




Article

Practical Implementation of Privacy Preserving Clustering Methods Using a Partially Homomorphic Encryption Algorithm

Ferhat Ozgur Catak ^{1,*}, Ismail Aydin ², Ogerta Elezaj ¹ and Sule Yildirim-Yayilgan ¹

¹ Department of Information Security and Communication Technology, NTNU Norwegian University of Science and Technology, 2815 Gjøvik, Norway; ogerta.elezaj@ntnu.no (O.E.); sule.yildirim@ntnu.no (S.Y.-Y.)

² Cyber Security Engineering, Istanbul Sehir University, 34865 Istanbul, Turkey; ismail.aydin@tubitak.gov.tr

* Correspondence: ferhat.o.catak@ntnu.no

Received: 31 December 2019; Accepted: 24 January 2020; Published: 31 January 2020



Abstract: The protection and processing of sensitive data in big data systems are common problems as the increase in data size increases the need for high processing power. Protection of the sensitive data on a system that contains multiple connections with different privacy policies, also brings the need to use proper cryptographic key exchange methods for each party, as extra work. Homomorphic encryption methods can perform similar arithmetic operations on encrypted data in the same way as a plain format of the data. Thus, these methods provide data privacy, as data are processed in the encrypted domain, without the need for a plain form and this allows outsourcing of the computations to cloud systems. This also brings simplicity on key exchange sessions for all sides. In this paper, we propose novel privacy preserving clustering methods, alongside homomorphic encryption schemes that can run on a common high performance computation platform, such as a cloud system. As a result, the parties of this system will not need to possess high processing power because the most power demanding tasks would be done on any cloud system provider. Our system offers a privacy preserving distance matrix calculation for several clustering algorithms. Considering both encrypted and plain forms of the same data for different key and data lengths, our privacy preserving training method's performance results are obtained for four different data clustering algorithms, while considering six different evaluation metrics.

Keywords: cryptography; clustering; homomorphic encryption; machine learning

1. Introduction

In recent years, there is an increasing demand for outsourced cloud systems that allow tenants to rapidly handle sensitive data that are collected from systems, including military systems, health care systems, or banking systems. Additionally, public big data systems are needed to analyze data without breach data privacy.

Machine learning [1] is a new era that can gather valuable information to make decisions in an efficient way using several public cloud systems. Occasionally, cloud platforms contain data batches with different privacy policies; however, they still have to be analyzed in a mutual way. Consider the case of different medical institutes that want to jointly build a disease diagnosis model using a machine learning algorithm. In this case, privacy policies and General Data Protection Regulation (GDPR) prevent these medical institutes from sharing with each other [2,3]. In this case, traditional machine learning methods cannot be applied. Sensitive data cannot be distributed publicly every time, due to the different privacy policies that different parties have.

1.1. Current Solutions

How can sensitive data be distributed and computed between the different computer cluster nodes without losing their privacy? Generally, in order to find a correct answer to this problem, symmetric and asymmetric (with public-private key cryptography) encryption algorithms [4] are applied to the data. The power of cryptographic algorithms consists of key pairs built on the power of privacy and randomness.

The power of cryptographic algorithms consists of key pairs built on the privacy of cryptographic keys and the strength of randomness.

In classical asymmetric key cryptographic methods, the secret key must be shared with the cloud service provider in order to perform arithmetic operations on the data. In this case, the data owner should trust to the cloud service provider. However, the need for disclosure of confidential information arises and existing laws prevent it. In addition, many asymmetric key encryption methods are not probabilistic, thus they are only one representation of a value. For example, with a secret key `1234567890abcdef`, the result of 0 will always be `B1B38CAC1F6A6BEB068BEC2C0643185`. The data sets we use today are sparse and therefore consist of a large number of 0s. If the value 0 has only one encrypted form, it can be easily estimated by an attacker. In order to protect privacy, the encryption methods must also be probabilistic.

When a privacy-preserving technique is implemented using classical cryptographic algorithms, the computation nodes are required to have appropriate cryptographic key or keys from public/private key pairs, and the keys have to be exchanged using a secure communication channel. The biggest problem in this type of system is that the cloud server that will perform the computation has keys that can access the private data in plain form [5].

Considering privacy issues related to data handling for machine learning algorithms, there are two primary approaches;

First, using the characteristic features of datasets for the generalization and suppression for anonymization. After that, the anonymized version of data can be distributed [6] to other computation parties to execute a data analytics algorithm.

Secondly, using cryptographically secure multi-party computation algorithms to build cryptographic protocols that can compute the same result with a plain form of data using an encrypted version of the data. This approach is applicable mostly when the relationship between data sharing nodes is symmetrical. A symmetrical relationship indicates that if a dataset is partitioned and distributed to different parties, then the model that is built by a machine learning algorithm and applied to the dataset is the same. Thus, the final result of the algorithm execution shows that all parties learn the same model based on a shared dataset.

The main difference between these two methods consists of the fact that in the first approach (the anonymization approach), the computation parties do not execute machine learning algorithms on the original data, which belong to themselves, and the data owner itself does not get back an output of the computation.

The objective of the method presented in this paper is to allow a user to create different cluster models using homomorphic encryption algorithm without accessing the plain version of the data. Accordingly, the data owner also would not know anything about the data clustering model. The algorithm performance was measured in different size and the clustering performance with different metrics is investigated using various metrics.

1.2. Contribution

Currently, the data, which need to be preserved privately, are part of a large volume and may differ in a wide range of ranges. There are several approaches in the literature for providing privacy for sensitive data. Classical cryptographic methods are not sufficient in each case for privacy issues with a data system. When a process needs to run on a sensitive database with the protection of classical cryptographic systems, there will be a need for a proper key for deciphering. After that, the needed

process can be executed on the decrypted version of the data. Thus, the privacy of the data is violated and there is a threat of data disclosure. Considering their competence level, it is evident that this procedure cannot be used in every situation. Of concern to this data privacy breach problem, there is always a need for a system that has to protect the confidentiality of this data when a computation is executed.

In our previous work, [7], we applied partially homomorphic encryption methods to build a probabilistic classifier using the extreme learning machine algorithm. Within the scope of this previous study, we have examined how to create classification models using homomorphic encryption algorithms. We created the privacy-protected version of the ELM algorithm, which constructs a classification model by creating a linear equation.

In this research, we have designed a system that uses Paillier Cryptography for clustering data systems without violating their privacy. This system would use the data in an efficient way while preserving the privacy of data.

The contributions of this paper are twofold. First:

- The Paillier cryptosystem encryption-based clustering model building protocol is proposed for preserving privacy and thus clustering model training is performed. Secondly:
- The computation of the distance metric matrix of four different clustering algorithms is distributed to independent parties, thus minimizing the overall computational time.

The remainder of the paper is organized as follows. Section 2 presents the related work on using cryptographic models and technologies for privacy preserving machine learning and data analysis. Section 3 describes data clustering, cluster model evaluation metrics, and homomorphic encryption. Section 4 describes our proposed privacy-preserving clustering learning model. Section 5 evaluates the proposed learning model. Finally, Section 6 concludes this paper.

2. Related Work

In this section, we describe the general overview of literature related to privacy-preserving machine learning models.

In [8], the authors suggested a privacy enhanced version of the Iterative Dichotomiser 3 (ID3) algorithm by using secure two party computation. They have stated that their learning model requires relatively less communication stages and bandwidth. In this method, the classification algorithm is combined with a decision tree algorithm, while the privacy of data is preserved as different users work. The results of each party are merged by using different cryptographic protocols. If there are more than two parties involved in the computation, their protocol cannot be applied to build a classification model. The ID3 algorithm can be used for only discrete values datasets; however, most of the today dataset contains only continuous variables.

In [9], the authors examined the balance between learnability from data and privacy, while developing a privacy preserving algorithm for sensitive data. They focused on the privacy preserving version of the logistic regression classification algorithm. Limiting the sensitivity due to distortion is calculated when a noise-adding feature is implemented to the regularized version of the logical regression classification algorithm. A privacy-preserving version of the regularized logistic regression algorithm is built, solving a perturbed optimization problem. Their approach uses a differential privacy technique to build a classifier model in a privacy-preserving environment. The differential privacy model can be applied to statistical databases only.

In [10], the authors proposed data processing methods that are aim to merge different security requirements on different platforms. They calculated the mathematical representation of the distributed data to its original form and tried to accurately approximate the true values of the original data from distributed data. They added some perturbation to the original dataset using a randomizing function.

Xu K., Yue H. et al. considered the conventional methods of cryptography for different nodes that do not share open data in respect of adequation. For this concern, the authors applied to minimize the

data that requires being computed, by using the data locality feature of the Apache Hadoop system to protecting the privacy [11]. Their approach is based on the map-reduce paradigm, which is a quite old, big data analysis technique. In this work, the authors use a random matrix to preserve the privacy of the dataset content.

In [12], the authors inspected the overheads of the privacy and transmission of partitioned data in supervised and non-supervised contexts. They concluded that is better to transmit the hyper-parameters of the generative learning models that are built on local data nodes to a central database, instead of sharing the original data. Their paper showed that through reproducing artificial samples from the original data probability distributions using Markov Chain Monte Carlo method, it is mathematically possible to represent all the data with a mean model. In their approach, the authors do not share a perturbed matrix, they instead produce artificial samples from the underlying distributions using the Markov Chain Monte Carlo technique. Producing artificial data can sometimes cause errors on the model. The resulting model may not perform adequately on the test dataset.

In [13], the authors used artificial neural networks to gather information and construct models from various datasets. Their aim was to design a practical system that allows different parties to jointly build an accurate neural network model without sharing their private data. The authors concluded that their approach has a strong privacy-preserving component compared to any existing method due to the minimal data sharing between parties, which is actually a small number of neural network hyper-parameters. In this paper, the authors proposed a distributed version of the stochastic gradient descent (SGD) optimization algorithm. Their classification model evaluation results are almost the same to the original model evaluation results. However, there are many other optimization techniques that have more efficiency over time.

In [14], the authors proposed a system that allows users an on point representative model for sensitive data, which also protects its privacy. The system calculates the frequencies of specific values or a group of values. In this phase, the model protects the data privacy. The authors stated that there is no information sharing except the frequency of data values. The proposed model works only in a specific scenario where each customer (or party) sends only one flow of communication to the central miner, and no customer sends data to other customers.

In [15], the authors developed a system that guarantees the data from a single party data source will not be disclosed. In order to protect the data privacy, Shamir's secret sharing is used with the distributed version of the decision tree learning, ID3 algorithm. The ID3 algorithm can be used for only discrete values datasets.

In [16], the authors used a multi key-fully homomorphic encryption alongside a hybrid structure, which combined double decryption. The authors stated that these two privacy-preserving algorithms are acceptable to use with different types of deep learning algorithms over encrypted data. Each party in computation chooses its key pairs and encrypts its own data. Encrypted data is sent to a cloud system and the computation over the data is executed by these two systems.

In [17], the authors developed a privacy preserving version of the Bayes classifier learning method for horizontally partitioned data and proposed two protocols. One of these protocols is a secure two-party protocol and the other one is a secure multi-party protocol. The secure multi-party protocol is used between owners of sensitive data and a semi trusted server for sending messages that have to be classified. The two party protocol is used for sensitive data between a user and a semi-trusted server for broadcasting the classification results made on the server. In this work it is assumed that these two protocols are trusted and can preserve privacy. Their approach can be applied only into a Bayesian classifier.

In [18], the authors proposed a parallel computing approach that can perform with high performance on the computationally intensive works of data mining. In this study they considered a system built on the idea of the existence of a cluster and a grid resource in one tier, and in another tier the system would simply abstract the communication for algorithm development. Their framework is applied only for classification algorithms.

3. Preliminaries

In this section, we briefly introduce clustering methods, homomorphic encryption, and the Paillier Cryptosystem used for our proposed privacy-preserving cluster learning model. We also introduce the evaluation metrics used to analyze the experimental results.

3.1. Data Clustering

Clustering can be described as combining the same instances of a dataset into the same group depending on special features of the data. For the evaluation criteria of a clustering model, the elements that are similar to each other should be in the same group as much as possible.

There are four different approaches for clustering algorithms: centroid-based, distribution-based, density-based, and connectivity-based.

Centroid-Based Clustering: Centroid-based clustering [19] presents a cluster by a central vector, which does not have to be an instance of the dataset. The required number of clusters should be predetermined and this is the main drawback of this type of algorithm (such as the K-Means algorithm). After deciding the number of clusters, then the central vectors for each cluster are computed by different distance metrics from the clusters to find the nearest instance to form clusters.

Distribution-Based Clustering: In this type of algorithm, instances of training data are clustered due to their statistical characteristics. Clusters can be defined as instances belonging most likely to the same distribution. One well-known mixture model for this method is a *Gaussian mixture* [20]. A dataset at the start is modeled with a *Gaussian distribution* arbitrarily and then the parameters are optimized to fit the dataset. It is an iterative algorithm, thus the output will converge into an optimum model.

Density-Based clustering: In this type of algorithm, clusters are defined, taking into consideration the area of high density of instances in the input dataset. Density-based clustering algorithms use different criteria for defining the density. One of the well-known criteria is called *density reachability* [21]. This logic works for searching for instances in a dataset that are within a certain threshold value, and adding those instances into the same cluster.

Connectivity-based clustering: This clustering method collects instances of an input dataset that are more interconnected to nearby instances, than farther away instances, to build a cluster model. In this approach, clusters are built based on their distance metrics, thus, a cluster can be defined by the maximum distance metric needed to group instances. Different clusters will be built at different distances, as a result, this approach can be visualized with a *dendrogram*, because these type of clustering methods yield a hierarchical model. As a result, a certain cluster also can be merged with another cluster.

There are a variety of different clustering algorithms. In our work, we analysed the *K-Means*, *Hierarchical*, *Spectral*, and *Birch* clustering algorithms.

The *K-Means algorithm* builds a cluster model considering *distance metrics* between input instances. The algorithm is applied generally when the data has a flat geometry, creating only a few clusters, and the number of clusters is evenly sized.

The *Hierarchical algorithm* builds a cluster model, taking into consideration the *pairwise distance metrics* between input instances. This algorithm is applicable generally when creating a large amount of clusters, and when there are possible connectivity constraints to building clusters.

The *Spectral algorithm* builds a cluster model taking into account the nearest-neighbors [22]. This algorithm is applicable generally when the input data has a non-flat geometry, creating only a few clusters and the number of clusters is even sized.

The *Birch algorithm* builds a cluster model taking into consideration the *Euclidean distance* between input instances. This algorithm is applicable generally when the data is large and data reduction, in respect to outlier removal, is needed.

3.2. Evaluation Metrics

In this paper, our proposed system is evaluated in consideration of six different metrics.

Homogeneity: This metric can only be fulfilled if the members of each single class are assigned into a distinct cluster in terms of homogeneity [23]. Thus, each cluster consists of only instances of a single class. The class distribution within each cluster should be to only a single class. This metric gets a value between 0 and 1. If clustering is done in the most ideal way, than the homogeneity value will be equal to 1. The homogeneity value can be evaluated to understand the quality of the clustering of elements belonging to the same class in the same cluster. The ratio of current homogeneity value and the ideal value can be expressed as $H(C|K)$, and this expression is equal to 0 when there is a perfect clustering. The value of $H(C|K)$ is dependent on the size of the dataset, therefore, instead of using this value, the normalized version is used and it is;

$$\frac{(H(C|K))}{(H(C))}. \tag{1}$$

Thus, the output value with 1 is desirable, and 0 is the undesirable state, the homogeneity can be defined as:

$$h = \begin{cases} 1 & \text{if } H(C, K) = 0 \\ 1 - \frac{H(C|K)}{H(C)} & \text{else} \end{cases} \tag{2}$$

where

$$H(C|K) = - \sum_{k=1}^{|K|} \sum_{c=1}^{|C|} \frac{a_{ck}}{N} \log \frac{a_{ck}}{\sum_{c=1}^{|C|} a_{ck}}$$

$$H(C) = - \sum_{c=1}^{|C|} \frac{\sum_{k=1}^{|K|} a_{ck}}{n} \log \frac{\sum_{k=1}^{|K|} a_{ck}}{n}. \tag{3}$$

Completeness is a symmetrical version of the homogeneity metric, expressing the balance of instances belonging to the same class, inside the same cluster. If the instances belonging to the same class are represented in the same cluster as the result of the clustering of the data, this metric, which takes the ideal value in this situation, is regarded as 1. If the grouping is far from the ideal state, the value of this metric will be 0. In order to satisfy this criterion, each of the clusters should be comprised of elements that belong to only one class. The distribution of cluster labeling inside each single class label is used to analyze the completeness metric. In ideal conditions, the value will be 0, $H(K|C) = 0$. The worst case conditions occur when each class is labeled by each cluster. In this case, the value will be 0, $H(K) = 0$. Completeness can be defined as:

$$c = \begin{cases} 1 & \text{if } H(K, C) = 0 \\ 1 - \frac{H(K|C)}{H(K)} & \text{else} \end{cases} \tag{4}$$

where

$$H(K|C) = - \sum_{c=1}^{|C|} \sum_{k=1}^{|K|} \frac{a_{ck}}{N} \log \frac{a_{ck}}{\sum_{k=1}^{|K|} a_{ck}}$$

$$H(K) = - \sum_{k=1}^{|K|} \frac{\sum_{c=1}^{|C|} a_{ck}}{n} \log \frac{\sum_{c=1}^{|C|} a_{ck}}{n}. \tag{5}$$

V-Measure is a balance metric between homogeneity and completeness criteria. *V-Measure* is calculated by using the harmonic mean of the homogeneity and completeness values of the cluster model. This metric value is between 0 and 1. As described above in previous homogeneity and completeness sections, these two metrics have working logics that are opposite to each other.

An increase in the homogeneity value results in a decrease in the completeness value, and vice versa. The *V-Measure* is shown as:

$$v - Measure = \frac{(2 * homogeneity * completeness)}{(homogeneity + completeness)}. \quad (6)$$

Adjusted Rand Index: The *Rand Index* metric is a measure of similarities between two data clustering labels, and should be calculated in order to find the *Adjusted Rand Index* (ARI). While the *Rand Index* may vary between 0 and 1, the ARI can also have negative values. ARI metric is shown as:

$$ARI = \frac{(RI - Expected RI)}{(max(RI) - Expected RI)}. \quad (7)$$

ARI becomes 0 when the clustering is done randomly and independent of the number of clusters; however, if the clusters are similar or identical, then the metric value becomes 1. This metric is also symmetrical.

$$(adjusted_rand_score(a, b) == adjusted_rand_score(b, a)). \quad (8)$$

Adjusted Mutual Information For this metric, as with ARI, mutual information describes how much information is shared between different clusters. Thus, *adjusted mutual information* can be considered as a similarity metric. In our paper, adjusted mutual information (AMI) measures the number of mutual instances between different clusters. This metric value becomes 1 when the clusters are completely identical, and when the clusters are independent from each other this metric value is equal to 0. Thus, there is no information shared between clusters. AMI is the adjusted shape of mutual information. The mutual information value is shown as:

$$AMI(U, V) = \frac{MI(U, V) - E(MI(U, V))}{max(H(U), H(V)) - E(MI(U, V))}. \quad (9)$$

This metric is also symmetrical, as with the ARI metric.

The *Silhouette Coefficient* metric is calculated by using both intra-cluster distance and mean nearest-cluster distance and is the distance between an instance and the nearest cluster that the instance that is not a part of each of the instances in a dataset. The silhouette coefficient is shown as:

$$Silhouette\ Coeff. = \frac{(b - a)}{max(a, b)} \quad (10)$$

where b is the distance value between an instance and the nearest cluster (which the instance does not belong to) and a is the mean value of distances within the cluster that the instance is a part of. This metric obtains the mean value of all silhouette coefficient values for the instances in the dataset.

The Silhouette coefficient can achieve values between -1 and 1 . When the Silhouette coefficient is closer to -1 , then it is more likely that the instance is in the wrong cluster. If this metric is considered in all instances in the dataset, the more the value moves closer to -1 . In this case, the clustering is not accurate and the instances are more likely assigned to wrong clusters. On the contrary, if the metric value gets closer to 1 , then, the clustering model is more accurate. When the metric value is near 0 , then the clusters are overlapped.

3.3. Homomorphic Encryption

Homomorphic encryption allows for computing with encrypted data and achieves the same results with the plain version of the data. The most important feature of this type of cryptographic scheme is to preserve the privacy of the sensitive data [24] as they allow work on the encrypted data instead of its plain form. Homomorphic encryption schemes can also be used in connecting different kinds of services without putting at risk the exposure of sensitive data. Homomorphical encryption systems can be divided into two different groups; partially homomorphic algorithms and

fully homomorphic algorithms. In our research, we applied the partially additive homomorphic Paillier Cryptosystem.

One can say that a public-key encryption scheme is additively homomorphic if, given two encrypted messages, such as $\llbracket a \rrbracket$ and $\llbracket b \rrbracket$, there exists a public-key summation operation \oplus such that $\llbracket a \rrbracket \oplus \llbracket b \rrbracket$ is an encryption of the plaintext of $a + b$. The formal definition is that an encryption scheme is additively homomorphic if for any private key, public key (sk, pk) , the plaintext space $\mathcal{P} = \mathbb{Z}_N$ for $a, b \in \mathbb{Z}_N$.

$$\begin{aligned} Enc_{pk}(a + b \text{ mod } N) &= Enc_{pk}(x) \times Enc_{pk}(y) \\ Enc_{pk}(x \cdot y \text{ mod } N) &= Enc_{pk}(x)^y. \end{aligned} \quad (11)$$

3.3.1. Paillier Cryptosystem

The Paillier cryptosystem [25] is an asymmetric, probabilistic, and public key cryptosystem. It preserves privacy [26] depending on the complexity of the problem of computing the n -th residue classes. A Paillier cryptosystem is a partially homomorphic system, which means that the encryption of M_1 and M_2 plain datasets with a K public key gives the same result as the encryption of multiplication of the same two datasets $(M_1 + M_2)$, using the same K public key. This encryption algorithm works by performing the two main jobs in an order. The first one is the key generation and the second one is the encryption/decryption of the dataset.

The Paillier cryptographic system has partial homomorphic properties, which makes this algorithm more appropriate against data exposure and it is used in several fields that have sensitive data, such as medicine, finance, and the military. These partial homomorphic properties are:

- The addition of two encrypted messages: The result of adding two encrypted messages that are exactly the same with the result of the plain version of adding these two messages.
- Multiplication of an encrypted message with a non-encrypted message: Multiplying an encrypted message with a number N is same with multiplying the plain form of that message with the same message N and encrypting it.

Let us give a set of possible plaintext messages M and a set of secret and public key pairs $K = pk \times sk$, where pk is the public key and sk is the secret key of the cryptosystem. Then the Paillier homomorphic encryption cryptosystem satisfies the following property for any two plaintext messages m_1 and m_2 and a constant value a :

$$Dec_{sk} \left(Enc_{pk} (m_1) \times Enc_{pk} (m_2) \right) = m_1 + m_2 \quad (12)$$

$$Dec_{sk} \left(Enc_{pk} (m_1)^a \right) = a \times m_1. \quad (13)$$

One of the main properties of the Paillier cryptosystem is that it is based on a probabilistic encryption algorithm. Large-scale data sets are sparse matrices, in which most of the elements are zero. In order to prevent the guessing of elements of the input data set, the Paillier cryptosystem has probabilistic encryption that does not encrypt two equal plaintexts with the same encryption key into the same ciphertext.

3.3.2. Floating Point Numbers

As the Paillier encryption systems works only on integer values, the proposed protocols are only capable of handling integers. This is considered to be an obstacle in applying these algorithms, as mostly of the real datasets contain continuous values. Nonetheless, in the case of an input dataset with real numbers in the protocol, we need to map the floating point input data vectors into the discrete domain with a conversion function (i.e., scaling).

The number of digits in the fractional part of the data, which is defined as a floating point number, can be very large and if these numbers are to be used, then it would definitely require more processing power and processing time. For this reason, in this work only the first five fractional digits have been used, and, due to this fact, the fractional digits of input data more than these digits are not used. These may be possibly generated as a result of computational work and are rounded into five digits. Using this approach potentially creates minimal data losses and deviations of computation. It is seen that this effect on the calculation result depends on the grouping algorithm used, but overall it does not affect the effectiveness of the algorithms.

4. System Model

4.1. Development Environment

In our experiments, we used Python 64-bit 3.3 to implement the mentioned algorithms to encrypt the training datasets using Paillier cryptography. We used scikit-learn, scipy python libraries to build privacy-preserving versions of four different clustering algorithms, which have been described in Section 3.

Each algorithm was run on a computer with an Intel Core i7-6700HQ CPU @ 2.60GHz octa core processor (4 real and +4 pseudo) along with 16 GB of RAM. Parallel computing has not been used in every experiment but where an algorithm allows the use of all processor cores while running, that property has been used.

4.2. Sequence Diagram

In this section, we explain our privacy preserving clustering training approach with several sequence diagrams. This system not only aims to preserve privacy but also aims to analyze and handle the data efficiently in terms of execution time and processing power. The model has two distinct phases; the client computation and the model building.

4.2.1. Client Computation

In this research, the client has the plain version of the data. The main task of the client side is to handle and to encrypt the sensitive data when it is needed. The client generates a public/private Paillier key pair by establishing a key exchange session with cloud service providers, then sends its public key and the encrypted form of the training data.

The cloud service provider also does not store the plain version of the sensitive data but has a huge computation infrastructure to compute intensive data analysis. On the cloud servers, the computations are executed to get an encrypted distance metric matrix using Paillier crypto-system encrypted data that will be used by the clustering algorithms. Cloud servers perform all computations for the tenant without violating their privacy and it sends the computation results (distance metric matrix). The tenant, then uses the encrypted distance metric matrix in order to build a cluster model. As illustrated in Figure 1, in this system, the tenant side does not require the plain data to use, but instead the cluster information can be derived from the encrypted distance metric matrix.

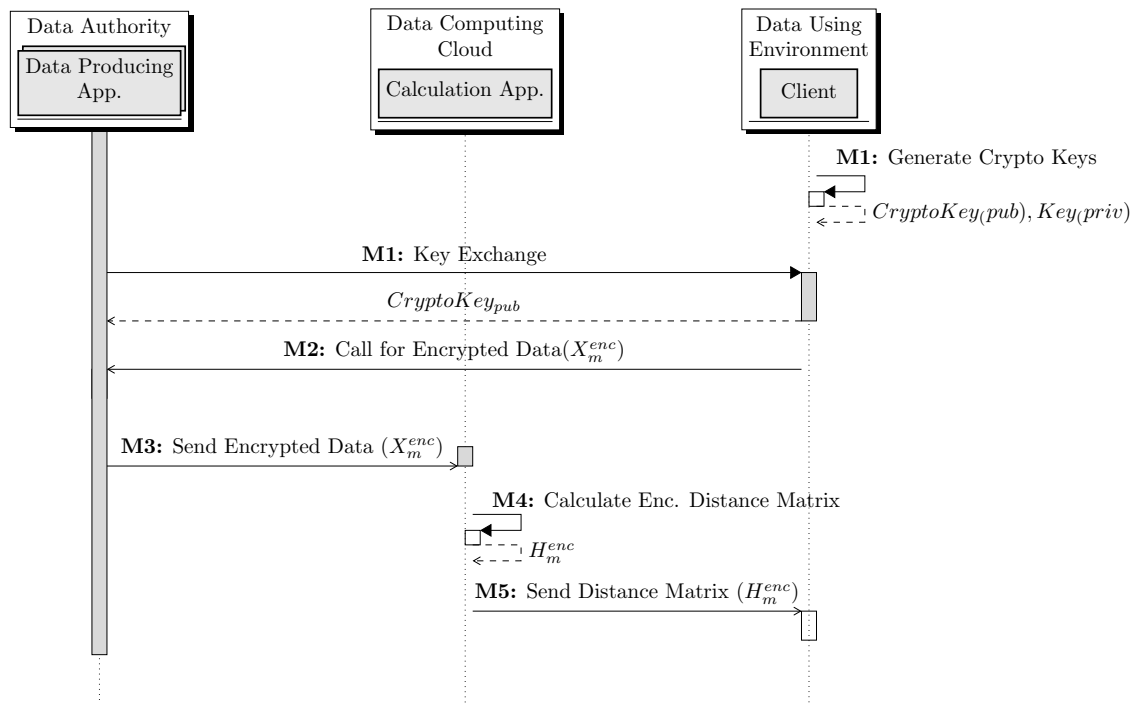


Figure 1. Sequence diagram for the client side.

The pseudocode for the *Client Computation* phase of the proposed system is shown in Algorithm 1. In lines 2–3, the client creates public/private crypto keys, keeps the private key internally, and sends the public key to the cloud service provider. In lines 4–5, the client encrypts the training dataset and sends it to the cloud service provider. At this stage, the cloud service provider has access to the public key and to the data to perform the calculation. In lines 6–7, the cloud server performs distance calculations on the encrypted data using the metric information given as parameters. It sends the encrypted distance matrix back to the client.

Algorithm 1 Client initialization and distance computation.

- 1: **Inputs:**
Dataset, \mathcal{X} , Crypto key length: l , metric type: m
 - 2: $(Key_{pub}, Key_{priv}) \leftarrow KeyGen(l)$ ▷ Key Generation
 - 3: $res \leftarrow sendCryptoKey(Key_{pub})$ ▷ send public key to cloud service provider
 - 4: $[\mathcal{X}] \leftarrow encrypt(Key_{priv}, \mathcal{X})$ ▷ encrypt dataset
 - 5: $sendEncryptedDataSet([\mathcal{X}])$ ▷ send encrypted dataset to cloud service provider
 - 6: $[\mathcal{H}] \leftarrow distanceMatrix([\mathcal{X}], m)$ ▷ Cloud service provider computes the distance matrix of each input instance according to metric parameter
 - 7: $res \leftarrow sendDistanceMatrix([\mathcal{H}])$ ▷ Cloud service provider sends the encrypted distance matrix to client
-

4.2.2. Model Building at the Client Side

At this stage, the client has an encrypted distance matrix. The first action of the client is to decrypt this matrix using the private key Key_{priv} . Since the clustering algorithms used in this study are distance based, the clustering model can be constructed by using the distance matrix as a parameter, explained in detail in Algorithm 2. As illustrated in Figure 2, the client receives the encrypted distance matrix and decrypts it using the private key.

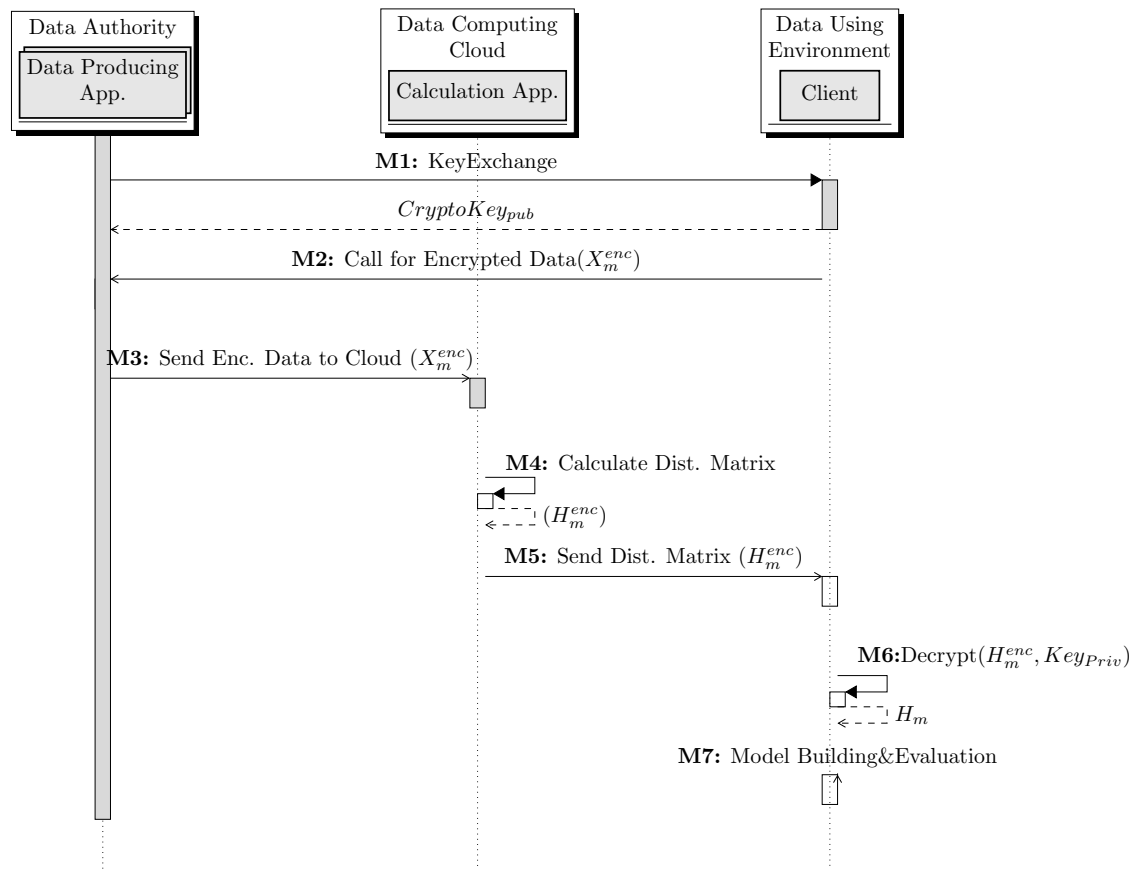


Figure 2. Sequence diagram for model building at the client side.

Algorithm 2 Cluster model building.

1: **Inputs:**

Encrypted distance matrix: $\llbracket \mathcal{H} \rrbracket$, Crypto keys: Key_{pub}, Key_{priv} , Cluster

algorithm: \mathcal{C}

2: $\mathcal{H} \leftarrow decryptDistanceMatrix(\llbracket \mathcal{H} \rrbracket)$

▷ Client decrypt the distance matrix

3: $m \leftarrow clusteringAlgorithm(\mathcal{C}, \mathcal{H})$

▷ Run clustering algorithm with plain domain distance matrix

4: **Outputs:**

Clustering model m

5. Experimental results

In this research, the clustering methods that have been described in Section 3.1 were examined by running each of them on 10 different datasets (from 500 to 5000 instances) using five different crypto key lengths. The data length of each dataset is also the name of the dataset (dataset 500, dataset 1000 etc.) and the used key lengths vary between 64 bits and 1024 bits.

All algorithms were run on a Python environment and all algorithm codes have been modified to create the same number of clusters (20 clusters) from the given data.

5.1. Plaintext Results

The experimental results for the plain domain computations are shown in Table 1. As no key is used, there is only one column of the table for the score of each evaluation metric. In the case of plain data computations, the same metric scores have been obtained except for the silhouette coefficient. The change is not large enough to change the metric score, thus it can be neglected. The evaluation

metrics obtained the maximum scores in plain data computation (except *silhouette coefficient*), as the used data are *artificial* and not *real*.

According to Table 1, all clustering performance metrics are the same for all algorithms. The K-means clustering algorithm takes more time to build a model for 1000–3000 row datasets, Hierarchical clustering needs more time for 4000–5000 row datasets, and the Spectral algorithm needs a lower time for training for all datasets.

Table 1. Plain domain evaluation of metric scores.

Length	Algorithm	Homogeneity	Completeness	V-Measure	Adj. Rand Ind.	Adj. Mut. Inf	Silhouette Coeff.	Time (s)
1000	K-means	1.0	1.0	1.0	1.0	1.0	1.0	36.95
	Hierarchical	1.0	1.0	1.0	1.0	1.0	1.0	14.89
	Spectral	1.0	1.0	1.0	1.0	1.0	1.0	8.62
	Birch	1.0	1.0	1.0	1.0	1.0	1.0	15.07
2000	K-means	1.0	1.0	1.0	1.0	1.0	1.0	163.58
	Hierarchical	1.0	1.0	1.0	1.0	1.0	1.0	123.05
	Spectral	1.0	1.0	1.0	1.0	1.0	1.0	26.47
	Birch	1.0	1.0	1.0	1.0	1.0	1.0	96.95
3000	K-means	1.0	1.0	1.0	1.0	1.0	1.0	413.85
	Hierarchical	1.0	1.0	1.0	1.0	1.0	1.0	410.76
	Spectral	1.0	1.0	1.0	1.0	1.0	1.0	53.83
	Birch	1.0	1.0	1.0	1.0	1.0	1.0	347.99
4000	K-means	1.0	1.0	1.0	1.0	1.0	1.0	789.06
	Hierarchical	1.0	1.0	1.0	1.0	1.0	1.0	964.55
	Spectral	1.0	1.0	1.0	1.0	1.0	1.0	98.60
	Birch	1.0	1.0	1.0	1.0	1.0	1.0	795.86
5000	K-means	1.0	1.0	1.0	1.0	1.0	1.0	1292.19
	Hierarchical	1.0	1.0	1.0	1.0	1.0	1.0	1592.06
	Spectral	1.0	1.0	1.0	1.0	1.0	1.0	148.25
	Birch	1.0	1.0	1.0	1.0	1.0	1.0	1509.15

5.2. Encrypted Domain Results

The experimental results of the encrypted domain computations are shown in Tables 2–5. Based on these results, we conclude that there is no need to use 64 bit and 128 bit keys, as, these keys did not effect the results of this work and, for this reason, the scores of 64 and 128 bit keys are not shown on the table.

According to Table 2, all clustering performance metrics are different for all algorithms. Generally Birch clustering algorithm takes more time to build a model, Spectral clustering needs needs lower time for training for all datasets. Generally Spectral clustering algorithm performance results are better than the other algorithms.

Table 2. Encrypted domain evaluation of metric scores for 256 bit key length.

Length	Algorithm	Homogeneity	Completeness	V-Measure	Adj. Rand Ind.	Adj. Mut. Inf	Silhouette Coeff.	Time (s)
1000	K-Means	0.741	0.724	0.732	0.488	0.705	0.350	228.41
	Hierarchical	0.708	0.661	0.684	0.422	0.638	0.272	167.56
	Spectral	0.842	0.848	0.845	0.694	0.832	0.291	219.22
	Birch	0.735	0.610	0.667	0.427	0.588	0.292	175.84
2000	K-Means	0.811	0.793	0.802	0.720	0.786	0.352	849.30
	Hierarchical	0.722	0.706	0.714	0.504	0.696	0.306	725.57
	Spectral	0.822	0.829	0.825	0.676	0.816	0.273	646.24
	Birch	0.706	0.569	0.630	0.364	0.557	0.282	1275.94
3000	K-Means	0.743	0.727	0.735	0.543	0.721	0.338	2056.95
	Hierarchical	0.703	0.675	0.689	0.428	0.669	0.295	2163.89
	Spectral	0.837	0.837	0.835	0.710	0.830	0.277	1427.10
	Birch	0.719	0.520	0.603	0.296	0.511	0.267	3687.20
4000	K-Means	0.795	0.781	0.788	0.671	0.788	0.345	3194.87
	Hierarchical	0.641	0.688	0.664	0.384	0.636	0.264	3254.51
	Spectral	0.777	0.780	0.779	0.579	0.774	0.281	3012.27
	Birch	0.721	0.533	0.613	0.363	0.527	0.271	3222.27
5000	K-Means	0.766	0.751	0.759	0.604	0.748	0.340	6257.98
	Hierarchical	0.654	0.686	0.670	0.405	0.649	0.245	5737.90
	Spectral	0.793	0.799	0.796	0.625	0.791	0.288	4031.74
	Birch	0.723	0.581	0.644	0.373	0.577	0.245	7191.10

According to Table 3, generally the Spectral clustering algorithm performance results are better than the other algorithms.

Table 3. Encrypted domain evaluation of metric scores for 512 bit key length.

Length	Algorithm	Homogeneity	Completeness	V-Measure	Adj. Rand Ind.	Adj. Mut. Inf	Silhouette Coeff.	Time (s)
1000	K-Means	0.741	0.724	0.732	0.488	0.705	0.350	725.91
	Hierarchical	0.739	0.679	0.708	0.453	0.657	0.296	579.17
	Spectral	0.815	0.819	0.817	0.646	0.803	0.284	770.70
	Birch	0.735	0.610	0.667	0.427	0.588	0.292	580.24
2000	K-Means	0.811	0.793	0.802	0.720	0.786	0.352	2994.48
	Hierarchical	0.747	0.728	0.737	0.540	0.719	0.298	2329.03
	Spectral	0.833	0.840	0.836	0.707	0.828	0.270	2298.56
	Birch	0.706	0.569	0.630	0.364	0.557	0.282	4679.90
3000	K-Means	0.743	0.727	0.735	0.543	0.721	0.338	5541.25
	Hierarchical	0.699	0.670	0.684	0.454	0.663	0.292	6722.31
	Spectral	0.817	0.821	0.819	0.678	0.813	0.269	5333.42
	Birch	0.719	0.520	0.603	0.296	0.511	0.267	5497.33
4000	K-Means	0.795	0.781	0.788	0.671	0.788	0.345	10,375.71
	Hierarchical	0.659	0.703	0.680	0.407	0.654	0.264	9928.27
	Spectral	0.766	0.770	0.768	0.556	0.762	0.283	11,371.57
	Birch	0.721	0.533	0.613	0.363	0.527	0.271	9963.32
5000	K-Means	0.766	0.751	0.759	0.604	0.748	0.340	15,636.90
	Hierarchical	0.634	0.659	0.646	0.360	0.629	0.245	15,691.37
	Spectral	0.806	0.813	0.809	0.648	0.804	0.280	14,343.40
	Birch	0.723	0.581	0.644	0.373	0.577	0.245	16,862.45

According to Table 4, generally, the Spectral clustering algorithms need a lower training time. The Spectral clustering algorithm performance results are better than the other algorithms.

Table 4. Encrypted domain evaluation of metric scores for 1024 bit key length.

Length	Algorithm	Homogeneity	Completeness	V-Measure	Adj. Rand Ind.	Adj. Mut. Inf	Silhouette Coeff.	Time (s)
1000	K-Means	0.741	0.724	0.732	0.488	0.705	0.350	4186.54
	Hierarchical	0.727	0.671	0.698	0.429	0.649	0.277	3534.10
	Spectral	0.823	0.822	0.822	0.644	0.810	0.295	3458.16
	Birch	0.735	0.610	0.667	0.427	0.588	0.292	3977.07
2000	K-Means	0.811	0.793	0.802	0.720	0.786	0.352	15,613.31
	Hierarchical	0.706	0.717	0.712	0.490	0.696	0.299	14,633.00
	Spectral	0.841	0.848	0.845	0.730	0.836	0.271	13,878.68
	Birch	0.706	0.569	0.630	0.364	0.557	0.282	14,160.69
3000	K-Means	0.743	0.727	0.735	0.543	0.721	0.338	34,827.04
	Hierarchical	0.673	0.702	0.687	0.455	0.666	0.271	34,352.03
	Spectral	0.814	0.820	0.817	0.688	0.810	0.266	31,492.53
	Birch	0.719	0.520	0.603	0.296	0.511	0.267	39,601.76
4000	K-Means	0.795	0.781	0.788	0.671	0.788	0.345	56,810.99
	Hierarchical	0.646	0.689	0.667	0.389	0.640	0.264	58,259.58
	Spectral	0.777	0.780	0.778	0.576	0.773	0.283	56,022.39
	Birch	0.721	0.533	0.613	0.363	0.527	0.271	56,957.61
5000	K-Means	0.766	0.751	0.759	0.604	0.748	0.340	93,672.79
	Hierarchical	0.651	0.680	0.665	0.401	0.646	0.245	90,506.86
	Spectral	0.792	0.798	0.795	0.621	0.790	0.288	96,517.34
	Birch	0.723	0.581	0.644	0.373	0.577	0.245	90,333.51

According to Table 5, generally, the Spectral clustering algorithm performance results are better than the other algorithms.

Table 5. Encrypted domain evaluation of metric scores for 2048 bit key length.

Length	Algorithm	Homogeneity	Completeness	V-Measure	Adj. Rand Ind.	Adj. Mut. Inf	Silhouette Coeff.	Time (s)
1000	K-Means	0.746	0.729	0.739	0.490	0.708	0.351	28,773.89
	Hierarchical	0.729	0.675	0.705	0.435	0.649	0.290	24,757.46
	Spectral	0.835	0.839	0.836	0.663	0.819	0.295	19,632.64
	Birch	0.744	0.618	0.677	0.435	0.592	0.301	29,590.15
2000	K-Means	0.813	0.799	0.808	0.724	0.79	0.355	99,402.12
	Hierarchical	0.735	0.717	0.731	0.516	0.703	0.311	104,967.10
	Spectral	0.833	0.842	0.839	0.705	0.835	0.272	96,352.80
	Birch	0.709	0.572	0.636	0.370	0.564	0.284	45,457.64
3000	K-Means	0.744	0.735	0.736	0.545	0.731	0.344	255,790.18
	Hierarchical	0.699	0.690	0.694	0.453	0.668	0.286	220,325.90
	Spectral	0.824	0.831	0.832	0.697	0.821	0.277	214,712.31
	Birch	0.724	0.524	0.610	0.297	0.512	0.274	337,003.50
4000	K-Means	0.797	0.782	0.788	0.680	0.790	0.349	377,794.82
	Hierarchical	0.650	0.702	0.674	0.399	0.652	0.268	406,524.12
	Spectral	0.778	0.785	0.780	0.573	0.771	0.285	338,876.22
	Birch	0.730	0.536	0.615	0.368	0.532	0.271	391,132.86
5000	K-Means	0.776	0.758	0.762	0.609	0.749	0.349	680,925.54
	Hierarchical	0.653	0.685	0.665	0.392	0.646	0.251	636,428.78
	Spectral	0.800	0.812	0.807	0.640	0.803	0.290	706,719.43
	Birch	0.727	0.581	0.648	0.383	0.583	0.252	628,563.14

Figures 3–6 show the execution time of the algorithms during the encryption of plain data and the calculation of the distance matrix of the encrypted data for each algorithm. If the amount of data and key size increases, the time complexity of the algorithm increases.

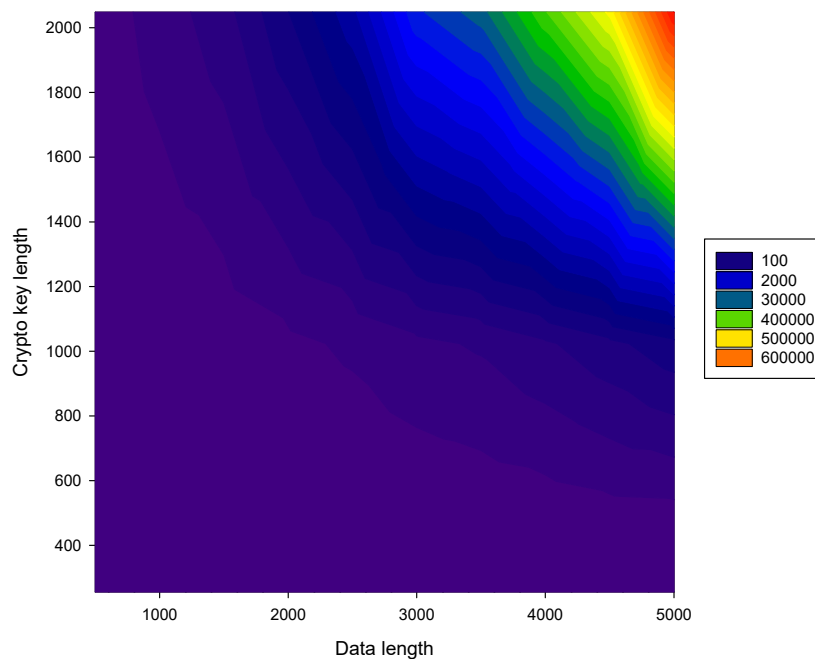


Figure 3. Calculation time graph for the KMeans algorithm.

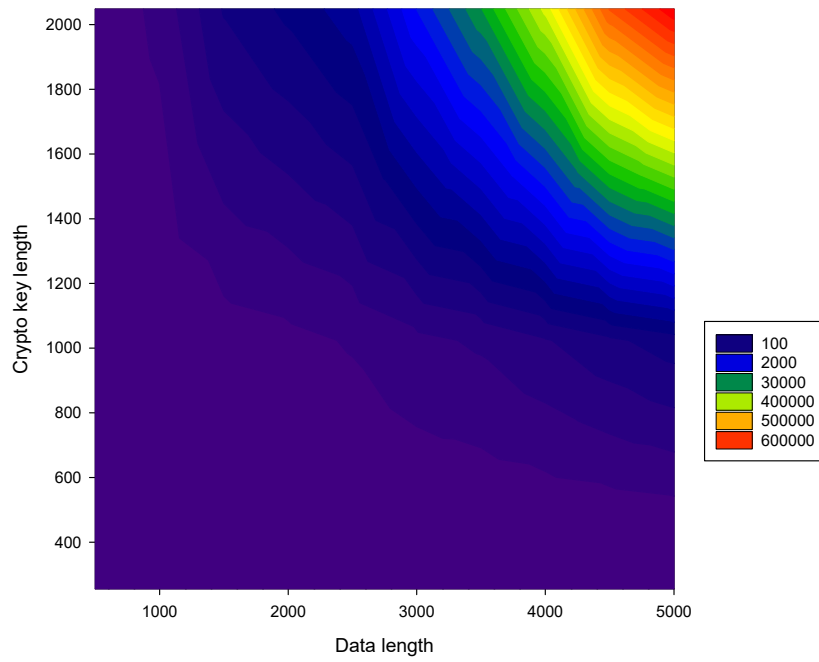


Figure 4. Calculation time graph for the Hierarchical algorithm.

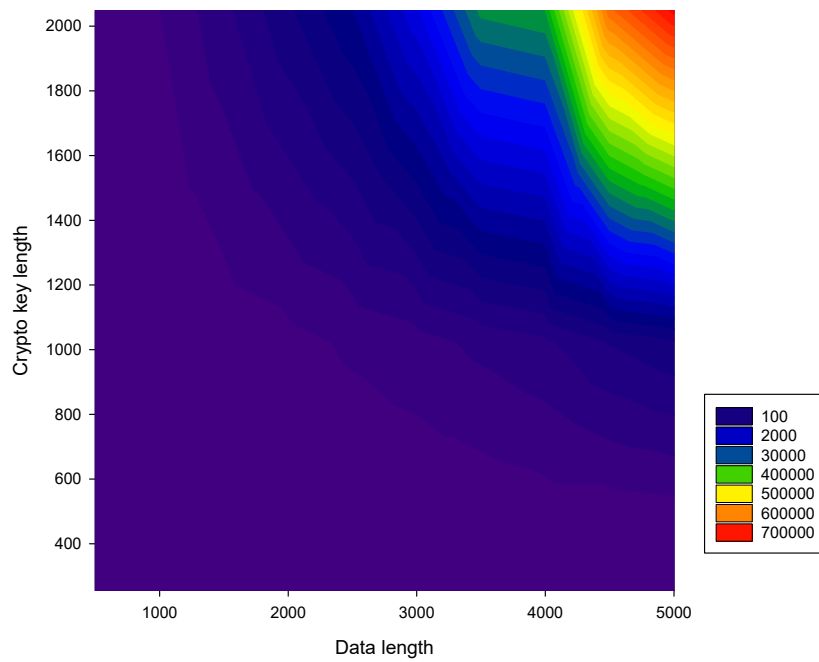


Figure 5. Calculation time graph for the Spectral algorithm.

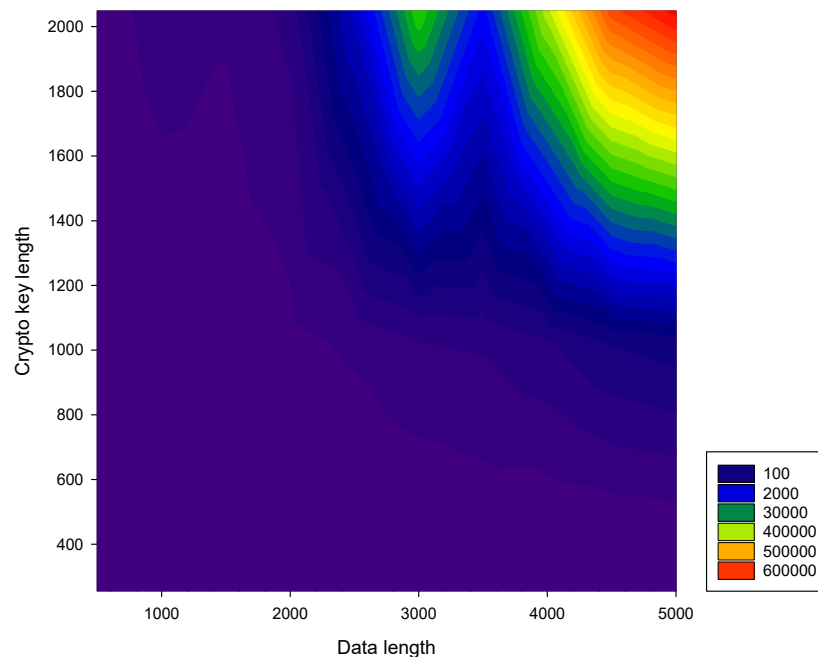


Figure 6. Calculation time graph for the Birch algorithm.

5.3. Results

As expected, based on the experimental results, the encrypted domain execution time is more than the plain data execution. As the Paillier crypto key size increases, the execution time also changes accordingly in the encrypted data execution. In the case of plain data, the execution time only changes by changing the data size.

Considering all six cluster evaluation metrics and time, we proposed a practical implementation of privacy preservation on the clustering training model using a partially homomorphic Paillier crypto system. Our proposed method is more efficient as we obtained better scores on the evaluation metrics and stronger key usage.

Based on the results presented in Section 5.2 and the computation results shown in Tables 2–5, we conclude that 2048 bit crypto key is the approach that provides the best results for all evaluation metrics. When it comes to the execution time of proposed training model it is clearly evident from the figures that this approach is not efficient enough. Even though we have achieved the best results in the clustering model performance with a crypto key with a 2048 bit length, it seems that the training time takes quite a long time. The training phase of the 2048 bit length crypto key is about eight times more of the training with the 1024 bit length crypto key. Security standards state that it is necessary to use crypto keys with a minimum length of 2048 bits in order to ensure privacy [27]. Although the 2048 bit key length is required by security standards, we also shared the results of other key lengths in this study. Figure 7 shows the duration of the clustering algorithm’s training phase with a 5000 row dataset when using crypto keys from 256, 512, 1024, and 2048 bit lengths.

If we examine the resulting Tables 2–5 that show cluster evaluation metric results of plain data clustering, we can conclude that the plain data would get perfect evaluation metric scores except silhouette coefficient.

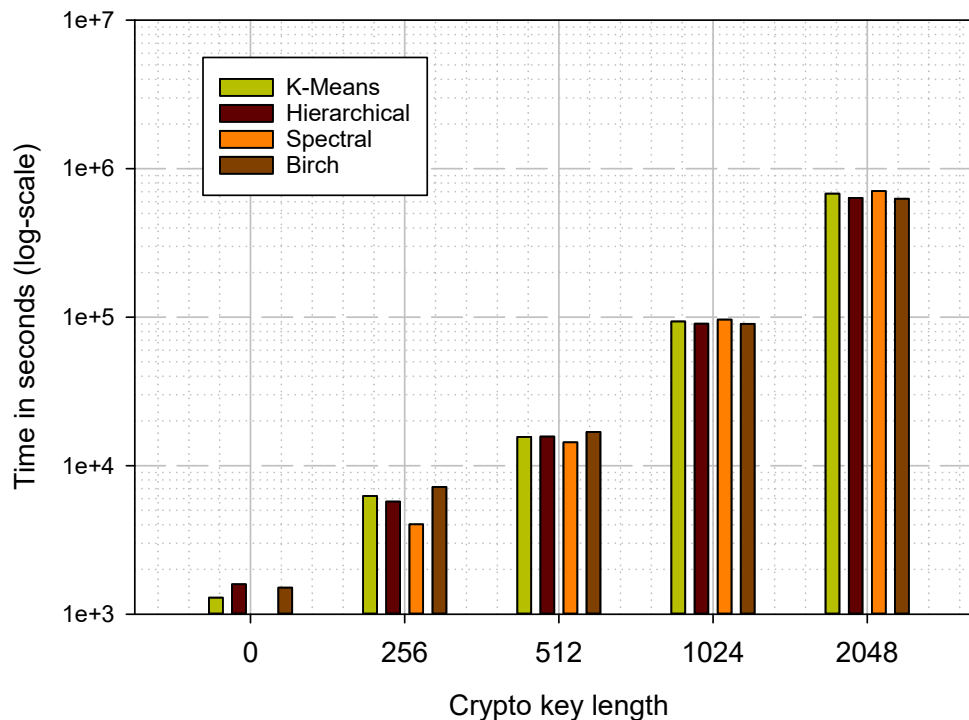


Figure 7. The training times on a logarithmic scale using the 5000 row dataset. The crypto key length with 0 refers to the plain domain execution time of the proposed method. This figure shows the nonlinear relationship between the increase in key size and the duration of algorithm training by using a logarithmic scale on y-axis.

6. Conclusions and Future Work

Currently, data privacy is one of the most crucial problems. Data come from many sources in online environments, including mobile phones, computers, and Internet of Things (IoT) tools and based on this data we can make various data analyses. As a result, we can improve the comfort of humanity. Due to the high volume of the data generated and collected by different sources, we have to conduct data analysis using cloud service providers, while taking into consideration that a mistrust of these providers as a well-known fact.

In our previous study, we represented the usage of the Paillier crypto system together with the classification algorithms. In this study, we have examined clustering algorithms, which are frequently used techniques in the field of machine learning. Privacy-preserving clustering algorithms are an important direction enabling knowledge discovery without requiring the disclosure of private data.

In this research, we applied four different clustering algorithms and their results are compared based on six different cluster evaluation metrics and their execution times. Each clustering algorithm obtained relatively similar results, but in the details they have differences.

The evaluation scores of our proposed privacy-preserving clustering models of the plain domain and encrypted domain are almost the same. Thus, the conversion from floating point numbers to integers creates only trivial metric differences for each clustering algorithm.

Our proposed cluster training model is a privacy-preserving method using the Paillier encryption system for outsourced sensitive datasets. The client builds a final clustering model with aggregation of each encrypted distance matrix calculated at each party. As a result, the final model is in the plain domain and some information such as cluster centroids are plain. If the client wants to share the model, then some information leakage may occur.

As a future work, in order to prevent data disclosure from the model, the model itself should also be encrypted using homomorphic encryption algorithms. In order to encrypt the model, the cluster centroids should also be encrypted. To allow the client to use the encrypted clustering model, a new model must be developed.

Author Contributions: Conceptualization was conceived by F.O.C., methodology, implementation, validation was done by I.A. and F.O.C. The manuscript's writing, review and editing by F.O.C., I.A., O.E. and S.Y.-Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Nasrabadi, N.M. Pattern recognition and machine learning. *J. Electron. Imaging* **2007**, *16*, 049901.
2. Tankard, C. What the GDPR means for businesses. *Netw. Secur.* **2016**, *2016*, 5–8. [[CrossRef](#)]
3. Edwards, S. Review of a medical illustration department's data processing system to confirm general data protection regulation (GDPR) compliance. *J. Vis. Commun. Med.* **2019**, *42*, 140–143. [[CrossRef](#)] [[PubMed](#)]
4. Simmons, G.J. Symmetric and asymmetric encryption. *ACM Comput. Surv. (CSUR)* **1979**, *11*, 305–330. [[CrossRef](#)]
5. Cuzzocrea, A. Privacy and Security of Big Data. In Proceedings of the First International Workshop on Privacy and Security of Big Data—PSBD-14, Shanghai, China, 3–7 November 2014. [[CrossRef](#)]
6. Verykios, V.S.; Bertino, E.; Fovino, I.N.; Provenza, L.P.; Saygin, Y.; Theodoridis, Y. State-of-the-art in privacy preserving data mining. *ACM Sigmod Rec.* **2004**, *33*, 50–57. [[CrossRef](#)]
7. Catak, F.O.; Mustacoglu, A.F. CPP-ELM: Cryptographically Privacy-Preserving Extreme Learning Machine for Cloud Systems. *Int. J. Comput. Intell. Syst.* **2018**, *11*, 33–44. [[CrossRef](#)]
8. Lindell, Y.; Pinkas, B. Privacy Preserving Data Mining. In *Advances in Cryptology—CRYPTO 2000*; Bellare, M., Ed.; Springer: Berlin/Heidelberg, Germany, 2000; pp. 36–54.
9. Chaudhuri, K.; Monteleoni, C. Privacy-preserving logistic regression. In *Advances in Neural Information Processing Systems 21*; Koller, D., Schuurmans, D., Bengio, Y., Bottou, L., Eds.; Curran Associates, Inc.: Dutchess County, NY, USA, 2009; pp. 289–296.
10. Agrawal, R.; Srikant, R. Privacy-preserving Data Mining. *SIGMOD Rec.* **2000**, *29*, 439–450. [[CrossRef](#)]
11. Xu, K.; Yue, H.; Guo, L.; Guo, Y.; Fang, Y. Privacy-Preserving Machine Learning Algorithms for Big Data Systems. In Proceedings of the 2015 IEEE 35th International Conference on Distributed Computing Systems, Columbus, OH, USA, 29 June–2 July 2015; pp. 318–327. [[CrossRef](#)]
12. Merugu, S.; Ghosh, J. Privacy-preserving distributed clustering using generative models. In Proceedings of the Third IEEE International Conference on Data Mining, Melbourne, FL, USA, 22 November 2003; pp. 211–218. [[CrossRef](#)]
13. Shokri, R.; Shmatikov, V. Privacy-Preserving Deep Learning. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS '15, Denver, CO, USA, 12–16 October 2015; pp. 1310–1321. [[CrossRef](#)]
14. Yang, Z.; Zhong, S.; Wright, R.N. Privacy-Preserving Classification of Customer Data without Loss of Accuracy. In Proceedings of the 2005 SIAM International Conference on Data Mining, Newport Beach, CA, USA, 21–23 April 2005; pp. 92–102. [[CrossRef](#)]
15. Emekci, F.; Sahin, O.; Agrawal, D.; Abbadi, A.E. Privacy preserving decision tree learning over multiple parties. *Data Knowl. Eng.* **2007**, *63*, 348–361. [[CrossRef](#)]
16. Li, P.; Li, J.; Huang, Z.; Li, T.; Gao, C.Z.; Yiu, S.M.; Chen, K. Multi-key privacy-preserving deep learning in cloud computing. *Future Gener. Comput. Syst.* **2017**, *74*, 76–85. [[CrossRef](#)]
17. Yi, X.; Zhang, Y. Privacy-preserving naive Bayes classification on distributed data via semi-trusted mixers. *Inf. Syst.* **2009**, *34*, 371–380. [[CrossRef](#)]
18. Secretan, J.; Georgiopoulos, M.; Koufakou, A.; Cardona, K. APHID: An architecture for private, high-performance integrated data mining. *Future Gener. Comput. Syst.* **2010**, *26*, 891–904. [[CrossRef](#)]
19. Wu, C.H.; Ouyang, C.S.; Chen, L.W.; Lu, L.W. A new fuzzy clustering validity index with a median factor for centroid-based clustering. *IEEE Trans. Fuzzy Syst.* **2015**, *23*, 701–718. [[CrossRef](#)]

20. Banfield, J.D.; Raftery, A.E. Model-Based Gaussian and Non-Gaussian Clustering. *Biometrics* **1993**, *49*, 803–821. [[CrossRef](#)]
21. Duan, L.; Xu, L.; Guo, F.; Lee, J.; Yan, B. A local-density based spatial clustering algorithm with noise. *Inf. Syst.* **2007**, *32*, 978–986. [[CrossRef](#)]
22. White, S.; Smyth, P. A spectral clustering approach to finding communities in graphs. In Proceedings of the 2005 SIAM International Conference on Data Mining, Newport Beach, CA, USA, 21–23 April 2005; pp. 274–285.
23. Rosenberg, A.; Hirschberg, J. V-measure: A conditional entropy-based external cluster evaluation measure. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), Prague, Czech Republic, 28–30 June 2007.
24. Samet, S.; Miri, A. Privacy-preserving back-propagation and extreme learning machine algorithms. *Data Knowl. Eng.* **2012**, *79-80*, 40–61. [[CrossRef](#)]
25. Paillier, P. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology—EUROCRYPT '99*; Stern, J., Ed.; Springer: Berlin/Heidelberg, Germany, 1999; pp. 223–238.
26. Min, Z.; Yang, G.; Shi, J. A privacy-preserving parallel and homomorphic encryption scheme. *Open Phys.* **2017**, *15*, 135–142. [[CrossRef](#)]
27. Barker, E.; Barker, W.; Burr, W.; Polk, W.; Smid, M.; Gallagher, P.D.; For, U.S. *NIST Special Publication 800-57 Recommendation for Key Management—Part 1: General*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2012.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).