Summer 7-20-2017

Privacy-preserving Data clustering in Cloud Computing based on Fully Homomorphic Encryption

Abdulatif Alabdulatif
RMIT university, Abdulatif.alabdulatif@rmit.edu.au

Ibrahim Khalil
RMIT University, Melbourne, ibrahim.khalil@rmit.edu.au

Mark Reynolds
Bosten university, markreyn@bu.edu

Heshan Kumarage
RMIT university, heshandk@gmail.com

Xun Yi
RMIT University, Melbourne, xun.yi@rmit.edu.au

Privacy-preserving Data clustering in Cloud Computing based on Fully Homomorphic Encryption

Completed Research Paper

Abdulatif Alabdulatif
Computer Science &
Software Engineering,
School of Science, RMIT university
Abdulatif.alabdulatif@rmit.edu.au

Ibrahim Khalil
Computer Science &
Software Engineering,
School of Science, RMIT university
Ibrahim.khalil@rmit.edu.au

Mark Reynolds
Computer Science Department
Bosten university
markreyn@bu.edu

Heshan Kumarage
Computer Science &
Software Engineering,
School of Science, RMIT university
heshandk@gmail.com

Xun Yi
Computer Science &
Software Engineering,
School of Science, RMIT university
xun.yi@rmit.edu.au

Abstract

Cloud infrastructure with its massive storage and computing power is an ideal platform to perform large scale data analysis tasks to extract knowledge and support decision-making. However, there are critical data privacy and security issues associated with this platform, as the data is stored in a public infrastructure. Recently, fully homomorphic data encryption has been proposed as a solution due to its capabilities in performing computations over encrypted data. However, it is demonstrably slow for practical data mining applications. *To address this and related concerns, we introduce a fully homomorphic and distributed data processing framework that utilizes MapReduce to perform distributed computations for data clustering tasks on a large number of cloud Virtual Machines (VMs). We illustrate how a variety of fully homomorphic-based computations can be carried out to accomplish data clustering tasks independently in the cloud and show that the distributed execution of data clustering tasks based on MapReduce can significantly reduce the execution time overhead caused by fully homomorphic computations. To evaluate our framework, we performed experiments using electricity consumption measurement data on the Google cloud platform with 100 VMs. We found the proposed distributed data processing framework to be highly efficient when compared to a centralized approach and as accurate as a plaintext implementation.*

Keywords: Privacy-preserving data clustering, Fully homomorphic encryption, Cloud-based framework

Introduction

Data analytics has proven its potential to extract knowledge and support decision-making. However, the continuous increase in data generated by enterprises, as evidenced by the rise of social media and the Internet of Things (IoT), has produced an overwhelming data flow that, in turn, raises critical issues concerning the capabilities of traditional analytics tools (Labrinidis and Jagadish 2012), (Cuzzocrea et al. 2011). The cloud platform has been introduced as a convenient platform with on demand unlimited 'on demand' access to resources and location independence. Figure 1 illustrates the applications domains that can benefit from delegating their services to cloud computing. Nevertheless, perspective users (e.g. individuals or organizations) have major concerns about the privacy of data because it is stored in public cloud infrastructure that is managed by the cloud service provider (CSP) (Subashini and Kavitha 2011). Accordingly, several data privacy issue can arise if the CSP misuses data or if outside malicious attackers gain unauthorized access to data stored on the cloud (Subashini and Kavitha 2011).

Preservation of the privacy of data mining algorithms has been an area studied extensively over the last decade. For example, Agarwal and Srikant (Agrawal and Srikant 2000) and Lindell and Pinkas (Lindell and Pinkas 2000) introduced privacy preserving models to extract useful information from private data. Since that time, several privacy-preserving data mining approaches have been introduced, including (Vaidya and Clifton 2003), (Bunn and Ostrovsky 2007). The existing models are either randomization-based models, as in (Agrawal and Srikant 2000), (Kargupta et al. 2003), or cryptographic-based models, as in (Doganay et al. 2008), (Inan et al. 2007). Although randomization-based models are efficient, they do not rely on a well-defined security mechanism and, because of this, add a large amount of noise causing the data to no longer represent the original values (Pedersen et al. 2007). On the another hand, although the cryptographic-based models preserve a high level of privacy, they suffer in terms of high computational cost and communication overheads (Pedersen et al. 2007). Therefore, overhead reduction is a focal issue in cryptography-based research.

The state-of-the-art cryptographic privacy-preserving data mining approach is categorized according to three main techniques: oblivious transfer, secret sharing and Homomorphic Encryption (HE). HE techniques have the ability to carry out computations on encrypted domains, such as the Pallier Homomorphic cryptosystem in (Paillier 1999) . However, Pillier and colleagues used a HE cryptosystem that is classified under Partially Homomorphic Encryption (PHE), where a limited number of computations are carried out and which lacks arithmetic operations variety. These drawbacks limit the usability the PHE cryptosystems in data mining applications. In short, the critical issues of current cryptography-based models can be summarized as follows: 1) The majority of the existing models rely on Trusted Third Party (TTP), as in (Inan et al. 2007), which creates critical privacy and security issues (Benjamin et al. 2010); 2) The secret sharing-based models have a limited scope for improvement in terms of communication overhead reduction, which would reduce the scaling capability of a number of parties in a distributed scenario (Doganay et al. 2008); and 3) The existing HE-based models adapt traditional public key cryptography techniques. However, the PHE cryptosystems are lacking in terms of computations capabilities and limited to certain number of computations. These incur high computation and communication overheads because they require many encryption/decryption operations for verification and computational processes(Jha et al. 2005). To address these concerns, in this paper we propose a privacy-preserving data clustering framework based on a fully homomorphic cryptographic technique with the following objectives: 1) Avoid using TTP, as this can give rise to several critical privacy and security issues; 2) Avoid intensive communication between various parties, as this can lead to information leakage during communication; 3) Avoid computation overheads resulting from a large number of encryption/decryption operations during data analysis, for encryption/decryption operations during data analysis, for either verification or computations purposes. These objectives can be achieved by adapting Fully Homomorphic Encryption (FHE), as described in (Gentry 2009).

With FHE, mathematical computations can be carried out in an encrypted domain. However, although FHE such as the Gentry (Gentry 2009) and van Dijk (Van Dijk et al. 2010) schemes, are considered to be impractical compared with PHE (Naehrig et al. 2011), they can perform an unlimited number of both addition and multiplication operations. PHE has the ability to perform limited additions or multiplications, but not both, such as the RSA (Rivest et al. 1978) and Paillier (Paillier 1999) approaches. FHE underwent several optimization stages intended to improve its performance efficiency. In 2012, Gentry et al. (Gentry et al. 2012) implemented an efficient encryption procedure of

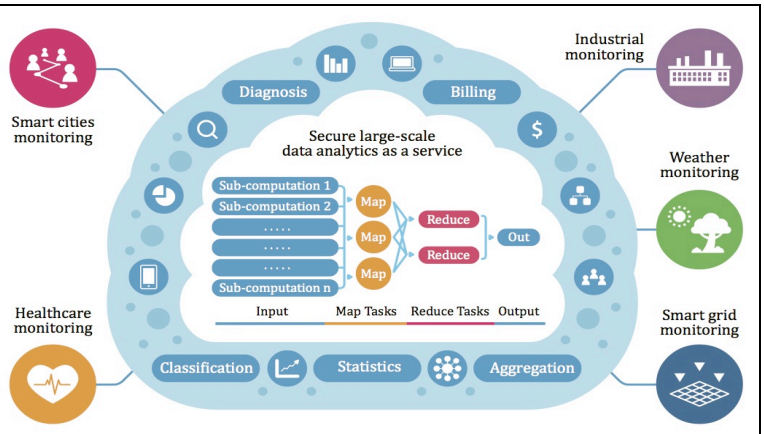

Figure 1. An overview of applications domains that can benefit from our privacy-preserving data clustering framework in cloud computing. Data owners can delegate their data analytic services to the cloud computing securely.

the Advanced Encryption Standard (AES) cipher. After several further optimizations to improve the evaluation time, the Brakerski-Gentry-Vaikuntanathan (BVG) scheme (Brakerski and Vaikuntanathan 2011) made it possible to perform an AES evaluation in less than three hours. Despite these optimizations, it is still far from ready to be applied in practical, real world situations.

In this paper, we investigate the possibilities of adapting FHE for privacy-preserving data analytics in cloud computing. Distributed FHE computations offer a viable approach to overcome the huge overhead incurred by FHE computation and allows us to build scalable and efficient data analytics models (Hayward and Chiang 2015). To build this privacy-preserving data analysis model, we used the HElib library (Halevi and Shoup 2014), which is written using C++ and based on the Number Theory Library (NTL). However, the HElib scheme has shortcomings in its ability to perform data analysis tasks, such as the ability to perform computations on floating-point numbers, which is essential in most data mining applications. Moreover, it also does not provide functionality for performing fundamental data analysis operations, such as division and comparison. In this paper, we provide a convenient solution to solve these shortcomings. The IEEE standard for floating-point arithmetic (IEEE 754) is utilized to convert floating-point numbers in their integer representation, which makes it suitable for processing in the HElib. Moreover, we built secure division and comparison functions on top of the HElib library to enable automation of the entire data analysis process. The MapReduce functionality was adapted to distribute the analysis computations to significantly reduce the FHE computation time overhead. For the sake of simplicity, we used Fixed-Width Clustering (FWC) as a possible data clustering method, as it can be deployed in various existing applications.

Contribution

The major contributions of this paper are as follows:

- The design and implementation of a distributed data clustering approach using fully homomorphic encryption.
- The distribution of encrypted computations amongst a large number of virtual machines (VMs) to perform data analysis tasks in a matter of minutes, significantly reducing the computation time overhead of a fully homomorphic approach as in BGV scheme .It is demonstrated that the proposed framework can perform analysis tasks on encrypted data completely in cloud, without the need to interact with other entities (e.g. data owners or TTP).
- The development and introduction of functions to perform comparison and division operations into the BGV scheme, with capability to perform operations on both integer and floating-point representations. We show that fully homomorphic floating- point arithmetic computations can be carried out using the IEEE standard for floating-point arithmetic (IEEE 754).
- Comprehensive experimental evaluations that compare the efficiency of the proposed

approach to that of a centralized data processing. We show that the proposed models work to significantly improve the computational performance efficiency for clustering tasks.

Related work

Privacy-preserving data mining approaches are classified into two main categories: randomization and cryptographic techniques. The latter includes secret sharing, oblivious transfer and HE. In the randomization approach, Agarwal and Srikant (Agrawal and Srikant 2000) proposed a randomization scheme which added a random number only to the sensitive value of an attribute. They claim that it is possible to reconstruct the distribution of the original data for a given distribution of random noise. Moreover, Kargupta et al. (Kargupta et al. 2003) introduced a random matrix based on a spectral filtering technique that has the ability to recover the original data from the perturbed data. Other randomization techniques have been proposed in (Oliveira and Zaiane 2010) and (Gurevich and Gudes 2006). Although compared to the cryptographic techniques the randomization approach is an efficient and simple technique that can be easily implemented, a great amount of information is lost during the transformation processes (Shanthi and Karthikeyan 2012). Moreover, the randomization approach does not provide a formal way of indicating how much security and privacy is guaranteed. Therefore, it is not a convenient approach for applications that require high security, privacy and accurate results (Pedersen et al. 2007).

Cryptographic approaches are classified as either Secure Multi-party Computation (SMC) or HE. SMC was originally introduced by Yao (Yao 1982) and allows different parties to jointly collaborate to compute a specific function on their private data, while preserving the data privacy of each party. Secret sharing (Doganay et al. 2008) and oblivious transfer (Inan et al. 2007), (Vaidya and Clifton 2003) are two of the main protocols of SMC and most SMC applications are based on semi-honest models (Panackal and Pillai 2013). The SMC applications based on malicious adversary models are too complex to be suitable for data mining application in their current form (Panackal and Pillai 2013). HE has the ability to carry out computations on encrypted domains and has two main categories: PHE, as in (Rivest et al. 1978), (Paillier 1999), and FHE schemes, as in (Gentry 2009), (Brakerski and Vaikuntanathan 2011). Despite that fact that PHE schemes are efficient, they are impractical because they lack the arithmetic operations capabilities required for data analysis applications (Aguilar-Melchor et al. 2013). They also cause a huge communication overhead as both the sender and the receiver are required to transmit data several times to each other to perform even simple addition and multiplication operations. On the other hand, FHE schemes require the encrypted data to be transmitted only once. However, due to the large computational overhead, most of them are still far from being ready to use in practical applications.

Data mining applications require a very large number of arithmetic operations, but FHE schemes can be made suitable if the required operations are executed in parallel or in a distributed environment to achieve near real-time performance. MapReduce on a cloud platform is considered to be the most viable option to achieve this objective. However, the absence of cryptographic security would be seen as a serious weakness in such applications. Since data analysis is already computationally heavy and HE adds an enormous computational burden to the existing woes, we took advantage of both to build secure, but scalable, distributed data mining algorithms.

Preliminaries

The Brakerski-Gentry-Vaikuntanathan (BGV) fully homomorphic encryption scheme

The BGV (Brakerski et al. 2014)is an asymmetric encryption scheme, the security of which is linked to the difficulty of ring-learning with errors (RLWE) problem. The BGV scheme is a leveled fully homomorphic scheme that eliminates the expensive re-encryption operation. Therefore, it can evaluate all circuits homomorphically up to a predefined depth without bootstrapping. The BGV operates over a polynomial ring $\mathbb{A} = \mathbb{Z}[X]/F(X)$ where $F(X)$ is a cyclotomic polynomial. The ciphertext space is polynomials over $\mathbb{A}_q = \mathbb{A}/q\mathbb{A}$ where q is an integer modulus, the plaintext space is a ring $\mathbb{A}_p = \mathbb{A}/p\mathbb{A}$ some integers ($p \ll q$) and $gcd(p, q) = 1$. The basic BGV operations can be distributed as follows:

- $(pk, sk) \leftarrow BGV.KeyGen(p, \lambda, \ell)$: The key generation algorithm has three inputs: a prime number (p) that defines the plaintext space, a security parameter (λ), and the depth level of the evaluated arithmetic circuit (ℓ). The outputs are a secret key (sk) and a corresponding public key (pk).
- $(c) \leftarrow BGV.Encyption(sk, m)$: The encryption algorithm has two inputs: the public key (pk) and the plaintext message $m \in \mathbb{A}p$. The output is a ciphtertext $c \in \mathbb{A}q$ that is homomorphically encrypted.
- $(m) \leftarrow BGV.Decryption(pk, c)$: The decryption algorithm has two inputs: the secret key (sk) and the ciphertext message $c \in \mathbb{A}q$. The output is a plaintext $m \in \mathbb{A}p$.

HElib library

We built our privacy-preserving data clustering framework on the HElib library that is an open source implementation of the BGV HE scheme and includes many optimizations to speed up homomorphic evaluation. The focus is on the effective use of the Smart-Vercauteren ciphertext packing techniques and the Gentry-Halevi-Smart optimizations. The HElib library does not support arithmetic computations on float-point numbers, which is an obstacle because most applications in data mining domain require computations on floating-point numbers. Therefore, we contributed to improving the HElib library by providing a practical solution to perform arithmetic computations on floating-point numbers by utilizing an IEEE standard for floating-point arithmetic (IEEE 754). IEEE 754 has the ability to transform a floating-point number to a set of integer values that can, in turn, be used in the HElib library. The process can reverse a set of encrypted integer values to determine: the floating-point value. We expand on the IEEE 754 representation and arithmetic operations below.

IEEE standard for floating-point arithmetic (IEEE 754)

IEEE 754 is a representation for floating-point computation. In this paper, IEEE 754 is used to perform floating-point arithmetic by converting floating-point numbers to a set of integers that in turn become compatible with the HElib library. The IEEE 754 representation of a floating point number F which in Figure 2 has 32 bits (and can be extended to 64 bits) is arranged as follows:

- S is the sign bit (a 1-bit field).
- E is the exponent field (an 8-bit field).
- M is the mantissa field (a 23-bit field).

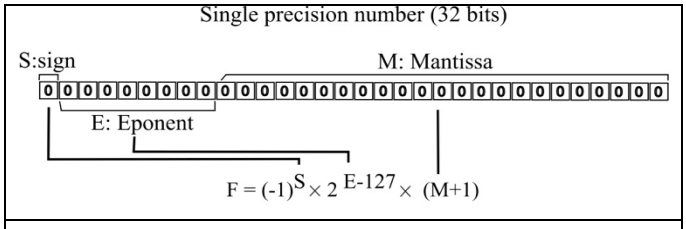

Figure 2. The IEEE 754 (32 bits) representation for a floating-point number.

Some implementations store a hidden 24th mantissa bit to help with rounding calculations. The sign bit S is 0 for a positive number and 1 for a negative number. The exponent field E is the actual exponent $+ 127$, so E should be treated as an unsigned value in the range $[0,255]$. The mantissa field represents a number in the range [1.0,2.0) except that the leading 1 is not encoded in M. Therefore, F can be mathematically expressed as follows:

$$F = (-1)^S \times 2^{(E-127)} (1 + M) \quad (1)$$

The integer values S, E and M can be used to perform arithmetic operations with other floating-point numbers that have the same representation in the HElib library.

Privacy-preserving distributed data clustering framework

In this section, we describe the privacy-preserving distributed data clustering framework. We illustrate how clustering computations can be carried out based on FHE. We also show FHE

computations based on a distributed approach. We used a Fixed-width clustering (FWC) algorithm as a case study to demonstrate how our framework can be adapted to preserve the privacy of data clustering in cloud computing. Therefor, we design a mathematical model based on IEEE 754 standard to be able carry out FHE computations for both integer and floating-point numbers.

FHE floating-point operations

Most of data clustering applications require to carry out computations based on floating-point representation which make exists FHE schemes are inefficient in this case because it lacks the ability to represent floating-point numbers. . The IEEE standard representation for a floating-point number F occupies 32 bits. These bits are arranged as follows: $S < 31 > E < 30{:}23 > M < 22{:}0 >$. S is an integer in the range $[0,1]$, E is an integer in the range $[0255]$ and M is a binary fraction that represents a floating point number in the range $[1,2)$. Henceforth, we will use M to denote the fractional part of the mantissa, so the actual mantissa is $M' = 1 + M$. The basic operations involved in data clustering are addition, subtraction, multiplication, division, and comparison. We show how these operations can be carried based on BGV scheme as follows:

- **Addition and subtraction operations**
  In the case of addition and subtraction operations involving two floating-point numbers, simple arithmetic shows that any expression of the form $F_1 + F_2$ or $F_1 - F_2$ can be written in the form $S_1\ (F_1\ \pm\ S_2\ (F_2))$, where S_1 and S_2 are sign bits. The first step in FHE addition and subtraction is to rewrite it in its canonical form.

$$F_1 \pm F_2 = ((-1)^{S_1} \times 2^{E_1} \times (1 + M_1)) \pm ((-1)^{S_2} \times 2^{E_2} \times (1 + M_2)) \quad (2)$$

  The second step is to find canonical shared components and carry out the computations of these components based on a default integer representation of the BGV scheme.
- **Multiplication operation**
  A multiplication operation of two floating-point numbers F_1 and F_2 requires determining canonical shared components as follows:

$$\begin{aligned} F_1 \times F_2 &= ((-1)^{S_1} \times 2^{E1} \times (1 + M_1)) \times ((-1)^{S_2} \times 2^{E2} \times (1 + M_2)) \\ &= ((-1)^{S_1+S_2} \times 2^{E1+E_2} \times (1 + M_1 + M_2 + (M_1 \times M_2))) \end{aligned} \quad (3)$$

  We can compute the product of $F_1 \times F_2$ in an encrypted fashion through three quantities: $S_{out} = S_1 + S_2$, $E_{out} = E_1 + E_2$, and $M_{out} = M_1 + M_2 + (M_1 \times M_2)$, which can be carried out in the BGV scheme.
- **Division operation**
  The division operation F_1 / F_2 can be expressed as in Equation 4, where $F_2 \neq 0$.

$$\begin{aligned} F_1/F_2 &= (-1)^{S_1-S_2} \times 2^{E1-E_2} \times (1 + M_1) / (1 \times M_2) \\ &= ((-1)^{S_1-S_2} \times 2^{E1-E_2} \times (1 + M_1) \times G \\ &= H \times G \end{aligned} \quad (4)$$

  Where $G = 1/(1 + M_2)$ and $H = (-1)^{S_1-S_2} \times 2^{E1-E_2} \times (1 + M_1)$. We already know how to compute H in encrypted form and therefore only need to develop a method to compute G. Because $|M_2| < 1$, we can use the Taylor expansion to evaluate $1 / (1 + M_2)$ and then take the product of that result with H to get the division result.
- **Comparison operation**
  The comparison operation can be carried by take the advantage of shift function in the HElib library. The library has the ability to test if an encrypted context Ctxt is zero, so we only need the $<$ and $>$ operators. Observe that if $(a < b)$ then $(a - b)$ is negative, and if $(a > b)$ then $(b - a)$ is negative, so we only need to be able to compute the sign function $signum\ (Ctxt)$ of an encrypted context. In unencrypted 32-bit arithmetic $-1 = 0xFFFFFFFF$, so if $(\gg)$ represents arithmetic shift right, then we have that $(-1) \gg 1 = (-1)$. Any unencrypted negative number will always have the sign bit set, so will always be of the form $-1 = 0x80000000 + X$, where X is positive in the range $0X7FFFFFFF$. However, for a floating point number there are at most 23 bits of precision. Observe that for a negative number F there must be some n, $0 < n < 23$ such that $(F \gg n) = (F \gg (n + 1))$, with F nonzero. For a

positive number F this is not the case. Thus we can compute $signum(F)$, where F is an encrypted quantity, by producing all the shifted values and testing for the equality of any two adjacent terms. If equality is found then F is negative; otherwise it is zero or positive.

FHE data clustering based on MapReduce

The FHE computations time overhead is an obstacle to practical implementation of large-scale data processing based on fully homomorphic encryption. Therefor, distributed fully homomorphic-based computations based on MapReduce can overcome this issue by parallelize processing tasks amongst a large number of virtual machines. Figure 3 illustrates the overview of the privacy-preserving distributed data clustering framework. The MapReduce is a programming model for processing large-scale data. Users can distribute computations by specify them in terms of a map and a reduce functions and the computations are automatically parallelized across large number of virtual machines. In this paper, we show how iterative distance-based computations can be parallelize based on MapReduce. We use FWC algorithm as a case study of how its computations can accomplished independently in cloud without interaction with any external parties such as data owner or TTP.

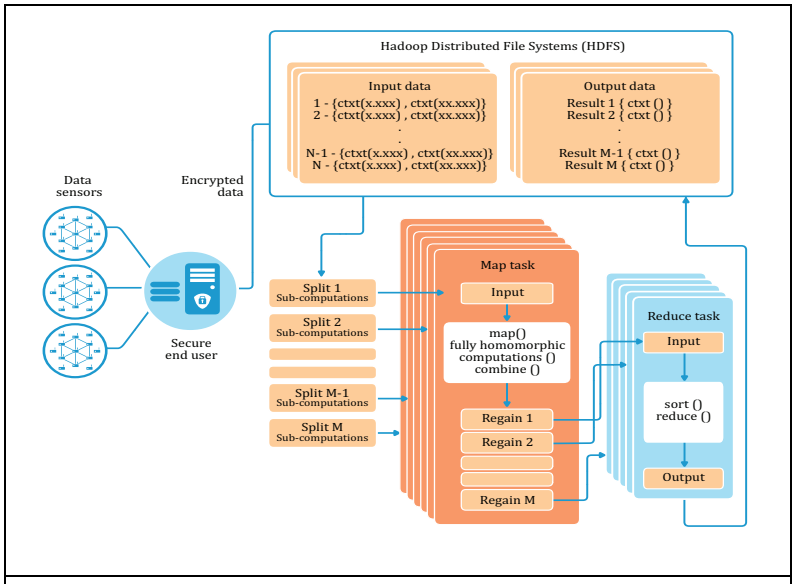

Figure 3. An overview of privacy-preserving distributed data clustering framework.

Fixed-width clustering (FWC) algorithm

Fixed-width clustering is an unsupervised clustering algorithm that works based on a distance measure. We show how the FWC algorithm works in the following steps:

- Choose a random set of clusters C_j where $j = 1\ to\ m$ from a given dataset D with a predetermined cluster width w.
- For each data point p_i where $i = 1\ to\ n$, calculate the Euclidean distance d_{ij} between p_i and each cluster C_j, as in Equation 5.

$$d_{ij}(p_i, c_j) = \sqrt{(p_{ix} - c_{jx})^2 + (p_{iy} - c_{jy})^2} \quad (5)$$

- If d_{ij} to the closest cluster centroid C_j from the current data point is $\leq w$, then p_i is added to C_j and the centroid of cluster C_j is adjusted to be the mean of the data points it now contains. For example, for n points in a cluster, the centroid is calculated as follows:

$$centroid(c_x, c_y) = ((p_{1x}+\ldots+p_{nx}/n), (p_{1y}+\ldots+p_{ny}/n)) \quad (6)$$

- If d_{ij} to the closest cluster centroid C_j from the current data point is $> w$, then p_i is considered as a centroid of a new cluster C_j.
- Repeat steps 2, 3, and 4 until the end of the dataset.

Distributed FWC using MapReduce

The FWC algorithm based MapReduce works by distributing arithmetic computations that can be parallelizes though a large number of VMs. The MapReduce framework that is implemented in Hadoop can handle the FWC algorithm through its map and reduce functions, as follows:

- **Inputs:** The MapReduce framework receives two inputs that are a dataset D and initial clusters c_1, c_2,..., c_n.
- **Data partitioning:** The dataset D is partitioned among a set of mappers as follows. We allocate the N data points equally among the M VMs available to perform MapReduce operations. If the result (i.e. $\frac{M}{N}$) is not an integer value, we add the remaining data points to the last VM. Expressed mathematically, the number of data points per VM equals $\left\lfloor \frac{N}{M} \right\rfloor$ for all VMs $1,2, \dots, (M-1)$, and the M^{th} one would receive $\left\lfloor \frac{N}{M} \right\rfloor + (N \bmod M)$ data points. We illustrate a generic example of how data is partitioned among a distributed set of Mappers in Figure 4.

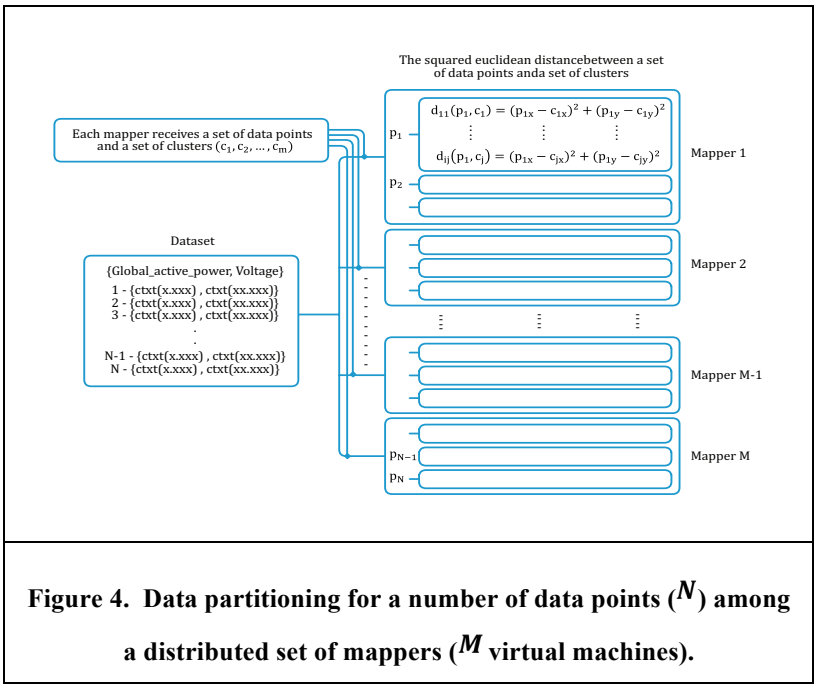

Figure 4. Data partitioning for a number of data points (N) among a distributed set of mappers (M virtual machines).

- **Map function:** The input encrypted dataset is stored on the Hadoop Distributed File System (HDFS) as a sequence of $< key , Ctxt(value) >$ pairs. The key is the index that refers to the position of a data point in a data file and the $value$ is the actual encrypted numerical value of the data point. The data file is globally shared and broadcast to all mappers. In our framework, the mappers calculate the encrypted Euclidean distances d_{ij} between each data point and a set of clusters where each mapper handles computations for Euclidean distances $E(d_{ij})$ through the ($computeDist( )$) function between a set of data points and a set of clusters. We use squared Euclidean distance rather than the standard Euclidean distance to avoid a square root operation as shown in Equation 7.

$$E(d_{ij}(p_i , c_j)) = \{(E(p_{ix}) - E(c_{jx}))^2 + (E(p_{iy}) - E(c_{jy}))^2\} \quad (7)$$

The outputs of the map function are $< key , Ctxt(value) >$ where the key is an index and the $value$ is a distance $E(d_{ij})$ between a data point $E(p_{ij})$ and a cluster $E(c_j)$. Algorithm 1 shows the pseudo-code of map function.

- **Reduce function:** The input data for a reducer is a set of $< key , Ctxt(value) >$ that is the output of the mappers. The reducer has two main tasks:

1. Find a minimum distance between each data point $E(p_i)$ and a set of the centroids of a set of clusters c_1, c_2,..., c_n through the (min ( )) function.

2. Assign each data point $E(p_i)$ to a specific cluster \$ $E(c_j)$ through the ($assign()$) function.

Algorithm 1 Map function

Data: Encrypted dataset $E(D)$
Result: $<key, ctxt(value)> \rightarrow$ <index , encrypted distance $E(d_{ij})$>
Initialisation : Choose a random set of clusters $c_1, c_2, \cdots, c_m$ from a given dataset $E(D)$
$index = 0$
for $(i = 0$ *to* $D.length)$ **do**
  for $(j = 0$ *to* $c.length)$ **do**
    $E(d_{ij}) = computeDist(E(p_i), E(c_j);$
    $index = i + j;$
  end
end
End
Take index as key
Construct value as an encrypted numerical value $E(d_{ij})$
output $< key, ctxt(value) >$ pair

In a case where a data point $E(p_i)$ is assigned to a cluster $E(c_j)$, we update the cluster centroid $E(c_j)$ through the ($update()$) function by calculating the mean of the data points it now contains, otherwise, we create a new cluster for the data point $E(p_i)$ through the ($createNewCluster()$) function. We illustrate the basic pseudo-code of reduce function in Algorithm 2.

Algorithm 2 Reduce function

Data: <index , encrypted distance $E(d_{ij})$>
Result: $< ctxt(key), ctxt(value)> \rightarrow < E(c_j), E(p_i) >$
$Initialisation : E(minDis)$
for $(i = 0$ *to* $D.length)$ **do**
  $E(minDis) = min(d_{i1}, ..., dij)$
  if $E(minDis) \leq w$ **then**
    $assign(E(p_i), E(c_j))$
    $update(E(c_j))$
  else
    $createNewCluster(E(p_i))$
  end
end
Take $E(p_i)$ as key
Construct value as a numerical value $(E(c_j))$
output $< ctxt(key)$, $ctxt(value) >$ pair

Security Assumptions

The security assumptions of our framework are based on the security specifications of the BVG scheme for overall security. If the BGV scheme is semantically secure (Coron et al. 2011), then the arithmetic computations with HElib performed for our data clustering framework are also secure. The secure client is the only application that has access to the raw data. It generates a public and private key pair (P_k, S_k) and makes the public key available to the non-secure server threads/nodes. When the secure client transfers data to the server, it transfers it in the form of $(pcode)$, followed by the names of files containing encrypted data. Thus, for example, an operation to compute the distance between a centroid (c_x, c_y) and a point (p_x, p_y) would be transmitted as $(DIST\ c_xfile\ \ c_yfile\ \ p_xfile\ \ p_yfile)$. If we assume that the security guarantees of the BGV scheme hold, then it is computationally intractable to decrypt the contents of any of the files and gain access to the plaintext data. Furthermore, even if the results of a computation are exposed there is no way to recover the original values. It is obvious that a computation such as $a + b = c$, given c, cannot be inverted to recover either a or b. The most that an attacker can learn by observing the traffic between the client and the server is the number of values that are being processed. This information has no practical value in terms of learning anything about the data points themselves. Therefore, if we assume that the BGV and its implemented HElib are secure, and then our implementation of floating point operations using the HElib library is also secure.

Experiments and results

We used the Google cloud platform to build and evaluate our framework. The secure data clustering framework based on MapReduce is designed to use up to 100 VMs in various experimental setups. Each VM has a standard 4 CPU, each with 2.6 GHz Intel Xeon E5 with 3.75 GB. To understand how performance varies in a distributed environment, the experiments deployed different settings by varying dataset sizes and the number of VMs used to run the MapReduce tasks.

Dataset

To evaluate the adapted FWC algorithm with MapReduce on the cloud platform, we used a dataset from the UC Irvine Machine Learning repository of individual household electricity consumption measurements with 10,080 data points. It represents measurements of electric power consumption in a household with one minute sampling rate over a period of almost four years. The preservation of privacy of such data is essential because electricity usage patterns can reveal sensitive behavioral information. Service providers or malicious intruders can take advantage of the data for business purposes or to monitor people's movements and behavior. The electricity consumption measurements dataset has nine attributes (date, time, global active power, global reactive power, voltage, global intensity, sub-metering 1, sub-metering 2 and sub-metering 3). We selected two attributes, global active power (averaged active power in kilowatts per minute) and voltage, to perform data analysis to classify the electricity usage rates for different lengths of time (day, two days, four days and one week). Different clusters represent different utilization patterns.

Baseline

We evaluated our cloud-based framework by running experiments with datasets of varying sizes in different MapReduce settings (number of VMs) to explore how the number of VMs affects execution time. We built both plaintext and ciphertext versions to identify discrepancies in results that may occur due to the conversion of plaintext to the encrypted format. Moreover, we compared the efficiency of our distributed based framework with a centralized model (on a single machine).

Performance evaluation

We implemented our fixed-with clustering model with MapReduce on the Google cloud platform. We ran four datasets sizes of electricity measurements containing $1{,}440, 2{,}880, 5{,}760$ and $10{,}080$ data points that represent the electricity measurements for one, two and four days and one week respectively. Figure 5 shows detected clusters of electricity utilization patterns for different lengths of time. We achieved an equally accurate result for both plaintext and encrypted versions. Furthermore, Table 1 shows the execution time of data analysis for different datasets of varying sizes and numbers of VMs. We found that in most cases, increasing the numbers of VMs by 20 decreased the execution time by 40% to 60%. We performed another experiment to demonstrate the efficiency of our framework compared with a centralized-based model. Table 2 shows the performance in terms of execution time using the same setting as that of the previous experiment and the centralized-based model. Tables 1 and 2 show a significant gap in performance between our distributed-based model and the centralized-based model. In the centralized model, a single giant block of memory was used to hold intermediary results which slowed down computational processes. In the distributed model, each VM had its own memory block which was much smaller. Thus, the computational processes in the VMs run much faster than a simple linear scaling would indicate. For example, the distributed-based model within 20 VMs can analyze 1440 data points in just 94 seconds. On the other hand, the centralized-based model using a single VM takes 13,750 seconds, which is impractical in real world applications. It is clear that our distributed model outperforms the centralized model while ensuring the same level of accuracy of in results.

Conclusion

In this paper, we built an innovative cloud-based framework for privacy-preserving data clustering that has the ability to perform scalable and distributed data analyses in a secure manner. We implemented a distributed FWC algorithm using MapReduce to perform data analysis tasks completely on the cloud platform without the need for any TTP. This is unlike the existing approaches that require interaction between two or more parties during data analysis which leads to critical privacy and security issues. The experimental results show that our framework can ensure secure analysis of the encrypted data as accurately as plaintext data with a highly efficient performance. The distributed FWC algorithm is built based on MapReduce, which plays an important role in overcoming the computational overhead of the HElib library. This in turn leads to a significant improvement in the data analysis process. The privacy of data is protected based on the security

assumption of the BGV Fully Homomorphic scheme. The proposed framework can be adapted efficiently to several data mining applications that desire to take advantage of public cloud computing without compromising data security and privacy.

Table 1. The execution time of data clustering based on different dataset sizes and number of VMs (unit: second).

	Dataset size			
No.VMs	1440 data points	2880 data points	5760 data points	10080 data points
20	94	189	396	710
40	50	123	216	381
60	28	70	133	202
80	19	44	83	113
100	17	26	35	42

Table 2. The execution time of data analysis based on different dataset sizes in a centralized-based model (unit: second).

	Dataset size			
	1440 data points	2880 data points	5760 data points	10080 data points
Execution time	13,750	27,601	54,470	105,910

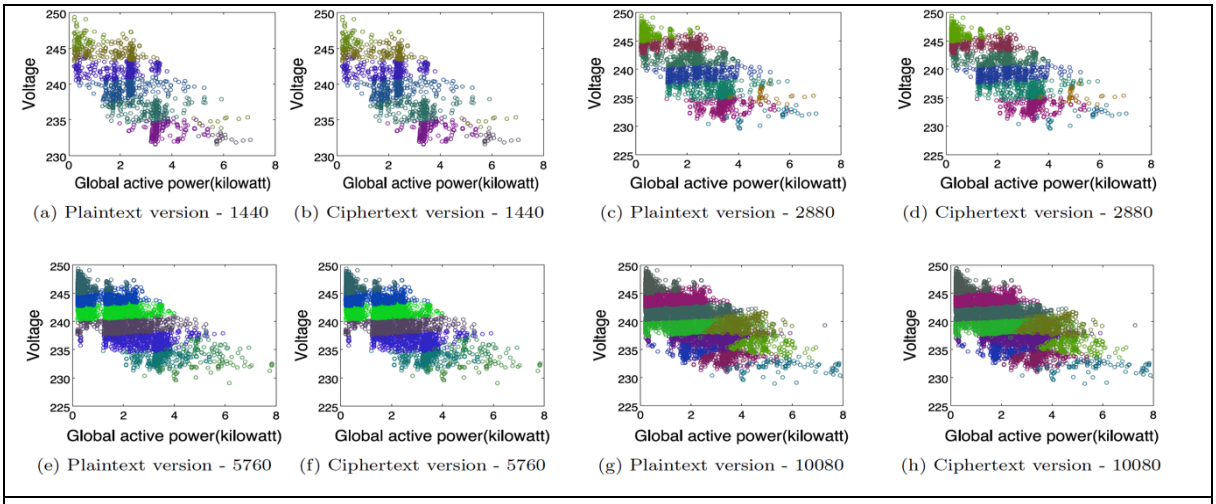

Figure 5. Detected clusters of electricity utilization patterns for different lengths of time for both plaintext ((a), (c), (e), (g)) and ciphertext ((b), (d), (f), (g)) versions and different datasets of varying sizes, 1440 data points represent a day of electricity usage (top, left) and 2880 represent two days of electricity usage (top, right), 5760 represent four days of electricity usage and 10080 data points represent one week of electricity usage (below, left) and (below, right) respectively.

References

Agrawal, R., and Srikant, R. 2000. "Privacy-Preserving Data Mining," *ACM Sigmod Record*: ACM, pp. 439-450.

Aguilar-Melchor, C., Fau, S., Fontaine, C., Gogniat, G., and Sirdey, R. 2013. "Recent Advances in Homomorphic Encryption: A Possible Future for Signal Processing in the Encrypted Domain," *IEEE Signal Processing Magazine* (30:2), pp. 108-117.

Benjamin, C., Fung, M., Wang, K., Chen, R., and Yu, P. 2010. "Privacy-Preserving Data Publishing: A Survey of Recent Developments," *ACM Computing Surveys* (42:4), pp. 141-153.

Brakerski, Z., Gentry, C., and Vaikuntanathan, V. 2014. "(Leveled) Fully Homomorphic Encryption without Bootstrapping," *ACM Transactions on Computation Theory (TOCT)* (6:3), p. 13.

Brakerski, Z., and Vaikuntanathan, V. 2011. "Fully Homomorphic Encryption from Ring-Lwe and Security for Key Dependent Messages," *Annual cryptology conference*: Springer, pp. 505-524.

Bunn, P., and Ostrovsky, R. 2007. "Secure Two-Party K-Means Clustering," *Proceedings of the 14th ACM conference on Computer and communications security*: ACM, pp. 486-497.

Coron, J.-S., Naccache, D., and Tibouchi, M. 2011. "Optimization of Fully Homomorphic Encryption," *IACR Cryptology ePrint Archive* (2011), p. 440.

Cuzzocrea, A., Song, I.-Y., and Davis, K. C. 2011. "Analytics over Large-Scale Multidimensional Data: The Big Data Revolution!," *Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP*: ACM, pp. 101-104.

Doganay, M. C., Pedersen, T. B., Saygin, Y., Savaş, E., and Levi, A. 2008. "Distributed Privacy Preserving K-Means Clustering with Additive Secret Sharing," *Proceedings of the 2008 international workshop on Privacy and anonymity in information society*: ACM, pp. 3-11.

Du, W., and Zhan, Z. 2003. "Using Randomized Response Techniques for Privacy-Preserving Data Mining," *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*: ACM, pp. 505-510.

Gentry, C. 2009. "Fully Homomorphic Encryption Using Ideal Lattices," *STOC*, pp. 169-178.

Gentry, C., Halevi, S., and Smart, N. P. 2012. "Homomorphic Evaluation of the Aes Circuit," in *Advances in Cryptology–Crypto 2012*. Springer, pp. 850-867.

Guo, L., Guo, S., and Wu, X. 2007. "Privacy Preserving Market Basket Data Analysis," *European Conference on Principles of Data Mining and Knowledge Discovery*: Springer, pp. 103-114.

Gurevich, A., and Gudes, E. 2006. "Privacy Preserving Data Mining Algorithms without the Use of Secure Computation or Perturbation," *Database Engineering and Applications Symposium, 2006. IDEAS'06. 10th International*: IEEE, pp. 121-128.

Halevi, S., and Shoup, V. 2014. "Helib-an Implementation of Homomorphic Encryption."

Hayward, R., and Chiang, C.-C. 2015. "Parallelizing Fully Homomorphic Encryption for a Cloud Environment," *Journal of applied research and technology* (13:2), pp. 245-252.

Inan, A., Kaya, S. V., Saygın, Y., Savaş, E., Hintoğlu, A. A., and Levi, A. 2007. "Privacy Preserving Clustering on Horizontally Partitioned Data," *Data & Knowledge Engineering* (63:3), pp. 646-666.

Jha, S., Kruger, L., and McDaniel, P. 2005. "Privacy Preserving Clustering," *European Symposium on Research in Computer Security*: Springer, pp. 397-417.

Kargupta, H., Datta, S., Wang, Q., and Sivakumar, K. 2003. "On the Privacy Preserving Properties of Random Data Perturbation Techniques," *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*: IEEE, pp. 99-106.

Labrinidis, A., and Jagadish, H. V. 2012. "Challenges and Opportunities with Big Data," *Proceedings of the VLDB Endowment* (5:12), pp. 2032-2033.

Lindell, Y., and Pinkas, B. 2000. "Privacy Preserving Data Mining," *Annual International Cryptology Conference*: Springer, pp. 36-54.

Naehrig, M., Lauter, K., and Vaikuntanathan, V. 2011. "Can Homomorphic Encryption Be Practical?," *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*: ACM, pp. 113-124.

Oliveira, S. R., and Zaiane, O. R. 2010. "Privacy Preserving Clustering by Data Transformation," *Journal of Information and Data Management* (1:1), p. 37.

Paillier, P. 1999. "Advances in Cryptology—Eurocrypt'99." Springer Heidelberg Berlin, Germany:.

Panackal, J. J., and Pillai, A. S. 2013. "Privacy Preserving Data Mining: An Extensive Survey," *ACEEE International Conference on Multimedia Processing, Communication and Information Technology*.

Pedersen, T. B., Saygın, Y., and Savaş, E. 2007. "Secret Charing Vs. Encryption-Based Techniques for Privacy Preserving Data Mining,").

Rivest, R. L., Shamir, A., and Adleman, L. 1978b. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM* (21:2), pp. 120-126.

Shanthi, A., and Karthikeyan, M. 2012. "A Review on Privacy Preserving Data Mining," *Computational Intelligence & Computing Research (ICCIC), 2012 IEEE International Conference on*: IEEE, pp. 1-4.

Subashini, S., and Kavitha, V. 2011. "A Survey on Security Issues in Service Delivery Models of Cloud Computing," *Journal of network and computer applications* (34:1), pp. 1-11.

Vaidya, J., and Clifton, C. 2003. "Privacy-Preserving K-Means Clustering over Vertically Partitioned Data," *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*: ACM, pp. 206-215.

Van Dijk, M., Gentry, C., Halevi, S., and Vaikuntanathan, V. 2010. "Fully Homomorphic Encryption over the Integers," *Annual International Conference on the Theory and Applications of Cryptographic Techniques*: Springer, pp. 24-43.