloud in a trusted mode. The secure data services allow mobile users to outsource data and data sharing overhead to a cloud without disclosing any information about the shared data. To achieve the secure data service, the proxy re-encryption and identity based encryption are implemented. The proposed secure data service provides not only data privacy but also fine-grained access control with the minimum cost of updating access policy and communication overheads. On the other hand, Shukla et al. [19] proposed a scheme for smart phones to ensure the security and integrity of mobile users' files stored on cloud servers. An archive mechanism that integrates cloud storage, hybrid cryptography, and digital signatures has been designed to provide security requirements for data storage of mobile phones. Such a mechanism not only can avoid malicious attackers from illegal access but also can share desired data and information with targeted friends by distinct access rights.

From our point of view, the schemes in [17] and [18] are based on the trusted mode. In mobile cloud environments, such schemes are not suitable for implementation as the cloud servers are assumed to be untrusted third parties. Therefore, they are not allowed to gain any information of the processed and stored data. Furthermore, work in [19] mainly focuses on security of the stored data. In mobile cloud environments, data storage is not the only services provided to the outsourced data. Data processing is one of the main services provided by the cloud servers as they have a huge amount of computing resources. Thus, securing the processed data is also important to prevent any security and privacy breaches to the user data. Based on such factors, LHE is proposed to provide a security solution to the processed data. With an improved efficiency, the LHE is believed to be the best scheme that enables data to be processed by third parties like cloud in an efficient and secure manner.

III. Application Settings

The LHE is proposed to enable mobile data to be outsourced and processed in cloud environments. However, data outsourcing itself is not the best approach to overcome the limitation of the mobile devices as the security and privacy of the data are highly important. Furthermore, heavy computation for securing the data on mobile devices before outsourcing degrades their battery lifetime. Thus, the security and computation complexity need to be balanced in order to provide a better scheme for the outsourcing data. In this section, we describe the application settings for implementing such a scheme.

To process data in a ciphertext form, we assume that there are a group of data distributors, a data client and a cloud server as detailed below:

- Data Distributors u_i:
  They are a group of people who are carrying mobile devices like smart phones or tablets. All devices can be connected to the Internet through wireless connection. The group members contribute their data to the data client and all the data has to be processed by the cloud server.
- Data Client (DC):
  It is a third party organisation that requires information from the data distributors in relation to specific tasks and purposes. It has low computing resources and storage spaces. It leverages the technology provided by the cloud to compute and store data purposely.
- Cloud Server (CS):
  A third party organisation which possesses a huge amount of computing power and storage space for computing and storing purposes. It is an untrusted party. CS provides a lot of Internet based applications and delivers as a service to the client through Internet connection. The client just needs to pay on a per use basis without any hassle to manage the software licence, maintenance, etc.

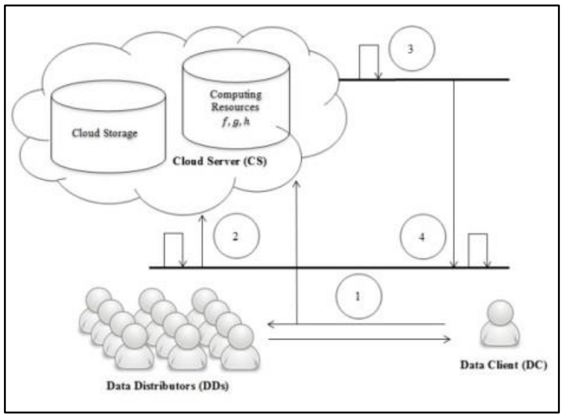

Figure 1. The protocol of implementing LHE scheme in cloud

For application setting purposes, we consider a public health scenario where a group of data distributors u_i in the park share their personal health information like heart rates, blood pressures and weights with a local hospital (DC) through a cloud server (CS) to get some statistical results based on the provided data. Such application setting can be illustrated in Fig. 1. Individuals may be unwilling to disclose their personal data to CS and DC if there were no guarantee that their data would not be used to invade their privacy. Thus, to allow such a scenario to be implemented in securely, we propose the following protocol as illustrated in Fig. 1. The protocol involves four phases, which are described below:

Phase 1: DC and data contributor group

- DC broadcasts a request to the group to request specific data to be processed by a CS application.
- Each group member u_i receives the request through Wi-Fi connection.
- u_i in the possession of the requested data responds to DC to confirm their willingness for the data provision.
- DC generates a set of private keys based on the responded group members and sends the keys to them.

Phase 2: Data contributor group and CS

- Each participating member u_i randomises its data to avoid data guessing by the other parties.
- u_i encrypts the randomised data.
- u_i sends the encrypted data to CS for processing.

Phase 3: CS

- CS receives the ciphertext data from every participating u_i.
- CS processes the received data using a set of applications (f, g, h) requested by DC.

Phase 4: CS and DC

- CS sends the processed result to DC.
- DC decrypts the result using its master key.

VI. The Proposed LHE Scheme

The proposed LHE scheme consists of four algorithms, which are key generation, data encryption, data recovery and data evaluation. All algorithms are described below.
Suppose that:

- There are n mobile devices participating as data distributors, each of which is denoted as u_i $(1 \leq i \leq n)$.
- The maximum length of data items to be computed is l_d bits.
- CS is employed to compute the received data from the n mobile devices.
- A function f is defined as summation, a function g is defined as multiplication and a function h is defined as the combination of both summation and multiplication.
- A client-server structure is used to represent the devices as nodes and their connections as links.
- Every connected device and CS have already authenticated each other and established a secure communication channel between them if necessary.

A. Key Generation

The proposed Lightweight Homomorphic Encryption (LHE) scheme employs a private key for data encryption by each contributor u_i. The private key is shared between its associated DC and u_i, and used for symmetric data encryption.
To produce this key, we adopt the parameter delineations for the verifiable encryption of RSA signatures [20]. That is, DC defines φ as the product of two safe primes p and w, i.e. $\varphi = pw$ where $p = 2p' + 1$ and $w = 2w' + 1$ with p' and w' being primes. Also, Q_φ is used to denote the subgroup of squares in $\mathbb{Z}_\varphi^*$ (i.e. the multiplicative group modulo φ), and a number $\acute{g} \in \mathbb{Z}_\varphi^*$ is randomly picked up to yield $g = \acute{g}^2 \bmod \varphi$, i.e. $g \in Q_\varphi$. φ and g will be used as public numbers, w needs to be discarded without disclosing it to anyone, and p should be kept securely.
Additionally, DC selects a prime r $(< p)$ and stores both p and r as its secret master keys. To generate keys for each u_i, DC picks up random numbers s_i and q_i to produce the following keys:

$$k_i = (rs_i + pq_i) \bmod \varphi.$$

k_i is u_i's private key. The n private keys k_i need to meet the following conditions:

(a) For the summation,
$2^l n < r$ and $rf(\bar{c}_i, s_i) < p$

(b) For the product,
$(2^l)^n < r$ and $rh(\bar{c}_i, e_i, r, s_i,) < p$.

where l is the maximal bit length of the data to be encrypted e_i and $\bar{c}_i$ is a random number chosen by an encryptor. The detailed reasons for the above conditions will be discussed later when the proposed data encryption and decryption are presented. In brief, the first part of both conditions says that the sum or the product of encrypted data items is less than r for the purposes of ensuring the recovery of the result. The second part of both conditions means that the first part of u_i's private key is multiplied by a summation or a combination of summation and multiplication of some values is below p. This condition also allows the summation or product result to be recovered.

B. Data Encryption

Similar to the work in [21], every encrypted data goes through a two-stage process: data randomisation and data encryption. Before encrypting the data, each u_i first needs to expand its data d_i to introduce sufficient randomness. Note that the data range in many mobile applications is usually limited. For instance, in a traffic monitoring application, the driving speed is between 0 to 120 km/hour. Thus, if u_i directly submits the encrypted d_i to CS without randomisation, then CS can deduce d_i by exhaustive search. To avoid this situation, each u_i expands d_i by adding a random number. In particular, assume that each data is of l bits. u_i generates a random number b_i of ϕ bits known only to itself and computes as below:

$$e_i = 2^{l_d + \lceil log_2 n \rceil} \cdot b_i + d_i \, .$$

Here, l_d is the maximal bit length of d_i, and b_i is a random number picked up by u_i, of which the bit size l_b is determined in terms of the difficulty of exhaustive search for decrypting an encrypted number. The maximal bit size of e_i is denoted as l, i.e. $e_i < 2^l$. In summation for instance, this means that the sum of all the n randomised values e_i meets condition (a) which has been defined earlier, i.e. $\sum_{i=1}^{n} e_i < 2^l n < r$.
We now present how u_i generates its encrypted data to be sent to CS, which is different from the work in [21]. u_i first does the following calculations:

$$\alpha_i = (e_i + \bar{c}_i k_i) \bmod \varphi.$$

Here, α_i is the encrypted data of e_i, and $\bar{c}_i$ is a random number picked up by u_i and will be further explained in the security flaw section. Briefly, the reason to include $\bar{c}_i$ in the encryption procedure is to enhance the security of the plaintext data. This is because the plaintext size is too small if compared to the symmetric key. Based on this reason, some information about the key can be retrieved from the encrypted data. Thus, by including such a random number $\bar{c}$ during the encryption, it will hide the information about the symmetric key as the encrypted data will be totally different from the key for encryption.

After the completion of the above calculation, u_i sends α_i to CS for storing and computing purposes. Upon the receipt of α_i from u_i, CS starts to do the computation on the ciphertexts received to generate a result based on functions f, g or h requested by DC.

C. Data Recovery

DC decrypts α_i with master keys p and r to recover the value e_i by:

$$e_i = (\alpha_i \bmod p) \bmod r.$$

This is due to the following relationships:

$$\begin{aligned} e_i &= (\alpha_i \bmod p) \bmod r \\ &= \Big(\big((e_i + \bar{c}_i k_i) \bmod \varphi\big)\, mod\ p\Big)\, mod\ r \\ &= \Bigg(\bigg(\Big(e_i + \bar{c}_i\big((rs_i + pq_i)\big)\Big) \bmod pw\bigg)\, mod\ p\Bigg)\, mod\ r \\ &= \big((e_i + \bar{c}_i rs_i + p\omega)\, mod\ p\big)\, mod\ r \\ &= (e_i + \bar{c}_i rs_i)\, mod\ r \quad \text{as } r\bar{c}_i s_i < p \text{ with } \omega < w \\ &= e_i \quad \text{as } e_i < r \end{aligned}$$

After the successful decryption, DC recovers the plaintext by calculating the following result:

$$d_i = e_i \bmod 2^{l_d + \lceil log_2 n \rceil}.$$

The above equation is true based on the following reason:

$$\begin{aligned} d_i &= e_i \bmod 2^{l_d + \lceil log_2 n \rceil}. \\ &= \left(2^{\tau + \lceil log_2 n \rceil} b_i + d_i\right) \bmod 2^{l_d + \lceil log_2 n \rceil} \\ &= d_i \quad \text{as } d_i < \bmod 2^{l_d + \lceil log_2 n \rceil} \end{aligned}$$

D. Data Evaluation

In this sub-section, we describe how CS computes the received data using addition and multiplication without the need for decryption. Once the computation on ciphertext completed, the encrypted result will be sent to DC for decryption to recover the plaintext result. The following are the steps of addition and multiplication together with the reason of getting a correct result of computing ciphertext data. Furthermore, in this sub-section we show how the scheme supports homomorphism under addition and multiplication. The homomorphism can be defined as follows:

Definition 1: A function $f: G \to H$ from one group G to another group H is a (group) homomorphism if the group operation is preserved in the sense that

$$f(g_1 *_G g_2) = f(g_1) *_H f(g_2)$$

1) Summation

Suppose $f(e_1, e_2, \dots e_n) = \sum_{i=1}^{n} e_i$, i.e. the summation of e_i for $1 \le i \le n$. Then, the summation on the ciphertext is defined as below:

$$f(\alpha_1, \alpha_2, \dots, \alpha_n) = \textstyle\sum_{i=1}^{n} \alpha_i\ mod\ \varphi$$

The randomised result can be obtained by the following computation:

$$\textstyle\sum_{i=1}^{n} e_i = (f(\alpha_1, \alpha_2, \dots, \alpha_n)\ mod\ p)\ mod\ r$$

This is because of the following relationships:

$$\begin{aligned} &\textstyle\sum_{i=1}^{n} e_i = (f(\alpha_1, \alpha_2, \dots, \alpha_n)\ mod\ p)\ mod\ r \\ &= (((\textstyle\sum_{i=1}^{n} \alpha_i)\ mod\ \varphi)\ mod\ p)\ mod\ r \\ &= \Big(\big(\textstyle\sum_{i=1}^{n}\big((e_i + \bar{c}_i k_i) \bmod \varphi\big)\big) mod\ p\Big)\ mod\ r \\ &= \Big(\big((\textstyle\sum_{i=1}^{n} e_i + \sum_{i=1}^{n} \bar{c}_i k_i)\ mod\ \varphi\big) mod\ p\Big)\ mod\ r \\ &= \\ &\Big(\big((\textstyle\sum_{i=1}^{n} e_i + \sum_{i=1}^{n} \bar{c}_i(rs_i + pq_i))\ mod\ pw\big)\ mod\ p\Big)\ mod\ r \\ &= \big((\textstyle\sum_{i=1}^{n} e_i + \sum_{i=1}^{n} \bar{c}_i rs_i + p\omega')\ mod\ p\big) mod\ r \\ &= (\textstyle\sum_{i=1}^{n} e_i + \sum_{i=1}^{n} \bar{c}_i rs_i)\ mod\ r \\ &= \textstyle\sum_{i=1}^{n} e_i. \end{aligned}$$

Here, we have $(\sum_{i=1}^{n} \bar{c}_i pq_i) \bmod pw = p\omega'$ with $\omega' < w$. Also the above fact is based on the following connections derived from conditions (a) $2^l n < r$ and (b) $rf(\bar{c}_i, s_i) < p$:

$\sum_{i=1}^{n} e_i < 2^l n < r$, $\sum_{i=1}^{n} e_i + r\sum_{i=1}^{n} \bar{c}_i s_i < rf(\bar{c}_i, s_i) < p$, and $\sum_{i=1}^{n} e_i + r\sum_{i=1}^{n} \bar{c}_i s_i + p\omega' < pw$.

To obtain the plaintext result, the computation is as follows:

$$\begin{aligned} \textstyle\sum_{i=1}^{n} d_i &= (\textstyle\sum_{i=1}^{n} e_i) \bmod 2^{l_d + \lceil log_2 n \rceil} \\ &= (\textstyle\sum_{i=1}^{n}(2^{l_d + \lceil log_2 n \rceil} b_i + d_i)) \bmod 2^{l_d + \lceil log_2 n \rceil} \\ &= \textstyle\sum_{i=1}^{n} d_i \quad \text{as } \sum_{i=1}^{n} d_i < 2^{l_d + \lceil log_2 n \rceil}. \end{aligned}$$

2) Product

Suppose $g(e_1, e_2, \dots e_n) = \prod_{i=1}^{n} e_i$, i.e. the product of e_i for $1 \le i \le n$. Then, the product on the ciphertext is defined as below:

$$g(\alpha_1, \alpha_2, \dots, \alpha_n) = \textstyle\prod_{i=1}^{n} \alpha_i\ mod\ \varphi$$

The product of randomised data can be obtained by the following computation:

$$\textstyle\prod_{i=1}^{n} e_i = (g(\alpha_1, \alpha_2, \dots, \alpha_n)\ mod\ p)\ mod\ r$$

This is because of the following relationships:

$$\begin{aligned} &\textstyle\prod_{i=1}^{n} e_i = (g(\alpha_1, \alpha_2, \dots, \alpha_n)\ mod\ p)\ mod\ r \\ &= \Big(\big((\textstyle\prod_{i=1}^{n} \alpha_i)\ mod\ \varphi\big)\ mod\ p\Big)\ mod\ r \\ &= \big(((\textstyle\prod_{i=1}^{n}(e_i + \bar{c}_i k_i)) \bmod \varphi)\ mod\ p\big)\ mod\ r \\ &= \Big(\big(((\textstyle\prod_{i=1}^{n} e_i) + r\bar{x} + p\bar{y})\ mod\ \varphi\big)\ mod\ p\Big)\ mod\ r \\ &= \big(((\textstyle\prod_{i=1}^{n} e_i) + r\bar{x} + p\omega")\ mod\ p\big) mod\ r \\ &= ((\textstyle\prod_{i=1}^{n} e_i) + r\bar{x})\ mod\ r \\ &= \textstyle\prod_{i=1}^{n} e_i. \end{aligned}$$

Here, we have $\bar{x} = h(\bar{c_i}, e_i, r, s_i)$ and $\bar{y} = h(\bar{c_i}, e_i, p, q_i, r, s_i)$. We also have $(p\bar{y}) \bmod pw = p\omega"$ with $\omega" < w$. Also the above fact is based on the following connections derived from conditions $(2^l)^n < r$ and $rh(\bar{c_i}, e_i, r, s_i) < p$. To obtain the plaintext result, the computation is as follows:

$$\begin{aligned} \textstyle\prod_{i=1}^{n} d_i &= (\textstyle\prod_{i=1}^{n} e_i) \bmod 2^{l_d + \lceil log_2 n \rceil} \\ &= (\textstyle\prod_{i=1}^{n}(2^{l_d + \lceil log_2 n \rceil} b_i + d_i)) \bmod 2^{l_d + \lceil log_2 n \rceil} \\ &= \textstyle\prod_{i=1}^{n} d_i \quad \text{as } \prod_{i=1}^{n} d_i < 2^{l_d + \lceil log_2 n \rceil} \end{aligned}$$

Based on Definition 1, the scheme is said to be homomorphic under addition and multiplication operations as long as both of the stated conditions i.e. (a) and (b) are satisfied.

V. Security Analysis

In this section, we provide an analysis on securing the scheme by considering several attacks to show that our scheme is secure enough against such attacks. Due to the space limit, we have only selected two types of attack on the keys, brute force attack on the master key and brute force attack on the symmetric key.

A. Brute Force Attack on the Master Key, (p, r)

By using our scheme to encrypt the data, this attack on the ciphertext can be formulised as follows. DC sends a request to the contributor group. Suppose several attackers are members of the group. The attackers pretend to have the data related to the request and will let DC knows about it. As DC is not able to differentiate the attackers from other genuine contributors, it will send a symmetric key to each attacker for data encryption. Having received the keys from DC, the attackers start the computation of deducing DC's master keys p and r. They subtract one key from another, hoping to remove r. If r were removed, the attackers could compute the Greatest Common Divisor (GCD) of the reminder with the public value φ. By doing this, master key p would be discovered by the attackers, which could then be used to work out r in a similar way.

We now present the above attack in detail. Having received the set of symmetric keys from DC, the attackers start the computation of recovering the master keys as follows:

Suppose that A_1 and A_2 are attackers with received keys k'_1 and k'_2 in the following form:

$k'_1 = (r + pq_1)\ mod\ \varphi$

$k'_2 = (r + pq_2)\ mod\ \varphi$

A_1 and A_2 then subtract both keys:

$\bar{k} = k'_1 - k'_2 = \left((r + pq_1) - (r + pq_2)\right) mod\ \varphi$

$= \left(p(q_1 - q_2)\right) mod\ \varphi$

To determine the master key p, they compute the GCD of the following values:

GCD $\left(\bar{k}, \varphi\right) = p$

This is due to the following relationships:

GCD $\left(\bar{k}, \varphi\right) =$ GCD $(p(q_1 - q_2), pw) = p$

Furthermore, the attacker can retrieve the value of r by computing $k'_1 mod\ p = r$.

After the completion of the above steps, the attackers obtain the master keys (p, r) generated by DC. If they can intercept the communication between another group member u_i and CS or CS and DC, then they can gain the information of the encrypted data as they have the decryption key.

To prevent such an attack, we attach a random parameter s_i to r in the definition of our keys k_i to avoid the above elimination of r when the attackers subtract two distinct keys they received from DC. This can be seen by repeating the elimination process as follows:

$k_1 = (rs_1 + pq_1)\ mod\ \varphi$

$k_2 = (rs_2 + pq_2)\ mod\ \varphi$

If A_1 and A_2 subtract both keys:

$\bar{k}' = k_1 - k_2 = \left((rs_1 + pq_1) - (rs_2 + pq_2)\right) mod\ \varphi$

$= \left(r(s_1 - s_2) + p(q_1 - q_2)\right) mod\ \varphi$

Now master key p cannot be retrieved by computing GCD $\left(\bar{k}', \varphi\right)$ due to $\bar{k}'$ being no longer a multiple of p, i.e.

GCD $\left(\bar{k}', \varphi\right) =$ GCD $(r(s_1 - s_1) + p(q_1 - q_2), pw) \neq p$

However, according to [22], such improvement allows known attacks like brute-forcing the remainders on the approximate-GCD problem. Thus, we review such an attack for two keys k_1 and k_2.

A simple brute-force attack is try to guess $\check{r}_1 = rs_1$ and $\check{r}_2 = rs_2$ and verify the guess with a GCD computation. Specifically, for two correctly guessed values $\check{r}'_1$ and $\check{r}'_2$ of $\check{r}_1$ and $\check{r}_2$ respectively (i.e. $\check{r}'_1 = \check{r}_1$ and $\check{r}'_2 = \check{r}_2$), compute:

$k''_1 = (k_1 - \check{r}'_1)\ mod\ \varphi$

$k''_2 = (k_2 - \check{r}'_2)\ mod\ \varphi$

Therefore, p can be computed by:

GCD $\left(k''_1, k''_2\right) = p$

This is true based on the following relationships:

GCD $(k'_1, k'_2) =$ GCD $\left((k_1 - \check{r}'_1), (k_2 - \check{r}'_2)\right)$

$=$ GCD $\left(\left((\check{r}_1 + pq_1) - \check{r}'_1\right), \left((\check{r}_2 + pq_2) - \check{r}'_2\right)\right)$

$=$ GCD (pq_1, pq_2)

$= p$

Therefore, to avoid the brute-force attack on the remainder, the length of $\check{r}_1$ (or $\check{r}_2$) should be large enough but must be smaller than p to allow the recovery of the plaintext data during the decryption.

B. Brute Force Attack on the Symmetric Key, k_i

To prevent the scheme from brute force attack on a symmetric key, a random parameter $\bar{c}_i$ is added to the ciphertext. Such a parameter can improve the security of the encrypted data by avoiding any information about the encryption key being disclosed to unauthorised users. This means that based on given public parameters such as the ciphertext data and φ, an attacker, which has some knowledge about the plaintext, is unable to retrieve useful information for successfully guessing an encryption key.

We now argue the above claim in detail. Suppose that the encryption algorithm for plaintext e_i with symmetric key k_i is:

$\alpha'_i = (e_i + k_i)\ mod\ \varphi.$

Such an encryption algorithm is said to be unsecured against a brute-force attack on k_i based on the following reason.

As mentioned earlier, $\bar{c}_i$ is added in our encryption algorithm for security purposes. The main reason is to hide the key information from the attacker and even a curious CS. For recovering purposes, the key should be large, i.e.

$log_2(f(d_i)) \ll log_2(k_i)$ for every i

where, f is the summation of d_i and k_i is the symmetric key for encrypting d_i.

Since $k_i = (rs_i + pq_i)\ mod\ \varphi$, we have:

$log_2(k_i) = log_2(\varphi).$

Furthermore, as $\alpha'_i = (e_i + k_i)\ mod\ \varphi$, this leads to:

$log_2(\alpha_i) = log_2(k_i) = log_2(\varphi).$

Based on the above relationships, we can see that when the plaintext is very small compared to the key used for encryption, the size of the ciphertext generated is almost identical to the key size. This means that for different encryptions with the same key, the difference among them is just the bits at the lower end of the ciphertexts, while the rest remains the same, which is the higher end of the key. In case the attacker is able to obtain the ciphertexts, it can compare them to spot their identical part so as to gain that part of the key. If the remaining part of the key is short, then the attacker can guess it by a brute force attack.

Such a weakness can be avoided by adding a random parameter $\bar{c}_i$ to the encryption as devised in our algorithm:

$$\alpha_i = (e_i + \bar{c}_i k_i) \bmod \varphi$$

Here $\bar{c}_i$ is only known by the encryptor. By adding such a parameter, each ciphertext will be a different number which does not directly leakage any information about the symmetric key.

VI. Performance Evaluation

In this section, we evaluate the total execution time of the LHE scheme and a chosen scheme for a comparison. These results demonstrate the importance of a lightweight homomorphic encryption scheme in terms of its efficiency.

A. Experimental Setup

In our total execution time tests, we evaluate the total execution time of the two schemes (the LHE and the compared scheme) for several phases like data randomisation and data encryption. The details of the compared scheme can be found in [23]. We implemented the two schemes with various numbers of data distributors. The evaluation is implemented using Matlab software.

B. Parameters Setting

In this experiment, we use the parameter settings defined in TABLE 1. Note that although the table only includes the three entities (i.e. DC, u_i and CS), the 'Not Know' column also covers any other entities apart from the two empty entries.

C. Experimental Results

The experimental results are shown in TABLE 2 and Fig. 2 – 5 for the LHE and the compared schemes.

Our first experiment is about the total execution time for data randomisation, encryption and decryption. As expected, the LHE scheme is faster than the compared scheme as the computation complexity of the compared scheme is higher than that of the LHE scheme. The result of the comparison is shown in TABLE 2.

TABLE 1. The Parameter Settings and Security Requirements

Para.	**Descriptions**	**Length in bits**	**Who should**	
			Know	**Not**
p	A part of master key	512	DC	u_i, CS
r	A part of master key	256	DC	u_i, CS
w	A value chosen by DC	512	DC	u_i, CS
d_i	A plaintext	10	u_i	CS, DC
e_i	A randomised plaintext	177	u_i	CS, DC
b_i	A random integer	160	u_i	CS, DC
$\bar{c}_i$	A random integer	184	u_i	CS, DC
s_i	A random integer	54	DC	u_i , CS
q_i	A random integer	1024	DC	u_i , CS
φ	A public number	1024	All	
k_i	A symmetric key	1024	u_i, DC	CS
α_i	A ciphertext	1024	All	

TABLE 2: Total Delay for Data Randomisation, Encryption and Decryption

Tasks	**Delay in Seconds**	
	The LHE Scheme	**The Compared Scheme**
Data Randomisation	0.1751	0.1751
Data Encryption	0.1053	205.8911
Data Decryption	0.3158	144.866

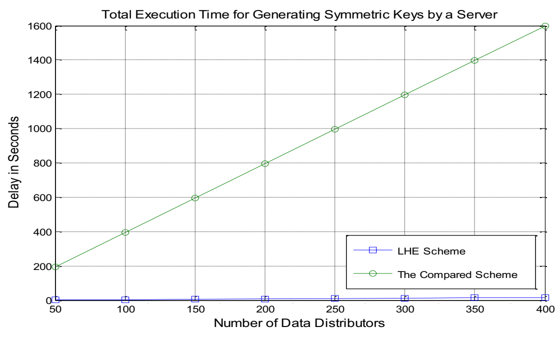

Figure 2. Total execution time for generating symmetric keys by a DC

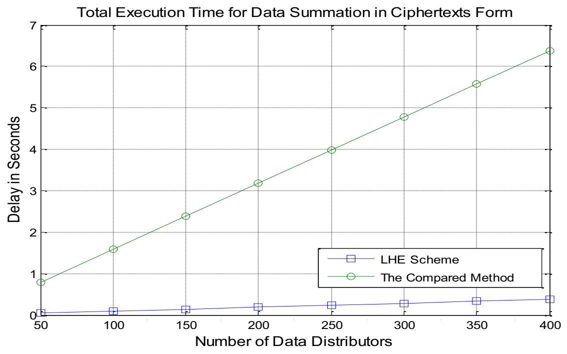

Figure 3. Total execution time for summation in the ciphertext form by CS

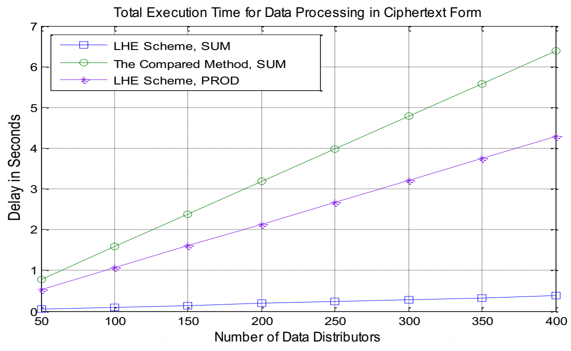

Figure 4. Total execution time for data summation and product in the ciphertext form by CS

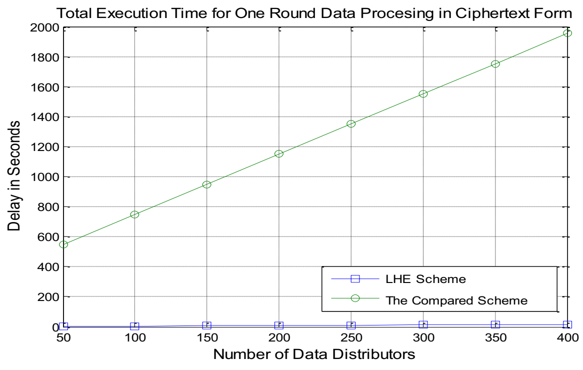

Figure 5. Total execution time for one round data processing (based on the designed protocol in Fig. 1) in the ciphertext form

Furthermore, we evaluated the total execution time of key generation for both of the schemes with different numbers of contributors. The result is given in Fig. 2, showing that the delay caused by using the LHE scheme is slightly increased in response to the increase of the number of contributors. In contrast, the delay rapidly increases using the compared scheme in the same circumstances. Moreover, we can see that there is a significant difference between these two schemes for generating keys. For 50 contributors, the key generation by LHE takes less than one second, whereas the other scheme takes much longer time, which is 200 seconds. Moreover, the graph also shows that as the number of contributors increases, the delay of generating the keys constantly increases with 200 seconds difference between the two schemes.

Fig. 3 provides the delay of data summation on the ciphertexts using the two different encryption schemes. The graph shows the delay produced using LHE gradually increases as the number of contributors increases. However, for the compared scheme, the delay rapidly increases in relation to the increase of contributors. For example, when the number of contributors is 400, the delay caused by LHE is below 0.5 second. For the compared scheme, the process takes longer time, which is more than 6 seconds.

In Fig. 4, we can see the delay of product on ciphertext data encrypted using LHE. It gradually rises according to the rise of the number of contributors, but it is still acceptable and below the delay produced by the compared scheme for ciphertext summation. For 400 contributors, the product of ciphertexts using LHE takes about 4 seconds, which is below the delay of data summation using the compared scheme that takes more than 6 seconds.

Fig. 5 shows the delay of the two schemes in one round of data summation over ciphertext data. The one round process is a process based on the designed protocol depicted in Fig. 1, where three parties communicate with one another to process data in the encrypted form. The graph obviously demonstrates the significant difference of the delay between the two schemes. For instance, for 400 contributors, the process takes almost 2000 seconds using the compared scheme to complete the process, while LHE only takes less than 10 seconds. This result shows that LHE enables data processing in the ciphertext form much more efficiently.

VII. Conclusion and Future Work

In this paper, we have proposed the new Lightweight Homomorphic Encryption (LHE) scheme that allows mobile users to outsource their data in a secure and privacy-preserved manner. Moreover, the scheme enables ciphertext data to be processed under both addition and multiplication operations without decryption. We have compared LHE with another related scheme through the experiments that have been developed using Matlab software. Our effort is mainly focused on the evaluation of the total execution time of the two schemes by providing comprehensive comparisons between them. The results show that LHE can operate faster than the compared scheme as it has less complexity in terms of computation. Furthermore, we have also provided a security analysis of LHE to show that although the scheme has less complexity, it can provide strong security to the outsourced data. Such a result has proved that our scheme can be implemented to provide strong security to protect mobile data.

References

[1] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A Survey of Mobile Cloud Computing : Architecture , Applications , and Approaches," *Wireless Communications and Mobile Computing*, no. 13, pp. 1587–1611, 2013.

[2] S. Singh and I. Chana, "Cloud Based Development Issues : A Methodical Analysis," *International Journal of Cloud Computing and Services Science*, vol. 2, no. 1, pp. 73–84, 2013.

[3] K. Nahrstedt, R. Campbell, E. Burger, J. Giffin, X. H. Gu, A. D. Joseph, E. Keller, D. Ma, and H. Weatherspoon, "Security for Cloud Computing," in *A Report: Directorate for Computer and Information Science and Engineering (CISE)*, pp. 1–19, 2012.

[4] D. Zissis and D. Lekkas, "Addressing cloud computing security issues," *Future Generation Computer Systems*, vol. 28, no. 3, pp. 583–592, Mar. 2012.

[5] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 1–11, Jan. 2011.

[6] Z. Mahmood, "Data Location and Security Issues in Cloud Computing," *2011 International Conference on Emerging Intelligent Data and Web Technologies*, pp. 49–54, Sep. 2011

[7] C. Gentry and N. P. Smart, "Homomorphic Evaluation of the AES Circuit," *Adv. Cryptol. - CRYPTO 2012. Lecture Notes Computer Science*, vol. 7417, pp. 850–867, 2012.

[8] J. Fan and F. Vercauteren, "Somewhat Practical Fully Homomorphic Encryption," *IACR Cryptology ePrint Archive*, 2012.

[9] D. Boneh, "Evaluating 2-DNF Formulas on Ciphertexts," in *Second Theory of Cryptography Conference, TCC 2005, Cambridge Proceedings*, 2005, pp. 325–341.

[10] M. Tibouchi, "Scale-Invariant Fully Homomorphic Encryption over the Integers," *Cryptology ePrint Archive 2014/032*, pp. 1–18, 2014.

[11] H. Zhou and G. Wornell, "Efficient homomorphic encryption on integer vectors and its applications," *2014 Information Theory Application Workshop ITA 2014 - Conference Proceeding*, 2014.

[12] R. P. Rudenko A., "Saving Portable Computer Battery Power through Remote Process Execution," *Mobile Computing and Communication Review*, vol. 2, no I, pp. 19–26, 1998.

[13] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generation Computer System*, vol. 29, no. 1, pp. 84–106, 2013.

[14] K. Kumar and Y.-H. Lu, "Cloud Computing for Mobile Users: Can Offloading Save Energy?," *IEEE Computer*, pp. 51–56, 2010.

[15] A. N. Khan, M. L. Mat Kiah, S. U. Khan, and S. a. Madani, "Towards secure mobile cloud computing: A survey," *Future Generation Computer System*, vol. 29, no. 5, pp. 1278–1299, 2012.

[16] A. Boldyreva, G. Tech, P. Grubbs, and S. Networks, "Making encryption work in the cloud," *Network Security*, vol. 2014, no. 10, pp. 8–10, 2014.

[17] W. Itani, A. Kayssi, and A. Chehab, "Energy-efficient incremental integrity for securing storage in mobile cloud computing," *2010 International Conference on Energy Aware Computing, ICEAC 2010*, pp. 26–27, 2010.

[18] W. Jia, H. Zhu, Z. Cao, L. Wei, and X. Lin, "SDSM : A Secure Data Service Mechanism in Mobile Cloud Computing," *The First International Workshop on Security in Computers, Networking and Communications*, pp. 1060–1065, 2011.

[19] S. C. Hsueh, J. Y. Lin, and M. Y. Lin, "Secure cloud storage for convenient data archive of smart phones," *Proceedings of the International Symposium on Consumer Electronics, ISCE*, vol. 18, no. 51, pp. 156–161, 2011.

[20] G. Ateniese, "Verifiable encryption of digital signatures and applications," *ACM Transactions on Information and System Security*, vol. 7, no. 1, pp. 1–20, Feb. 2004.

[21] R. Zhang, J. Shi, Y. Zhang, and C. Zhang, "Verifiable Privacy-Preserving Aggregation in People-Centric Urban Sensing Systems," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 9, pp. 268–278, 2013.

[22] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully Homomorphic Encryption over the Integers." IACR Cryptology ePrint Archive, pp. 1-28, 2010

[23] M. R. Baharon, Q. Shi, D. Llewellyn-Jones, and M. Merabti, "Secure rendering process in cloud computing," *2013 11th Annual Conference on Privacy, Security and Trust, PST 2013*, pp. 82–87, 2013.