Privacy Preserving Third Party Data Mining Using Cryptography

Thesis submitted in accordance with the requirements of
the University of Liverpool for the degree of Doctor in Philosophy by

NAWAL MOHAMMED ALMUTAIRI

February 2020

Dedication

To my dearest parents, and to my beloved husband.

Acknowledgements

This PhD thesis barely merits its name without the intensive effort and invaluable support of my supervisors and patience offered by my family members, to whom I would like to express my gratitude. This thesis would not have been possible without the immense knowledge, constant patience and valuable advise offered by my primary supervisor Prof. Coenen. His constructive criticism, gracious mentorship and research ideas have made the completion of my PhD possible. I have learnt more from him than I could ever express. Prof. Coenen, there are no words that express my sincere gratitude to you. I have been immensely lucky to work with you. I would like to extend my profound gratitude to my second supervisor Dr. Dures and PhD advisors Dr. Lisitsa, Dr. Jackson and Prof. Tuyls for their constructive feedback and suggestions. I would also like to express my sincerest gratitude to my parents, sibling and all my family for their constant support that kept me strong throughout this journey. I would especially like to express a heartfelt thank you to Abdullah and Malik who always stood beside me when this journey seemed most challenging. To my daughters for bringing joy to the family. Finally, I gratefully acknowledge the generous funding received from the Government of Saudi Arabia, especially the Saudi Arabian Cultural Bureau in London.

Abstract

The research presented in this thesis is directed at investigating and evaluating the usage of cryptography to provide secure data analysis using a third party. The motivation is the emergence Data Mining as a Service (DMaaS), which in turn has been motivated by cloud computing technology that provides the potential for reducing the operational cost of analysing data by utilising the storage and computing services provided by cloud service providers. DMaaS has also opened the door for collaborative data mining whereby multiple data owners pool their data for analysis, using a cloud provider offering DMaaS, to gain some mutual benefit. The challenge is for the data analysis to be conducted in a secure manner. Data privacy can be substantially preserved using cryptography. With the emergence of Homomorphic Encryption (HE) schemes encrypted data can, to an extent, be securely processed without decryption. However, current HE schemes have imposed constraints on the computation, both in terms of the arithmetic operations provided (not all operations required by data mining algorithms are supported) and computational overhead (multiplication can become very slow). Solutions that have been introduced in the literature include: (i) resorting to Secure Multi-Party Computation (SMPC) protocols or (ii) substantial data owner involvement whenever unsupported operations are required. In both cases, the amount of data owner participation is significant, calling into question the advantages that DMaaS has to offer. The research presented in this thesis asks the question "*Using cryptography is it possible to securely, effectively and efficiently delegate data analysis to a third party data miner while minimising any required interaction with data owners?*". The fundamental idea presented, so as to achieve secure DMaaS, is the idea of using a proxy for the data rather than the data itself. In particular to use the concept of distance matrices as the proxy. A range of distance matrix implementations are presented each of increasing sophistication. The utility of the data proxy idea is illustrated using a collection of proposed secure data clustering and classification algorithms that operate over encrypted data. The thesis also introduces several encryption schemes designed to address the limitations of existing schemes in the context of DMaaS. Throughout the thesis two distinctive DMaaS scenarios are considered, the single data owner scenario and the multiple data owner scenario. The proposed distance matrices directed at the single data owner scenario are: (i) Updatable Distance Matrices (UDMs), (ii) Encrypted Updatable Distance Matrices (EUDMs), (iii) Encrypted Distance Matrices (EDMs) and (iv) Secure Chain Distance Matrices (SCDMs); while the distance matrices directed at the multiple data owner scenario are: (i) Global EDMs (GEDMs) and Super SCDMs (SSCDMs). The proposed concepts, schemes and secure data mining algorithms were evaluated using two categories of data; UCI datasets and randomly generated synthetic datasets. The synthetic datasets were used to evaluate the scalability of proposed solutions by analysing the runtime as the data size increases. The evaluation was conducted to compare the operation of the proposed approaches with each other, and the relevant standard (insecure) algorithms. The evaluations considered the proposed approaches in

terms of: (i) the amount and complexity of the data owner participation in preparing data and participating when secure data mining was undertaken by the TPDM, (ii) efficiency of the secure data mining algorithms, (iii) accuracy of proposed approaches, (iv) the security and (v) the scalability in the case of collaborative data mining approaches. The accuracy was measured by comparing the operation of the proposed algorithms to that of standard algorithms operating over unencrypted data. The evaluations indicted that the proposed solutions reduced the data owner participation, compared to alternative approaches, while maintaining the effectiveness of the data analysis.

Contents

Illustrations

List of Figures

List of Tables

List of Algorithms

Notations

The following notations and abbreviations are found throughout the thesis:

CSPs Cloud Service Providers

DMaaS Data Mining as a Service

FHE Fully Homomorphic Encryption

HE Homomorphic Encryption

OPE Order Preserving Encryption

PHE Partial Homomorphic Encryption

PPDM Privacy Preserving Data Mining

PPE Property Preserving Encryptions

QID Quasi Identifier

QOs Query Owners

RSA Rivest-Shamir-Adleman

SMPC Secure Multi-Party Computation

STP Semi-honest Third Party

SWHE Some What Homomorphic Encryption

TCV Ten-fold Cross Validation

TPDM Third Party Data Miner

TTP Trusted Third Party

YMPP Yao's Millionaires Problem Protocol

Chapter 1

Introduction

1.1 Overview

Recent technological advances and digital innovations have exponentially increased the amount of data collected by enterprises of all kinds. This has created a storage and processing challenge; most enterprises have only limited storage and processing capabilities because of the resource involved [1]. Many data owners (enterprises) have consequently moved their data to "the cloud". Figure 1.1 shows an organisation view of the most important benefits that can be gained from adopting cloud storage capabilities as reported in a Microsoft TechNet survey [2]. Cloud storage also facilitates cloud computing, the ability to process data using the cloud service providers servers rather than servers local to the enterprise. Cloud computing, as an alternative to traditional local computing, offers advantages of: (i) high scalable storage and computational capacity, (ii) low cost (low infrastructure investment) and (iii) increase in the collaboration potential between cloud users.

Coupled with a rise in the usage of cloud storage and computing is a corresponding increase in the quantity of data that enterprises collect. This increase, in turn, is coupled with an increased desire, on behalf of enterprises, to apply the techniques of machine learning and data mining to their data. Many Cloud Service Providers (CSPs) thus also provide Data Mining as a Service (DMaaS) facilities. Examples include Microsoft Azure, Google Cloud Platform (GCP) and Amazon Web Services (AWS). More generically, throughout this thesis, any provider of DMaaS will be referred to as a Third Party Data Miner (TPDM) who may, or may not, also be a CSP. The DMaaS provided typically comprises a set of data analytics and data mining algorithms such as anomaly detection, data classification and clustering. The "studios" and "workbenches" provided frequently also provide easy to use "drag and drop" interfaces. However, there are significant concerns, on behalf of data owners, related to data privacy and security. This, in turn, has served to limit the adoption of DMaaS [3, 4]. This is evidenced by a recent report compiled by the Cloud Security Alliance (CSA)[1] which concluded that, despite the many security measures currently adopted by CSPs, 66.5% of enterprises continue to have reservations about cloud security in the context of the outsourcing of sensitive data [5]. These concerns have been reflected in the actions of legislative bodies that have enacted privacy legislation, such as the U.S. Health Insurance Portability and Accountability Act (HIPAA) [6] and the European Union's General Data Protection Regulation (GDPR) [7]. The work presented in this thesis is therefore motivated by the reservations that enterprises have expressed concerning the adoption of DMaaS, as

[1] A non-profit organisation that promotes research into best practices for securing cloud computing and the use of cloud technologies to secure other forms of computing.

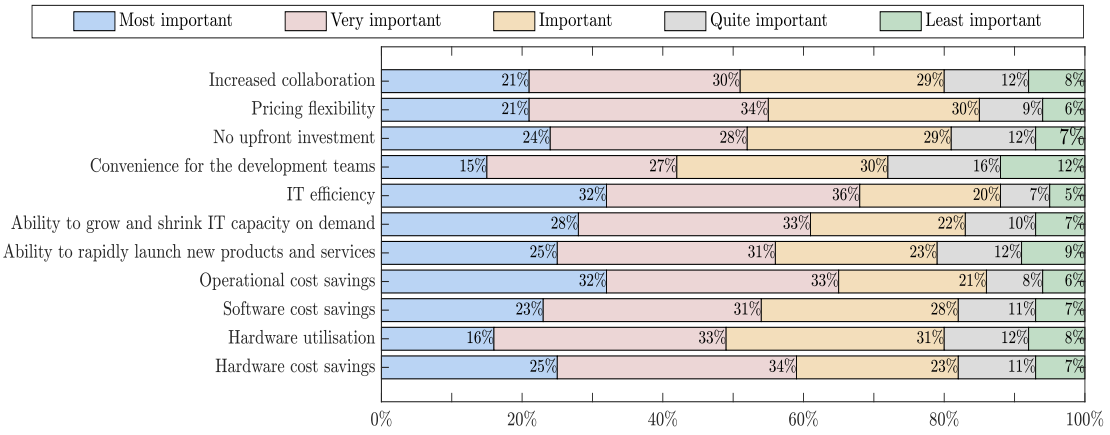


FIGURE 1.1: Statistics concerning the most important benefits of adopting cloud storage [2]

facilitated by current cloud storage and cloud computing capabilities, with respect to data privacy and security in the context of the requirements of frameworks such as HIPAA and GDPR. More specifically, the work presented is directed at an investigation into mechanisms to provide secure, efficient and effective third party data clustering and classification without compromising the accuracy of results.

The CSPs (TPDMs) who offer DMaaS are governed by contractual commitments under which the services are provided. However, the majority of CSPs do not guarantee the security and privacy in their Service Level Agreement (SLA) [8–10], a part of the contractual terms and conditions between CSPs and consumers (data owners). Many CSPs maintain data privacy and security using "*best practices*", such as: (i) the encryption of data when in transit between a CSP and data owners and when "at rest" in storage; (ii) providing data owners with permissions to configure data access controls, (iii) deletion of confidential data once data analysis is complete (this encompasses any data backups) and (iv) usage of Trusted Execution Environments (TEEs), a secure area within the main processor (the CPU) that serves to isolate the execution environment therefore preventing data from being accessed during processing [11]. However, using these best practices, there is no guarantee that the data will be completely secure, and data privacy preserved when data is manipulated, by TPDMs, in plaintexts form. TPDMs are vulnerable to external malicious attacks and internal misuse behaviours within the confines of a CSP. Therefore, encrypting the dataset in transit or at rest using the key of the CSP is also of concern.

Given the above, in addition to the best practices applied by the CSPs, data owners have adopted additional protection methods to preserve their data confidentiality such as data anonymisation [12] and the private cloud computing model [13]. Using data anonymisation, data owners remove confidential and deemed "*personal*" attributes before outsourcing the data to a TPDM. However, recorded incidents have shown that this method is insufficient and data cannot be 100% anonymised [14–16]. For example, Sweeney et al. [16] identified (by name) 84% of patients that had consented to participate in the Personal Genome Project (PGP) by "*cross-referencing*" the PGP dataset with publicly available voter registration archives. This allowed not only access to DNA data, but also other types of highly sensitive data present in the PGP dataset. The second method is to adopt the private cloud model that provides computing power as a service within a virtualised environment using an underlying pool of physical computing

resources which are only accessible by a single enterprise (data owner), therefore providing that enterprise with greater control and privacy. However, using a private cloud, the data owners not only have to invest in computing and storage resources, but also in the related software and maintenance activities which in many cases negates the reason for adopting cloud storage in the first place.

Data encryption provides a powerful potential alternative solution to the DMaaS data confidentiality problem [17]. Although the majority of encryption schemes prevent any manipulation of data once it is encrypted, thus precluding any form of DMaaS. There are a number of emerging encryption schemes, named Property Preserving Encryption (PPE) schemes [18], that maintain some of the properties of the plaintext data so that the cyphertext data can still be manipulated, although in a prescribed manner. One form of PPE is Homomorphic Encryption (HE) which allows simple mathematical operations over cyphertext without decryption. The precise nature of the mathematical operations supported by a HE scheme is dependant on the nature of the adopted scheme. The primary homomorphic properties supported by different HE schemes are limited to cypher addition ($\oplus$), cypher multiplication ($\otimes$) and in some cases multiplying cyphers by real numbers ($\circledast$). However, although HE schemes do partially address the security obstacles limiting the widespread adoption of DMaaS, the existing HE schemes identified in the literature do not provide all the mathematical operations required for any form of sophisticated DMaaS and typically their usage entails a large computational overhead. For example similarity calculation operations, required with respect to many data mining algorithms, are not supported.

The solutions that have been developed to date to address the limitations associated with the adoption of HE schemes can be broadly categorised (at least at a high level) as follows:

1. **Recourse to data owner:** The operations not supported by the selected HE scheme are conducted using unencrypted data by the data owner or data owners (see for example [19–24]).

2. **Multiple-CSPs model:** Use two (or more) non-colluding CSPs where one holds the encrypted data and the other holds the private key (an example can be found in [25]). In this context, the operations not supported by the adopted HE scheme will be delegated to the key holder CSP party who acts on behalf of a data owner or data owners [26]. Several methods have been adopted to preserve the data confidentiality and privacy for delegated operations.

3. **Secret sharing:** Use HE schemes that mathematically split a secret key among multiple semi-honest and non-colluding parties, such as the scheme presented in [27]. The TPDM will uses the HE properties of the adopted scheme to manipulate the data whilst the unsupported operations are collaboratively conducted using the party's shared secret key on behalf of the data owner (see for example [19, 28]).

4. **Secure Multi-Party Computation (SMPC):** Use well-established SMPC protocols [29–32] to securely calculate statistical values over data distributed across several data owners. A solution only applicable in the context of collaborative data mining. In this case, the TPDM is typically involved only as a mediator as in the case of [20, 33–37], or not used at all as in the case of [36, 38–40].

To date, most existing work falls into the first and last of the above categories, both entail a significant element of data owner participation. The degree of data owner participation is such that the only remaining reason for adopting a DMaaS facility is to

support collaborative data mining. In the case of a single data owners there seems little point in conducting third party data analysis if most of the work needs to be conducted by the data owners themselves (given that there is no HE scheme that provides all the mathematical operations that might be required by a DMaaS provider). The requirement for a number of semi-honest and non-colluding parties (in the remaining categories of solution) is also of concern, and for many data owners a security risk as the secret key cannot be revoked even when a party is deemed to be untrustworthy. Given the above, there is a widespread requirement for effective and accurate DMaaS solutions where data can be outsourced in encrypted form and mined in a secure manner by a TPDM without the need for significant interaction with data owners and without the need to expose secret keys to non-colluding parties. The need for appropriate encryption schemes and supporting frameworks to achieve this is therefore the central concern of this thesis. To address this issue this thesis proposed a number of encryption mechanisms, coupled with secure processes and protocols, that dramatically reduce, or in some case avoid, data owner participation with respect to data mining operations conducted by a TPDM. The thesis considers scenarios where data belonging to a single data owner is outsourced, and scenarios that involve two or more data owners who wish to conduct collaborative data mining.

The rest of this introductory chapter is organised as follows. In Section 1.2 the motivation for the research is discussed in further detail. The research question and associated research issues are then presented in Section 1.3. Section 1.4 outlines the research methodology adopted to address the research question and the associated issues. Section 1.5 summaries the research contributions followed, in Section 1.6, by a list of published work to date resulting from the work presented in this thesis. In Section 1.7, the structure of the remainder of the thesis is outlined. Finally, the chapter is concluded, in Section 1.8, with a brief summary.

1.2 Motivation

From the foregoing, the main motivation for the work presented in this thesis is the need for secure, scalable and effective (in terms of accuracy, cost and the complexity of data owner participations) mechanisms for third party data analysis. The fundamental idea presented in this thesis is, as already noted above, to use cryptography to eliminate current privacy and security concerns when delegating data analysis to a TPDM. The proposed cryptography methods, mostly founded on the use of distance matrix data proxies as a means of preserving data privacy, provide the following advantages over other Privacy Preserving Data Mining (PPDM) methods found in the literature:

- The cryptography has a mathematical foundation that can be used to prove the security of the individual proposed schemes and/or identify the potential attacks that can be instigated over any proposed solution [17].
- The mathematical properties preserved in the cyphertexts generated using the proposed schemes permit data manipulation (for example distance calculation and cluster centroid calculation) without loss of accuracy with respect to final data analysis results. This was not the case in other methods [41–44] which featured trade-offs between accuracy and security.
- The entire dataset can be encrypted without the requirement of deleting the private (or confidential) attributes as in the case of annonymisation PPDM methods [12].

Existing research directed at HE for secure data analysis, in the case of a single data owner scenario, has either featured extensive data owner participation and a corresponding communication overhead [21, 23, 24] or made use of a Trusted Third Party (TTP) [25, 45]. Both solutions entailed significant security concerns rendering them to be inappropriate with respect to many application domains. Furthermore, most of the existing work in the context of collaborative data mining (the multiple data owners scenario) relies on SMPC protocols that introduce a considerable computation and communication overhead with respect to data owners, and thus require the availability of a good local IT infrastructure. The disadvantages associated with this previous work provides further motivation for the work presented in this thesis.

It should also be noted that the work on secure data analysis presented in the literature has concentrated on providing a solution with respect to particular individual data mining algorithms that cannot easily be extended to support alternative data mining algorithms; in other words a generic solution. Therefore, in this thesis, the intuition is that any proposed new method directed at DMaaS, that does not feature the disadvantages associated with earlier work, should also support a range of data analysis algorithms; proposed solutions should feature "versatility".

From the preceding discussion, the work presented in this thesis, is motivated by the disadvantages associated with existing approaches to secure data mining concerned with mechanism to address the disadvantages of current HE schemes.

1.3 Research Questions and Related Issues

Given the above, the research question which this thesis seeks to address, is:

"Using cryptography is it possible to securely, effectively and efficiently delegate data analysis to a third party data miner while minimising any required interaction with data owners?"

The provision of an answer to this research question entailed the resolution of a number of subsidiary questions as follows:

1. What is the most appropriate HE scheme that satisfies the requirements for outsourcing data mining activities to a TPDM in the context of PPDM and the single data owner scenario?

2. Is HE the best form of cryptography to provide an effective solution to the PPDM problem?

3. Following on from 2, if HE is not the most appropriate form of cryptography, what is the most appropriate form of encryption required to provide a better solution?

4. What is the most effective way of supporting the required operations not supported by an encryption scheme (without decrypting the data) so as to, where necessary, minimise the number of data owner interactions?

5. Given potential solutions to above, what is the most appropriate mechanism, or mechanisms, for evaluating these solutions?

6. Can the proposed single data owner scenario solutions be extended to support scalable collaborative data mining (the multiple data owners scenario) while keeping the data owner participation at a minimum?

7. Is it possible to tailor (or modify) any proposed HE scheme so that it can support the operations required by a range of data mining algorithms, as opposed to only a small number of specific algorithms?

1.4 Research Methodology

To provide answers to the above listed research issues, and the overriding research question, the start point for the work presented in this thesis, was to review established HE schemes and their HE properties. The aim was to identify appropriate schemes that could, with some adaptation, provide a solution to a wide range of data mining algorithms by considering the following criteria: (i) schemes efficiency, (ii) scheme security and (iii) the supported HE mathematical properties. The Property Preserving Encryption (PPE) schemes were also reviewed to identify the potential properties that can be preserved in cyphertext and could therefore go some way to support PPDM. At the commencement of the research the potential of using PPE schemes instead of, or in conjunction with, HE was not clear as most of the research reported in the literature directed at PPDM was founded on the exclusive usage of HE schemes. From the review of existing work a number of schemes were identified to act as a foundation for further investigation.

One of the aims of the proposed research was to derive a generic (versatile solution) to the PPDM challenge encompassing a range of data mining algorithms. However, to limit the scope of the thesis, it was decided to commence the investigation with a focus on data clustering algorithms and then move on to data classification algorithms, particularly Back-Propagation Neural Networks (BPNNs). Regardless of the considered algorithm, the key issue was how to achieve the envisioned DMaaS in such a way that the entire data analysis process (or at least most of it) was delegated to the TPDM. At the commencement of the research, the focus was on the single data owner scenario where a single data owner outsourced their data clustering to a TPDM as shown in Figure 1.2(a). The first phase of the research presented in this thesis was therefore directed at investigating and evaluating a series of single data owner secure data cluster techniques that, with respect to existing work, reduced data owner participation and minimised the communication overhead between data owner and the TPDM, both while maintaining the accuracy of final data analysis results. The proposed techniques were evaluated using the established k-Means [46], Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [47] and Nearest Neighbour Clustering (NNC) [48] algorithms. The research was then extended by considering the multiple data owners scenario in terms of collaborative PPDM [49]; the idea being to enable analysis over large datasets supplied by a number of data owners, so as to gain some mutual advantage (as illustrated in Figure 1.2(b)). The preliminary idea was to extend the single data owner solutions by involving multiple data owners. To this end, the previously proposed schemes had to be extended to support multiple data owner data privacy preservation.

The final phase of the adopted research methodology was to consider data classification. Two types of classification algorithm were considered: kNN data classification and Back-Propagation Neural Networks (BPNNs). The work commenced by considering the application of the previously proposed secure clustering techniques to support classification. The initial work directed at secure BPNN classification highlighted a further challenge with respect to non-linear activation function estimation over encrypted data. Another challenge, when a collaborative query is allowed, was how to best allow authorised parties to securely query an encrypted classification model.

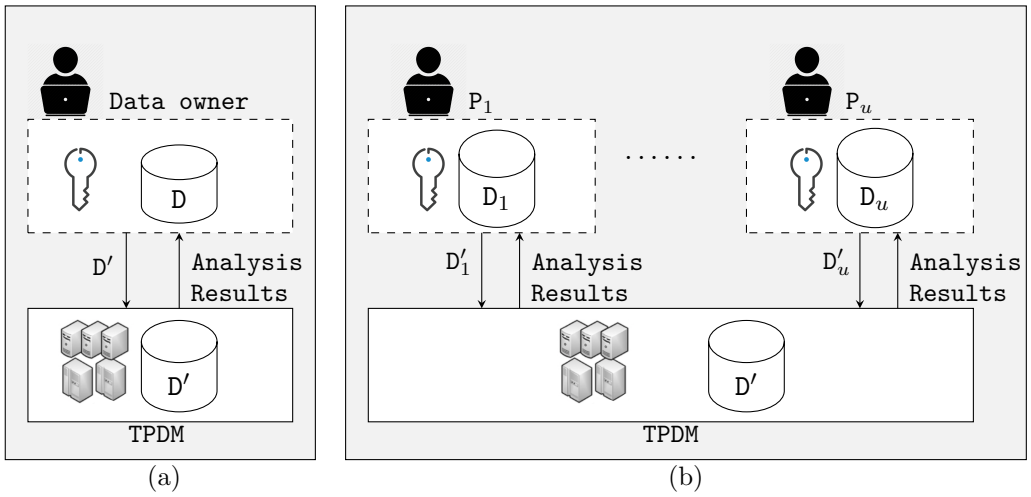


FIGURE 1.2: Data owner scenarios considered in this thesis; (a) the single data owner scenario where data belonging to a single data owner is outsourced to a TPDM for analysis and (b) the multiple data owners scenario featuring collaborative outsourcing of data mining activities where $\mathtt{D}' = \cup_{i=1}^{i=u} D'_i$ belongs to different participating parties p_1 to p_u

In all cases the proposed solutions (algorithms, concepts and protocols) were evaluated and (where possible) compared with respect to: other mechanisms proposed in the literature and to standard algorithms that operated over unencrypted data. The evaluation was conducted using synthetic datasets and a number of benchmark datasets taken from the University of California Irvine (UCI) machine learning repository [50], these were selected so that datasets of a variety of sizes and different numbers of classes could be considered. The evaluation was performed to assess: (i) the amount and complexity of data owner participation, (ii) the accuracy (effectiveness) of the data mining, (iii) the efficiency (the runtime complexity), (iv) the level of security and (v) the scalability of the proposed solutions.

The reasons for choosing the selected algorithms (k-Means, DBSCAN and NNC data clustering, kNN data classification and BPNN) were as follows:

- The selected algorithms had wide application and are currently offered by many DMaaS service providers although in an insecure manner.
- Clustering algorithms that involve distance comparison against threshold values entail a security challenge in the context of collaborative data mining; as other participants can use the knowledge of their own data and results of comparisons (when the clustering is undertaken) to instigated overlapping attacks that could help to estimate the record attribute values of the private data belonging to other participants.

1.5 Contributions

The work presented in this thesis makes a number of contributions with respect to PPDM and cryptography that can be summarised as follows (the relevant chapter where the contribution is presented is included in parenthesis):

1. Improvement in data security by obviating the need for sending data in any form to TPDMs and/or sharing it with other participating parties using the concept of

Super Secure Chain Distance Matrices (SSCDMs) and the idea of virtual record lists (Chapter 8).

2. Reducing the amount of data owner participation when undertaking secure data mining operations, and in some cases avoiding it entirely, when the data mining is undertaken by a TPDM using the concept of distance matrices (Chapters 4 to 8).
3. Introducing the idea of Cryptographic Ensembles that comprise a HE scheme and a bespoken OPE scheme, therefore providing a solution to secure data clustering involving distance comparison (Chapter 5).
4. Six secure data clustering algorithms, in the context of the single data owner scenario, that operated over encrypted data, without requiring decryption and without delegating keys to non-colluding parties:
   (a) Secure k-Means using Updatable Distance Matrices (UDMs) (Sk-Means) (Chapter 4).
   (b) Double Blind Secure k-Means using Encrypted UDMs (EUDMs) (DBSk-Means) (Chapter 5).
   (c) Double Blind Secure Nearest Neighbour Clustering (DBSNNC) using Encrypted Distance Matrices (EDMs) (Chapter 5).
   (d) Sk-Means, SNNC and SDBSCAN using Secure Chain Distance Matrices (SCDMs) (Chapter 7).
5. Four Secure collaborative data clustering algorithms, addressing the multiple data owners scenario, that operated over encrypted data without decryption and without sharing data between participants or delegating keys to non-colluding parties:
   (a) Secure DBSCAN (S-DBSCAN) and Secure NNC (S-NNC) using Global EDMs (GEDMs) (Chapter 6).
   (b) Secure DBSCAN (SDBSCAN) and Secure NNC (SNNC) using Super SCDMs (SSCDMs) (Chapter 8).
6. A Secure Neural Network algorithm (SecureNN) that facilitates model training and query classification/prediction with only very limited data owner participation (Chapter 9).
7. A novel Order Preserving Encryption scheme, the Multi-Users Order Preserving Encryption (MUOPE) scheme, that supported collaborative data clustering and facilitates preservation of the ordering of data distributed across multiple data owners (Chapter 6).
8. The Frequency and Distribution Hiding OPE (FDH-OPE) scheme, an amalgamation of two order preserving encryption schemes, the nonlinear order preserving scheme [51] and Zheli et al.'s scheme [52], each of which facilitates information hiding and the reduction of information leakage (Chapter 5).
9. A "dimensionality reduction" algorithm that addressed the scalability issue of the exponentially increasing size of cyphertexts with the application of each multiplication operation as in the case of Liu's fully HE scheme (Chapter 9).
10. The Modified Liu's Scheme (MLS) that supports secure data comparison while preserving the same HE mathematical properties as Liu's original scheme [53] (Chapter 9).

11. Scalable collaborative data clustering that allows a very large number of parties (multiple data owners) to conduct data analysis over the union of their data without sharing private data and without recourse to the expensive SMPC protocols (Chapters 6 and 8).

12. The Secure Query Cyphering (SQC) protocol that preserves a query record's privacy, while at the same time preserving the data owner key confidentiality when classifying the query record using a proposed Secure kNN (SkNN) algorithm (Chapter 7).

1.6 Publications

The work presented in this thesis has been the subject of a number of refereed publications. Extended and revised versions of a number of published conference papers have been submitted to journal special issues and, at time of writing, some of these are still under reviewing. The published and submitted papers are itemised below, each with a short summary of the paper highlighting its significance with respect to this thesis. In each case a reference to the chapter where the material appears is also given.

Journal Papers

1. Nawal Almutairi, Frans Coenen and Keith Dures (2018): *Cryptography for Secure Third Party Data Clustering Using an Encrypted Updatable Distance Matrix, Submitted to: Transactions on Large Scale Data and Knowledge Centered Systems.* Journal article comprising an extended and revised version of conference paper (1) (see below). Usage of HE and OPE in secure data clustering is introduced and evaluated using Encrypted Updatable Distance Matrices (EUDMs). The proposed solution addresses the issue of the potential of the reverse engineering of the original dataset that was a feature of the UDM concept published earlier, while at the same time significantly reducing the data owner participation. The work presented in this paper is included in Chapter 5.

2. Nawal Almutairi, Frans Coenen and Keith Dures (2019): *Secure Third Party Data Clustering Using SecureCL, Φ-Data and Multi-User Order Preserving Encryption, Submitted to: Expert Systems: the Journal of Knowledge Engineering.* Journal paper that presents the Secure CLustering (SecureCL) collaborative data clustering mechanism founded on the idea of the Super Secure Chain Distance Matrix (SSCDM) and two secure clustering algorithms; the SNNC and SDBSCAN. The SecureCL operates over the SSCDM data proxy, without requiring access to the original data (in any form), and without the requirement of involving any data owner participation while SecureCL is undertaken by a TPDM. Three forms of data partitioning (distribution) are considered; horizontal, vertical and arbitrary data partitioning. The content of this paper was used as the foundation for work presented in Chapter 8.

3. Nawal Almutairi, Frans Coenen and Keith Dures (2019): *A Cryptographic Ensemble for Secure Third Party Data Analysis: Collaborative Data Clustering Without Data Owner Participation, Accepted for publication in Data and Knowledge Engineering (DKE).* The work presented in this paper was an extended version of conference paper (2) (see below). The paper introduces the Cryptographic Ensemble, Global Encrypted Distance Matrices (GEDM) and Multi-Users OPE (MUOPE),

that facilitate secure collaborative data clustering. The proposed solution was extensively evaluated in the context of NNC and DBSCAN clustering. The content of this paper was used with respect to the work presented in Chapters 5 and 6.

4. Nawal Almutairi, *Frans Coenen and Keith Dures (2019): SecureNN: A Third Party Privacy Preserving Neural Network With Back-Propagation Learning, to be submitted to: Transactions on Machine Learning and Data Mining.* The paper introduces the Modified Liu's Scheme (MLS) that addresses the problem of the exponential increasing size of cyphertexts in Liu's original HE scheme after encrypted multiplication $\otimes$ has been applied, and also presents a mechanism that serves to produce order preserving cyphers. An illustrative secure neural network using back-propagation learning algorithm, that operates over MLS, was presented and evaluated using UCI datasets. Chapter 9 uses material from this paper.

5. Nawal Almutairi, *Frans Coenen and Keith Dures (2019): Secure Outsourced kNN Data Classification Over Encrypted Data Using Secure Chain Distance Matrices, Submitted to: Communications in Computer and Information Science (CCIS).* The paper extends conference paper (3) (see below). The concept of Secure Chain Distance Matrices (SCDMs) is evaluated in the context of the kNN data classification and the kNN query process. The paper also presents a protocol for securely encrypting the query belonging to a query owner while maintaining the privacy of the data owner encryption key. The content presented in paper has contributed to the material presented in Chapter 7.

Conference Papers

1. Nawal Almutairi, *Frans Coenen and Keith Dures (2017): K-Means Clustering Using Homomorphic Encryption and an Updatable Distance Matrix: Secure Third Party Data Clustering with Limited Data Owner Interaction, 19th International Conference on Big Data Analytics and Knowledge Discovery (DaWaK 2017), pp. 274-285. Springer International Publishing.* This conference paper reports on the results of the UDM concept applied in the context of the Sk-Means algorithm. The idea was to encrypt the data belonging to a single data owner, using Liu's scheme, and use the associated homomorphic properties to calculate distances and cluster centroids on each clustering iteration. The UDM served to guide the TPDM to determine the data similarity between records and cluster centroids. The UDM was updated on each iteration using only very limited data owner participation. A criticism of the approach was that it might support re-engineering of the original data given that a UDM is, in essence, simply a large collection of linear equations. The ideas presented in this paper contributed to significantly reducing the data owner participations while maintaining the accuracy of final clustering results. The work presented in this paper, and results from the evaluation, are included in Chapter 4.

2. Nawal Almutairi, *Frans Coenen and Keith Dures (2018): Third Party Data Clustering Over Encrypted Data Without Data Owner Participation: Introducing The Encrypted Distance Matrix., 20th International Conference on Big Data Analytics and Knowledge Discovery (DaWaK 2018), pp. 163-173. Springer International Publishing.* This paper built on the earlier conference paper, listed above, by considering the issue of a potential attack, founded on reverse engineering processes, given that the UDM discloses distances between records in terms of a set of linear

equations. Two encryption schemes were used; a HE scheme and a proposed Frequency and Distribution Hiding Order Preserving Encryption (FDH-OPE) scheme. The proposed FDH-OPE was an amalgamation of two existing encryption schemes so as to provide an adequate level of security. The solution proposed in this paper is presented in Chapter 5 along with the results and evaluation.

3. Nawal Almutairi, *Frans Coenen and Keith Dures (2018): Data Clustering Using Homomorphic Encryption and Secure Chain Distance Matrices, 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2018), pp. 41-50. Springer International Publishing.* This conference paper built on the UDM concept with the aim of reducing the memory requirement and hence improving the scalability. The SSCDM concept was proposed which substantially reduced the number of elements in a UDM, and consequently the required memory resource. The reported evaluation demonstrated that the SCDM concept, and the non-deterministic properties of the scheme (presented in Chapter 5), led to small differences between standard algorithms and the proposed secure algorithms (Sk-Means, SNNC and SDBSCAN). However, the difference is not significant and positively served to reduce information leakage. The SCDM concept, the associated secure data clustering algorithms and obtained evaluation results are included in Chapter 7.

4. Nawal Almutairi, *Frans Coenen and Keith Dures (2018): Secure Third Party Data Clustering Using Φ-data: Multi-User Order Preserving Encryption and Super Secure Chain Distance Matrices, 38th BCS SGAI International Conference on Artificial Intelligence (BCS SGAI 2018), pp. 3-17. Springer International Publishing.* **Winner of the British Computer Society Specialist Group on AI (BCS-SGAI) prize for best technical paper.** This paper introduced the concept of SSCDMs, a proxy for real data, directed at secure collaborative data clustering. This idea served to obviate the need for outsourcing the real data in any form, or the sharing of real data with participants. The data clustering was delegated entirely to the TPDM. Three different data partitionings (distribution formats) were considered and three associated "binding" processes to build a SSCDM for data belonging to different data owners. The work in this paper is included in Chapter 8.

1.7 Thesis Structure

The rest of this thesis is organised as follows:

Chapter 2: Privacy Preserving Data Mining Background and Related Work. This chapter presents the background and relevant literature to the work presented in this thesis. The chapter commences with some definitions and a presentation of the formal notations used throughout the thesis. This is followed by a review of existing PPDM techniques and the associated security considerations in the context of both the single data owner and the multiple data owner scenarios.

Chapter 3: Cryptography Fundamentals and Preliminaries. The fundamentals and associated preliminaries of Homomorphic Encryption (HE) are presented in this chapter by considering the basic notation for these scheme, the HE mathematical properties they possess, their limitations and their security. The chapter also introduces, in further detail, the Liu [53] and Paillier [54] schemes which are the foundation for schemes and solutions presented later in the thesis.

Chapter 4: Updatable Distance Matrices. This chapter considers the first proposed approach directed at secure outsourced data clustering founded on the idea of Updatable Distance Matrices (UDMs). This chapter also presents the Secure k-Means (Sk-Means) clustering algorithm which operates over encrypted data and is founded on the UDM concept. The chapter presents a mechanism for updating the UDM, whenever this is required, with very limited data owner participation. The proposed approach is evaluated using benchmark datasets taken from the UCI machine learning repository.

Chapter 5: Encrypted Updatable Distance Matrices. The chapter introduces the idea of the Cryptographic Ensemble and Encrypted (Updatable) Distance Matrices (EUDMs/EDMs) which address the problem of the potential reverse engineering aspect of original data when using UDMs. The Cryptographic Ensemble comprises a HE scheme and a bespoken OPE scheme (FDH-OPE), which in turn is an amalgamation of two existing OPE schemes. The proposed approach is evaluated in the context of two clustering algorithms; Double Blind Sk-Means (DBSk-Means) and Double Blind Secure Nearest Neighbour Clustering (DBSNNC). Extensive evaluation is reported using the UCI datasets used earlier. The chapter also includes a worked example of the DBSk-Means algorithm using the Cryptographic Ensemble and the EUDM concept.

Chapter 6: Global Encrypted Distance Matrices. The chapter presents fully outsourced collaborative data clustering using the Cryptographic Ensemble and the Global EDM (GEDM), the first collaborative data clustering approach (the multiple data owners scenario) considered in this thesis. The chapter presents the Multi-Users OPE (MUOPE) scheme designed to facilitate data order preservation for data belonging to multiple users/sources. The chapter also presents the Pooling algorithm that allows multiple data owners to "pool" their EDM, with respect to horizontally partitioned data, so as to conduct collaborative data clustering. Two clustering algorithms are considered using the proposed GEDM and the Cryptographic Ensemble; Secure NNC (S-NNC) and Secure DBSCAN (S-DBSCAN). An extensive evaluation concerning the performance, security and scalability of the algorithms is presented with respect to randomly generated synthetic data and UCI datasets.

Chapter 7: Secure Chain Distance Matrices. In this chapter, secure outsourced data clustering and secure data classification are considered. The algorithms are founded on the concept of Secure Chain Distance Matrices (SCDMs) and Cryptographic Ensemble (as presented in the previous chapter). SCDMs dramatically reduce the required memory resources compared to when UDMs or EUDMs/EDMs are used. The Secure k-Means (Sk-Means), Secure DBSCAN (SDBSCAN), Secure NNC (SNNC) and Secure kNN (SkNN) classification algorithms are also introduced in this chapter. Particular challenges are considered associated with kNN query classification concerning queries that belongs to authorised Query Owners (QOs), namely: (i) preserving data owner key confidentiality when encrypting a QO's query, (ii) determining the similarity between the outsourced datasets and the new query records without involving the data owner or QO in the process and (iii) making the query process controllable by the data owner but with a minimum data owner participation. The chapter thus also presents the Secure Query Cyphering (SQC) protocol and secure binding algorithms. The presented evaluation again uses random synthetic datasets and UCI datasets.

Chapter 8: Super Secure Chain Distance Matrices. The chapter introduces the Super Secure Chain Distance Matrix (SSCDM) concept that obviates the need for outsourcing or sharing data in any form (encrypted or in plaintext form) to a TPDM. The concept of SCDMs, presented in Chapter 7, is extended by considering collaborative data mining over data distributed among different data owners. Three different "Binding" processes are introduced in this chapter each coupled to a different data partitioning scenario: horizontal, vertical and arbitrary data partitioning. This chapter also introduces the idea of virtual record lists and the possibility of updating the SSCDM by allowing record deletion and/or addition. Finally the chapter provides detail concerning the adoption of the proposed approach in the context of Secure Clustering (SecureCL) that encompasses NNC and DBSCAN. Evaluation is presented with respect to both random synthetic datasets and UCI datasets.

Chapter 9: Secure Neural Network Using Modified Liu's Scheme. This penultimate chapter presents the SecureNN approach that allows model training and prediction to be securely delegated to a TPDM with minimal data owner participation. The chapter also presents the Modified Liu's Scheme (MLS), a modification of Liu's original HE scheme to address the computational complexity of model learning; a complexity caused by the increasing size of cyphertexts through the continuous application of multiplication operations ($\otimes$) as the learning progresses. The proposed MLS, as far as the author is aware, is the first fully HE scheme that also provides an order preserving feature. The results from an extensive evaluation using the UCI datasets and random synthetic datasets are presented.

Chapter 10: Conclusion and Future Work. Chapter that concludes the thesis with a summary of the presented work, the main findings regarding the research question and the associated subsidiary questions and some discussion concerning possible future research directions.

1.8 Summary

This chapter has introduced the background and main ideas underpinning the research presented in this thesis. In particular the chapter has presented the research motivation, the research question, the adopted research methodology and the main contributions of the thesis. The next chapter (Chapter 2) provides further background to the work presented and a literature review designed to provide more detail regarding existing work and previous research concerning the research presented in this thesis.

Chapter 2

Privacy Preserving Data Mining Background and Related Work

2.1 Introduction

As noted in Chapter 1, the work presented in this thesis is directed at a cryptographic approach for providing privacy preserving third party data clustering and classification. Therefore, there are two aspects of research which, from the perspective of this thesis, are important to review: (i) the previous work related to PPDM and the adopted privacy preservation techniques, in particular those that can be adopted when a TPDM is involved; and (ii) the previous work on cryptography and encryption schemes which provide the potential for secure data analysis without first decrypting the data. This chapter will thus provide the necessary background on the PPDM techniques, security fundamentals and potential attacks that can be instigated when data analysis is delegated to a TPDM. The following chapter, Chapter 3, will present the necessary background concerning cryptography with emphasis on HE and PPE schemes.

In the introduction to this thesis it was noted that data mining is a well-established research field that aims to extract useful information by exploration and analysis of data. The resources facilitated through cloud computing have allowed businesses to reduce the operational cost of such processing by outsourcing the data and delegating any consequent data analysis to a TPDM who provides DMaaS. It has also opened the door for collaborative data mining where a number of data owners pool their data for analysis so as to gain some mutual advantages. However, the nature of the data mining process and the associated legal data security requirements, have raised privacy and security concerns when a third party is involved. This has instigated the emergence of the research domain of Privacy Preserving Data Mining (PPDM) [55, 56] that aims to provide techniques to allow secure data mining by attempting to achieve a balance between data utility and data privacy preservation.

The remainder of this chapter is structured as follows. Section 2.2 considers what is meant by the phrase "data privacy". The privacy requirements for individual and collaborative data analysis, the scenario where a single data owner outsources their data analysis to a TPDM and the scenario where multiple data owners mine their data collaboratively, are then considered in Section 2.3. In Section 2.4, a comprehensive review of PPDM techniques is presented. The section includes material on data modification, SMPC, secret sharing and HE techniques; coupled with discussion of the associated privacy measures, potential attacks and security vulnerability of each technique. The adopted security model in the context of work presented in the main chapters of this

thesis and the possible security threats that might be instigated, are discussed in Section 2.5. The chapter is concluded with a short summary in Section 2.6.

2.2 Data Privacy

Data privacy, confidentiality and security, are used in the literature in an interchangeable manner. Given the nature of the work presented in this thesis it is important that a single definition of the term data privacy is identified. Many definitions of the concept of data privacy found in the literature are expressed in the context of a particular research field to which the definition is applied [57]. In the context of data mining, data privacy is defined in [58] as "*getting valid data mining results without learning the underlying data values*" and in [59] as "*prevention of unwanted disclosure of information when data mining is performed on aggregate results*". These definitions highlighted: (i) the dilemma of balancing privacy preservation with maintaining the effectiveness of the data mining results and (ii) the importance of the "unwanted" disclosure of underlying data. The concept of data privacy is also dependent on differing perspectives, including individual desires and expectations, legislative perspectives and subjective ethical considerations (which are all subject to change over the time). The above has led to different understandings of what the phrase data privacy actually means, and by extension what "unwanted disclosure" is understood to mean [60].

A start point for establishing a definition with respect to this thesis is to consider the nature of the individual data attributes that may be included in a dataset D. These can be classified as follows:

1. Explicit Identifiers (EIDs): Attributes that directly link to a subject (which might be a person or enterprise) such as name, national identification number or phone number.
2. Quasi Identifiers (QIDs): Attributes which individually are not EIDs, but which jointly may link a data record in D to a specific subject, for example date of birth, post code and gender [61].
3. Sensitive Attributes: Attributes which hold information, such a medical condition, which will cause embarrassment or in some way be detrimental to an individual if disclosed.
4. Other: None of the above; attributes that create no problem if revealed even to untrustworthy parties.

It has been suggested that the last category of data can be disclosed without any consequence, whilst the remaining categories need to be kept secure and confidential. In other words the suggestion is that data can be split into "confidential" and "none-confidential" attributes and PPDM techniques only applied to confidential data attributes (an example for this argument and approach can be found at [62]). However, given the changing regulatory landscape and the changing popular perception of what data privacy is, it is argued in this thesis that all data should be considered to be confidential irrespective of the nature of the data. This is a view for which supported can also be found in the literature, see for example [63]. The definition of data privacy adopted in this thesis, and one of the central ideas presented in this thesis, is that all data is private and therefore any outsourced data should be encrypted in its entirety, analysed in its encrypted form, and that the analysis results should be maintained in an encrypted form accessible only to the data owner or owners.

2.3 Privacy Requirements for Individual and Collaborative Data Analysis

In Chapter 1 it was emphasised that two different data owner scenarios have been considered in this thesis. The single data owner scenario and the multiple data owners scenario. These can also be referred to as the individual data mining/analysis and collaborative/shared data mining/analysis scenarios, respectively. The data owner (owners) in both cases is (are) the data holder (holders). In this section, the two scenarios are formally presented and their data privacy preservation requirements, in the context of the definition of data privacy presented in Section 2.2 above and with respect to the rest of this thesis, are defined. In the context of the multiple data owners scenario different forms of data distribution are also considered.

2.3.1 Single Data Owner Scenario

The single data owner scenario refers to the situation where a single data owner outsources their private dataset D to a TPDM for analysis. This scenario is as depicted earlier in Figure 1.2-(a) in Chapter 1. The aim is to reduce the operational cost of conducting the data mining process in-house by harnessing the computational power available to the TPDM (typically a cloud computing provider). The data owner instructs the TPDM to perform the desired data analysis on their behalf, over the outsourced data, by specifying the nature of the desired analysis and the associated parameters. The TPDM performs the desired analysis and then sends the results back to the data owner. The ideal privacy preservation requirements in this case are that:

1. The TPDM must not have any direct access to the unencrypted outsourced dataset.
2. Any intermediate results, such as cluster centroids, actual distances between records, results of neural network activation functions and neural network weights and biases, must not be revealed to the TPDM.
3. The data owner is the only party to have access to the data mining results and/or developed model.

2.3.2 Multiple Data Owners Scenario

The multiple (distributed) data sources scenario refers to the scenario where the data $D = \{D_1, \ldots, D_u\}$ is distributed across multiple data owners $P = \{p_1, \ldots, p_u\}$ who wish to share their data in the context of some data mining enterprise, for example the generation of a shared classification model or cluster configuration. This scenario is depicted in Figure 1.2-(b) in Chapter 1. It is argued that by sharing data a better data mining result is produced than that which might have been obtained if only local data was used. This scenario is therefore also referred to as the collaborative data mining scenario. In terms of data privacy, collaborative data mining clearly needs to be conducted in a secure manner [49]. In this case the ideal privacy preservation requirements, in the context of this thesis, will be as follows:

1. The privacy of the data belonging to each participating party p_i must be preserved with respect to the TPDM and all other participants.
2. Intermediate results produced when the data mining is in progress must not be revealed to the TPDM or any other parties.

3. Each participant only receives the data analysis results of their own data (not the other participants results and not TPDM).

Additional privacy preservation requirements are imposed in the context of collaborative query service provision where authorised parties use a shared (classification or prediction) model to label additional private data records. Figure 2.1 shows the system model for this scenario. The figure shows a number of "data owners", one data owner whose data has been used to create a data model, a number of additional data owners, identified as authorised Query Owners (QOs) and a TPDM. The idea in the context of this thesis, and the above identified requirements, is that the data is outsourced to the TPDM in encrypted form (D'), using a key belonging to the data owner, who builds the desired model. Therefore for any query to be processed correctly the query record needs to be encrypted using the same key used to encrypt the data D. In this case, given the previous multiple data owners privacy preservation requirements, the following should be added:

1. The data owner private key should be kept confidential so that QOs do not have access to it.
2. The secure collaborative querying process must be controllable by the data owner but with minimum data owner participation (the point of DMaaS is that the data owner does not want to be involved in the actual analysis or query processing).
3. QOs must be unable to launch queries without being authorised to do so by the data owner.
4. The secure collaborative query process must preserve the privacy of the outsourced data from the QOs.
5. The privacy of the query record, and derived predicted label/value, must be preserved with respect to the TPDM and the data owner or data owners.

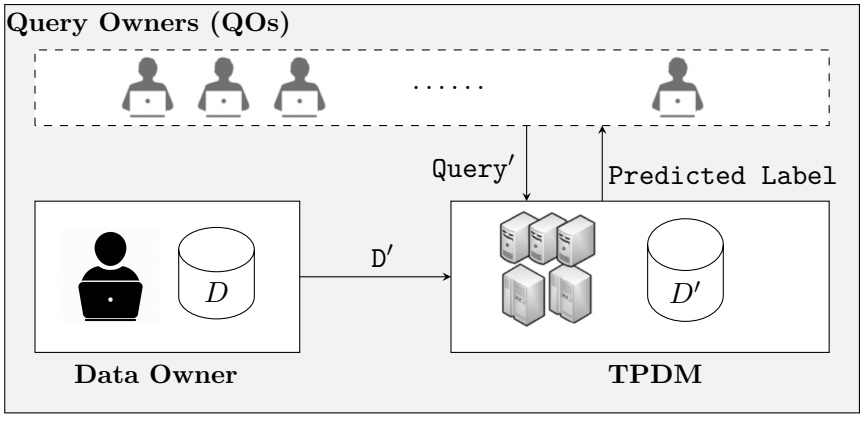
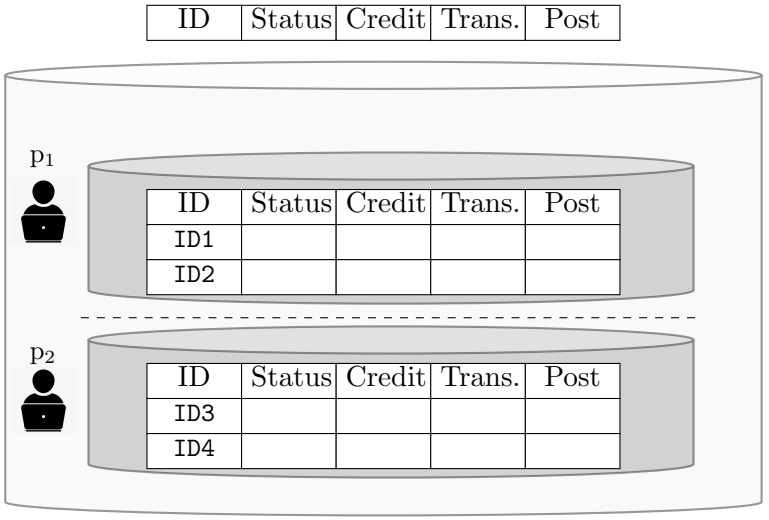


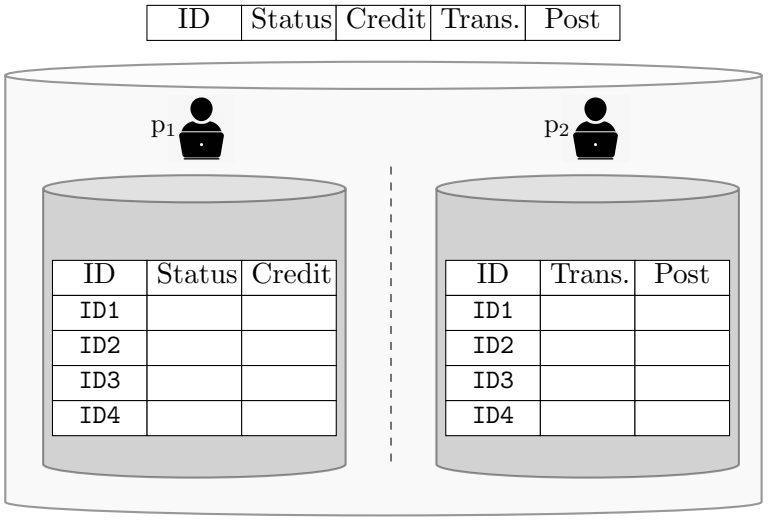
FIGURE 2.1: The collaborative query service system model

Data distributed across multiple data owners, in the context of a collaborative data mining scenario, can be partitioned in different forms. Three different forms of data partitioning can be identified from the literature; horizontal [64], vertical [65] and arbitrary. Horizontal data partitioned is where the participating parties conform to the same set of attributes, A, but each holds different records [64]. In other words, the global dataset

D is decomposed into "horizontal" segments each belonging to a single party. Figure 2.2-(a) shows an example of a dataset that has been horizontally partitioned across two parties. All the participating parties have the same data schema that comprises ID, Status, Credit, Trans. and Post. Vertical data partitioning is where the participating parties conform to the same set of records, but each holds different attributes derived from a global set of attributes A [65]. In other words, the global dataset D is decomposed into "vertical" segments each belonging to a single party. Figure 2.2-(b) shows an example of a dataset that has been vertically partitioned across two parties. In this case, each participant has a different data schema but all the collected data is referenced to the same set of entities. The arbitrary data partitioning generalises the vertical and horizontal partitioning cases and thus there is no schema agreed across parties for the collected data.



(a)



(b)

FIGURE 2.2: Horizontal (a) and vertical (b) data partitioning scenarios

2.4 Privacy Preserving Data Mining Techniques (Previous Work)

Over recent years, the problem of Privacy Preserving Data Mining (PPDM) has become prominent due to the increasing business need to outsource data analysis and for share/collaboratively data analyses. Consequently many techniques have been proposed designed to preserve data privacy while at the same time maintaining data utility when data analysis needs to be securely conducted in a collaborative setting. Some of these techniques are directed at the single data owner scenario (the individual data mining scenario), while others are only applicable to the multiple data owner scenario (the collaborative data mining scenario). In this section, a review of existing PPDM techniques is presented. These techniques can be categorised as follows: (i) data modification, (ii) Secure Multi-Party Computation (SMPC), (iii) secret sharing and (iv) Homomorphic Encryption (HE). Each is discussed in turn in the following four sub-sections, Sub-sections 2.4.1 to 2.4.4. Each sub-section will also discuss the relevant privacy measures and associated security vulnerability.

2.4.1 Data Modification

Data modification is a straightforward PPDM technique which operates by modifying the data, before it is released for data mining, in such a way that the quality of the released data is sufficient to maintain the effectiveness of the data analysis outcome. Two well documented data modification techniques are data anonymisation and data perturbation, these are considered in further detail in the following two sub-sections; Sub-sections 2.4.1.1 and 2.4.1.2.

2.4.1.1 Data Anonymisation

Data anonymisation has been widely used with respect to publishing datasets that have been made available for research and public benefit purposes [12]. It has been adopted with respect to many research disciplines including medical/healthcare [66], social networks [67] and marketing [68]. Data anonymisation has been used as a privacy preserving technique in the context of many data mining applications, examples include data clustering [69], data classification [70–72] and association rule mining [62].

Typically, data anonymisation operates by first removing the EID attributes from the dataset to prevent the direct identification of subjects. Then modifying the QID and/or sensitive attributes, according to whatever desired privacy requirements are in force. The data is modified using one or more of the following data "sanitising" operations: (i) data suppression, (ii) data generalisation and (iii) data permutation. Data suppression modifies data by masking part of the attributes' values (for example replacing part of the values using asterisks). Data generalisation aims to reduce the precision of the data fields by replacing values with a parent value in a given taxonomy. For example, replace exact phone numbers in a given dataset with area codes, or replacing numeric values (such as age) with an interval within which the value belongs. Data permutation (or anatomisation) aims to de-associates QIDs and sensitive attributes either by separating QIDs and sensitive attrubites in two datasets making it more difficult to link the two sets or by swapping the attributes values among records [73–75]. In the former technique the attribute values remain unchanged [75]. How the dataset is anonymised in practice, using the above operations, mainly depends on the nature of the privacy requirements, which in turn is influenced by the application domain.

From the literature there is much reported work that uses the above operations to implement data anonymisation [76–85]. The most common data anonymisation techniques found in the literature are: k-anonymisation [76–79, 85], l-diversity [81] and t-closeness [80]. The first, k-anonymisation, typically operates by modifying the QIDs, using generalisation and suppression operations, in such a way that the QID attributes for at least $k-1$ records in dataset are the same in the anonymised version of the data. The k-anonymisation will thus reduce the risk of identifying identities using a "linkage attack" that uses QIDs and some background knowledge, or alternative linkage attack that uses external data sources that hold QID values associated with entities in the target dataset. However, k-anonymisation entails a trade-off between privacy and data utility especially when the QID attributes are of particular relevance to the data analysis to be undertaken. More importantly, k-anonymisation is only applied on the QID attributes and disregarding sensitive attributes as it is assumed that without QIDs, there is no risk of a linkage attack with public information. The fact that sensitive attributes are not taken into consideration when forming the k-anonymised dataset may lead to the sensitive attribute values within the $k-1$ records having the same value which may lead to the disclosure of private information concerning the individuals referenced by the $k-1$ records. For example, all the employees referenced in the $k-1$ records have the same salary or all the patients in the same $k-1$ records suffer from the same disease. In k-anonymisation, the value of k is used as a privacy measure; the higher the value of k, the harder it is to de-anonymise records but at the cost of reducing the utility of the data.

The l-diversity [81] and t-closeness [80] techniques were introduced to address the limitation of k-anonymisation. When using the l-diversity technique the sensitive data attributes values within the $k-1$ records in the k-anonymised dataset are further modified, by either generalising or suppressing data attributes values, in such a way that the probability of associating entity with a sensitive attribute is bounded by $\frac{1}{l}$ (the value of l is used as a measure of privacy). The higher the value of l the higher the variability of the existing values of the sensitive attribute in the $k-1$ data records and the lower the possibility of sensitive attribute disclosure. The t-closeness technique further reinforced the l-diversity by ensuring that the distribution of a sensitive attribute values in the $k-1$ data records is close to the distribution of the whole data (the distance between the two distributions are not greater than a threshold t).

Generally, data anonymisation features a number of limitations:

1. All the data anonymisation techniques cause information loss, which must be minimised to maintain the accuracy of data mining results. Data anonymisation can adversely affect the accuracy of the data mining as demonstrated by the extensive experimentation reported on in [86].

2. In many cases QID attribute data modification is only applicable with respect to the single data owner scenario as to operate correctly data modification operations required access to the entire data to be modified [71, 72, 85].

3. Data anonymisation tends not be effective when using high dimensional datasets, even in the case of the single data owner scenario. This is because of the difficulty of perform data modification operations on partial ranges to find an optimal anonymisation as discussed in [87].

4. Limitation of application, as demonstrated in [86], because the way that data is anonymised is very dependent on the data itself and the purpose for which it is to be used, consequently limiting the type of data analytics that may be applied.

5. Security concerns in that data anonymisation techniques do not have provable security guarantees as in the case of many SMPC and cryptographic approaches [66]. Many incidences have been reported [14–16] that demonstrate the possibility of de-anonymising the anonymised data using "linkage attacks" as will become clear shortly in Sub-section 2.5.2. Therefore, data anonymisation does not sufficiently preserved the data privacy as the TPDM has access to sensitive attributes and the degraded QID.

Given the first of the above limitations, various metrics for measuring information loss, or loss of data utility for data mining purpose, have been proposed such as the minimal distortion metric [12, 78], the discernibility metric [79] and the normalised average equivalence class size metric [88]. To address the limitation that many proposed QID attribute modification techniques are only suited to the single data owner scenario a number of alternative k-anonymisation techniques have been introduced for collaborative data mining in the context of both vertically partitioned data [70, 84] and horizontally partitioned data [85]. The broad idea in both cases was to agree on the QID attributes to be modified and the adopted sanitising operations. However, the complexity of such agreement limits the scalability of this technique. In [89] the scalability of data anonymisation was addressed by introducing a Trusted Third Party (TTP) to act on behalf of multiple data owners. The TTP will thus perform the desired data anaonymisation without the requirement of interaction between data owners. However, usage of a TTP is not always applicable and in many cases may present a significant security risk.

Given the foregoing, and the requirements for PPDM in the context of both the single data owner scenario and the multiple data owner scenario, as identified above, it is suggested in this thesis that data anonymisation is unsuitable.

2.4.1.2 Data Perturbation

Data perturbation has been widely used as a disclosure control mechanism for preventing the release of sensitive *statistical information*. For example, before publishing a survey that holds sensitive/personal questions, statistical agencies perturb real responses (according to some distribution) so as to prevent (or at least limit) the potential of linking a respondent to a specific identity in the population [90]. This technique has been used extensively in the domain of PPDM for preserving data privacy while conducting various kinds of data mining, especially classification model generation [55].

Data perturbation preserves the data privacy by first suppressing or generalising the EIDs and QIDs and then distorting individual sensitive data attributes value in such a way that statistical information is retained in the perturbed data (unlike in the case of data anonymisation). This is achieved by introducing statistically generated "random noise" applied to the original data attribute values [55, 91–93]. The random noise can be *added* to distort individual values as in the case of [55, 94] or can be *multiplied* as in the case of [91, 92, 95]. Noise multiplicative guarantees stronger security compared to noise additive due to the amount of noise introduced using multiplication [91, 95]. For the purpose of security, it is assumed that the variance of the added noise is large enough so that the original data cannot be easily "guessed" from the perturbed version. The statistical quality of perturbed data, that is correlated to the accuracy of data analysis, is measured in terms of *bias*, *precision*, and *consistency* [96]. Bias represents the difference between the unperturbed statistics and the expected value of its perturbed estimate. Precision refers to the variance of the estimators obtained by the users while consistency represents the lack of contradictions and paradoxes.

Data perturbation has been used to implement varies secure data mining algorithms; these are surveyed in [97]. Specific examples can be found in [92], [93] and [55] in the context of k-Means, anomaly detection and decision trees. Data perturbation is applicable in both the single data owner [91, 92] and collaborative data mining contexts [55, 93, 94, 98].

Data perturbation is usually conducted over plaintext data. However, it can be applied over encrypted data as in the case of [94, 99] where the data is encrypted using homomorphic schemes that support addition or/and multiplication. In this case the data perturbation is used to preserve data privacy in the multiple data owner, collaborative, data mining context. In [99], an *interactive protocol*, to delegate the evaluation of the sigmoid activation function to the data owner by adding noise to encrypted weights and biases is presented. While the study in [94] introduces a secure collaborative DBSCAN that includes the sharing of encrypted perturbed data records between parties that facilitates the distances calculation between records belonging to different data owners. However, data perturbation has a number of major disadvantages as follows:

1. It is not easy to apply with respect to categorical attributes due to the absence of any natural ordering which limits the application of this technique to numerical datasets.

2. As already noted above, perturbation involves a security-accuracy trade-off, since the higher the level of security provided by the perturbation method the worse the accuracy of the applied data mining algorithm. This is especially the case with respect to the multiple data owners scenario when each party applies a local perturbation method to their data. To address the latter it is suggested in [94] that, to maintain the accuracy of results, all participating parties should use the same perturbation method so as to ensure an acceptable accuracy. This constraint raises a consequent security vulnerability and security risk when a third party, such as a TPDM, is involved [56].

3. Data perturbation does not provide an adequate level of security as it provides the potential for reconstructing the original data distributions using reverse engineering methods [100, 101].

2.4.2 Secure Multi-Party Computation

Secure Multi-Party Computation (SMPC) is a well-established research field that comprises a set of protocols which enable a number of parties to compute collaboratively functions or statistics concerning their data without disclosing their respective inputs [29, 30]. There are many SMPC protocols that were designed to support the calculation of different functions, including secure sum [102], secure multiplication [37], secure scalar product [36, 37, 103] and secure data comparison [30, 104]. The security primitives used to guarantee security of the underlining computation run jointly across parties depending on the nature of the individual SMPC protocol.

SMPC was first proposed, in [56], as a potential solution for PPDM security and privacy, with a focus on a secure implementation of the ID3 decision tree classifier. Subsequently many SMPC protocols have been suggested directed at the fundamental functions required by different data mining algorithms. Using these protocols, a range of secure data mining algorithms have been implemented. Examples include DBSCAN [36, 37, 40, 102, 105], k-Means [33] and Nearest Neighbour Clustering (NNC) [103]. These implementations considered different numbers of participants, two-party [36, 40,

102, 103] and multiple-party [33, 37]; and different data partionings, horizontal [33, 36, 37, 40, 103], vertical [36, 40, 102] and arbitrary [40]. A range of mechanisms was also employed to determine "similarity" between data records distributed across multiple data owners: (i) "secure scalar product" as in the case of [36, 37, 103], (ii) secure accumulation using "secure sum" as in the case of [102] and (iii) secure comparison using "Yao's Millionaires Problem Protocol" (YMPP) as in the case of [36, 37, 40, 102, 103]. In these proposed solutions, data owners are expected to undertake a significant proportion of the work and thus are required to have adequate IT resources and ability to carry out the desired analysis. Therefore, the use of SMPC protocols, regardless of the precise nature of the adopted protocol, introduces a computation and communication overhead on behalf of data owners, rendering the approach only applicable for small datasets and a limited number of data owners.

Research has been directed at reducing the SMPC computation and communication overhead by involving a third party to act as mediator [20, 33–37]. The reported research describes a range of behaviours for the third party with the third party being viewed as either a trusted party or a semi-honest party. The Trusted Third Party (TTP) will "honestly" behave and thus not deviate from pre-designed process and not use any intermediate computations for any further investigation, while the Semi-honest Third Party (STP) will have honest behaviours when executing the pre-designed process, however the intermediate computations will be analysed to learn additional information. Therefore, the former behaviour does not require any security measures to be adopted whilst the latter required the adopting of some security measures to prevent the inference of sensitive information using the intermediate computations. Sub-section 2.5.1 will consider further expelanatios for potential third party behaviours that required security measures. In [36] a TTP was used to calculate global data statistics on behalf of data owners to reduce the communication complexity of the adopted SMPC protocol. However, in this case the data owners are still required to run the entire data mining and thus the computational complexity is not avoided. More importantly, the involvement of a TTP is of concern, and for many data owners a security risk. This security issue was address in [33] by involving a STP who only had access to encrypted data.

SMPC provides a solid theoretical underpinning for PPDM. It supports a higher level of security compared to the data modification techniques considered above in Subsection 2.4.1. The security of each protocol can be proven using the "zero knowledge proof" which is used to ensure that each party has gained "zero" knowledge concerning the data and results of other participants [106]. However, in the case where the intermediate results of a data mining algorithm are revealed to all participants, this raises a security vulnerability, especially when a non-honest data owner is involved. A non-honest participant can launch an "overlapping attack" and use the obtained results of distance calculations, and data comparisons with thresholds, to determine how similar the data belonging to other participants is to the attacker's own records [40, 105]. This information can then be used to estimate the nature of the data belonging to the other participants as demonstrated in Section 2.5. A further security issue associated with SMPC is that, depending on the nature of the data mining to be undertaken, for the data mining to be executed correctly, divulging of some sensitive statistics is often required. For example in the special context of DBSCAN the total number of records within the ϵ-radius has to be revealed to all participants. In the context of k-Means clustering, "global" centroid need to be divulged to all participating parties. These disclosures constitute a threat in the presence of "non-honest" data owners participating with bogus datasets (such as empty datasets) [56]. These limitations render the SMPC technique to be inadequate for many instances of secure collaborative data mining.

2.4.3 Secret Sharing

The idea of secret sharing as a cryptography method was first introduced in [107, 108]. It involves using an encryption scheme in such a way that the secret key SK is split into a number of shares $\{sk_1, sk_2, \ldots, sk_u\}$ distributed across a predefined set of u collaborating parties. Each party can encrypt their data records independently from the other participants using the scheme public key. However, decrypting the data is not straightforward. The secret key SK, used to decrypt the data, should be reconstructed by combining the shares belonging to participating parties. For example, in t threshold setting schemes, the cyphertexts can be decrypted by combining shares of any group of t or more parties. However, below this threshold non valid decryption results. There are varies encryption schemes designed to work using the concept of secret sharing, ranging from standard forms of encryption, as in [109], to PPE schemes as in [27]. As already noted in Chapter 1, PPE schemes maintain some properties from the plaintext data with respect to the generated cyphertext. This allows collaborative secure data manipulation using the preserved properties. PPE schemes which support secret sharing have provided for a new form of collaborative data mining where a number of parties can locally encrypt their datasets which are then sent to a TPDM who will use the properties of the scheme to manipulate the entire outsourced dataset on behalf of the data owners. However, as will become clear in Chapter 3, there is no existing PPE scheme that provides an entire solution to a PPDM problem in that data owner participation is still considerable. For example in [19] secret sharing is adopted in the context of a secure DBSCAN clustering approach where data owners participation is required in order to compare distances that are calculated by the TPDM. This comparison required jointly decrypting the calculated distances and resorting to SMPC comparison protocols. In [28] a collaborative secure neural network mechanism was presented using a TPDM who manipulates the data, so far as possible, using the properties of an HE scheme. However, the data owners' participation was still required to calculate the activation function and to determine if the termination condition had been achieved or not. The intermediate results were randomly shared between data owners. The random shares allowed the parties to collaboratively execute the required calculations without knowing the provenance of the values.

Although data owner participation is reduced using the concept of secret sharing compared to that found when using SMPC protocols, secret sharing has some limitations. Firstly, the requirement that the parties are semi-honest and non-colluding is of concern, and for many data owners present a security risk. As the secret sharing allows the decryption of data whenever t shares of secret key are combined, therefore, in the case when t parties are colluding the data can be decrypted without the data owner's permission. Secondly secret sharing tends to be inefficient for large datasets and thus it is only suitable for limited collaborative data mining. Thirdly, as the secret key is split among the parties, data owners cannot decrypt their own data without the involvement of the other parties, and thus a copy of the private data needs to be kept locally by each individual data owner.

2.4.4 Homomorphic Encryption

Homomorphic Encryption (HE) is an emerging form of encryption that allows *limited mathematical operations* over cyphertexts without decryption. The nature of the supported mathematical operations, and their associated meaning with respect to plaintext equivalents, is dependant on the nature of adopted HE scheme. The permitted operations are usually referred to as homomorphic properties of the scheme. Because of the

significance with respect to the work presented in this thesis an extensive review, coupled with examples, of relevant HE schemes is presented in the following chapter, Chapter 3. Using an HE scheme data privacy is preserved by encrypting the dataset before it is outsourced to a TPDM so that the TPDM does not have access to data in its plaintext form and does not have access to the relevant decryption key. The TPDM then manipulates the encrypted data using the homomorphic properties of the chosen HE scheme. For example to calculate distances between encrypted records, or to calculate cluster centroids, and so on. Many secure data mining algorithms have been implemented using this technique. Examples include k-Means [20–22, 25, 26, 39, 110], DBSCAN [19, 94], association rule mining [111], k-NN [24, 45] and BPNN [112, 113]. However, as already noted, HE schemes support only limited operations and thus when unsupported operations are required by the data mining algorithm in question alternative methods need to be implemented.

The most common approach used to address the limited operations of HE schemes is again recourse to data owners who will perform the required operations on plaintext values, encrypt the results and send the encrypted result back to the TPDM [19, 20, 28]. To give one detailed example, in [20] where a secure k-Means mechanism is considered, the calculation of distances between data records and cluster centroids, calculated using the homomorphic properties of the utilised HE scheme, are delegated, on each k-Means iteration, to the data owner to determine the appropriate cluster of each encrypted record. The complexity of data owner participation in this case is thus $O(k \times n \times i)$ where k, n and i are the desired number of clusters to be generated using k-Means, the number of records in the dataset and the number of iterations, respectively. Further examples can be found in other contexts: (i) the secure DBSCAN implementation presented in [19] requires that distances between records need to be compared with threshold values by the data owners; (ii) the secure Artificial Neural Network (ANN) implementation presented in [28] requires that the activation function that needs to be calculated for each neuron is done by the data owners; and (iii) the secure feed-forward neural network also presented in [28] requires that the termination condition is checked by data owners.

The main advantage of the HE solution is that no party, except the data owner, has access to intermediate results when mining algorithms are in progress. However, the amount, and the complexity, of delegating functions to data owners introduces an undesirable communication and computation overhead on behalf of the data owners. More importantly, in the context of collaborative data mining, the data owner's involvement in comparing distances between data partitioned across parties raise the potential of an "overlapping attack" where one party can estimate the values of data attributes owned by other data owners using knowledge of their data records and results of comparisons [40]. Such attacks can be prevented by adopting established multi-party secure comparison protocols, such as YMPP [30] and Cachin's scheme [104]. However, these have the same major disadvantages as those encountered when using SMPC; the significant computation and computation overhead introduced with respect to the data owners in turn limits the scalability of proposed solution.

Some research has been directed at reducing the amount of data owner participation [20–22, 25, 26]. Of noted, is the "trapdoor" concept proposed in [21, 22] that aims to reduce data owner participation when comparing cyphertexts belonging to a single data owner. In this solution, *static* and *dynamic* trapdoors are provided, prior to outsourcing, and used to allow cyphertext comparison by a TPDM without data owner involvement. However, in the case of secure k-Means clustering which generates new cyphertext cluster centroids, the dynamic trapdoors need to be recalculated on each iteration. To maintain the accuracy of the final clustering results, any cyphertexts generated, by HE properties,

when k-Means is in progress, needs to be re-encrypted by the data owner. Although this approach reduces the data owner participation compared to "naïve" approaches, data owner participation remains high. In addition, from an efficiency perspective, dynamic trapdoor calculation requires data owners to maintain all intermediate random values used to encrypt their data; in other words, there is a significant computational overhead, particularly in the case of large datasets.

To reduce the data owners participation when a dataset is partitioned across multiple parties, in [20] the similarity between records is determined with reference to a randomly selected data owner. In [25, 26] a multiple non-colluding Cloud Service Provider (CSP) setting was introduced to avoid data owner participation. The primary idea was to outsource data to one CSP and a secret key to another CSP who will act on behalf of the data owners whenever unsupported operations were required. The CSP who holds the secret key can decrypt the intermediate results, perform the required operations on plaintext and send the encrypted results back to the CSP who runs the data analysis. However, this approach increases the cost of outsourcing the data and the assumption that the CSPs are non-colluding may not always hold.

2.5 Security

This section presents some security fundamentals concerning adversarial behaviours and potential attacks that can be instigated against proposed PPDM solutions. The section comprises two sub-sections. Sub-section 2.5.1 considers the adversarial behaviour that can be expected and introduces the assumed behaviour of the participants for all the proposed solution presented in this thesis. Sub-section 2.5.2 then defines, and when possible gives examples, of different types of attack model.

2.5.1 Adversarial Behaviour

In the work presented in this thesis a *client-server model* is assumed. The client is the data owner (or a set of data owners) while the server is the TPDM who uses the data owners' data to develop a data mining model of some kind, and when desired provides data querying/labelling or prediction services for a set of authorised Query Owners (QOs) as established in Sub-section 2.3.2. In general, cryptographic design considers two possible behaviours for the parties involoved [17]. The first is the *semi-honest* or *passive adversary* behaviour where the parties involved follow the pre-designed data mining algorithms correctly, but in the meantime learn additional information by analysing the intermediate results and messages exchanged during algorithm execution, which means that parties could infer private information. The second is *malicious* or *active adversary* behaviour where the parties involved arbitrarily deviate from the pre-designed data mining algorithm. For example, intermediate results that may be sent to data owners, whilst data mining is in progress, by the TPDM can be altered by the TPDM or by one of the participated data owners. In the work presented in this thesis, the behaviour of the TPDM and the data owners are assumed to be semi-honest while the QOs are assumed to be active adversaries. This is a reasonable assumption since the main objective of the CSP (the adopted TPDM) is to deliver a high quality and accurate service by constructing accurate data mining models; this can only be achieved when the TPDM is honestly follow the designated, pre-designed, algorithm. The data owner or owners also share the same goal of developing accurate models, however at the same time they may be "curious" about the data content held by the TPDM; and, in

the multiple data owner scenario, data belonging to the other participating data owners. It is assumed that the QOs will deviate from the designated, pre-designed, algorithms.

2.5.2 Attack Models

In cryptography the word "attack" refers to any attempt to break an encryption scheme or reveal information concerning private data. There are many types of attacks that can be instigated against the different PPDM techniques presented earlier in this chapter and later in this thesis. Some of these attacks rely on information leakage of the adopted PPDM technique and others assume the attackers have some background knowledge (such as data frequency). When the attack is successful it implies that data security has been compromised. In this sub-section, an overview of the major sources of threats to user privacy using PPDM techniques, and the most common attacks, are presented. Knowledge concerning these attacks is required by the reader as a precursor to the work presented later in this thesis.

Linkage Attack (LA): A LA can be instigated when an attacker has accessed to anonymised data that retains the QIDs and other sensitive data for analysis purposes. The anonymised dataset can be *de-anonymised* by cross-referencing the anonymised dataset with some public dataset where the QIDs attributes also exist. Figure 2.3 shows an example. Consider an anonymised patient's dataset (D_{anon}) that holds {*date of visit*, *symptoms*, *diagnosis*, *disease*, *charges*, *data of birth*, *sex*, *race*, *postCod*}. Consider a voter archive dataset ($D_{archive}$) that holds { *data of birth*, *sex*, *race*, *postCod*, *IdNo*, *name*, *parentage*, *address*, *econStatus*, *issue date*}. From the attribute values present in both datasets the corresponding medical records published in D_{anon} for some entities may be revealed.

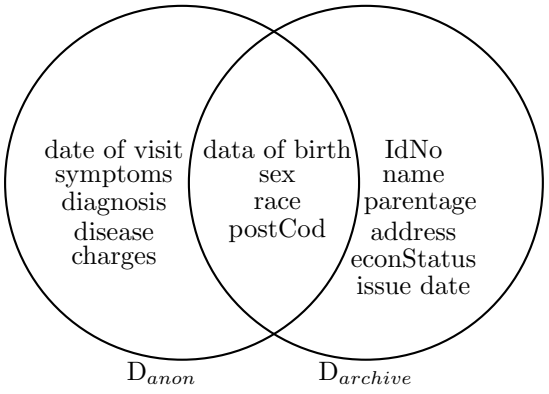


FIGURE 2.3: Example of a linking attack

Overlapping Attack (OA): An OA can be lunched in the context of a collaborative data mining process when a non-honest party has access to intermediate results when data clustering or classification model generation is in progress. The most likely intermediate results revealed in collaborative data clustering and classification are distances between records and the results of comparisons. Figure 2.4 shows two parties (Alice and Bob) who collaboratively run a DBSCAN data clustering process with radius parameter ϵ. The parties will jointly calculate the distances between each pair of data records belong to different parties. Consider three records for Bob $\{B_1, B_2, B_3\}$ and one record for Alice $\{A\}$. The following distances are calculated (B_1, A), (B_2, A) and (B_3, A) and the results will then be compared

with the DBSCAN radius parameter ϵ. The results of the data comparisons will be revealed to both Alice and Bob. Using these intermediate results Bob (a non-honest party) can determine how similar the record A is to his own records. The intermediate results coupled with the final clustering configuration will define the cluster within which record A is contained. Bob may then be able to estimate the attributes value of record A using his known values for $\{B_1, B_2, B_3\}$.

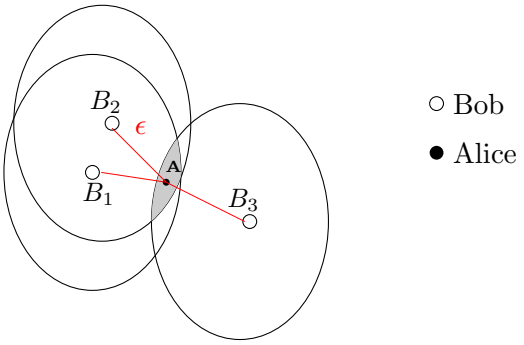


FIGURE 2.4: Example of an overlapping attack

Cyphertext Only Attack (COA): A COA is where an attacker only has access to cyphertexts, but no access to any plaintext. The cyphers might represent an encrypted dataset, encrypted cluster centroids, encrypted queries and/or a trained model's weights and biases. This attack is most likely to be encountered in real life when the adversary has access to TPDM storage or when the TPDM behaves as a curious party and thus wants to analyse cyphertexts to reveal some information. For a COA to be successful requires the adversary to have some background knowledge concerning the data under analysis, such as data distributions or data frequencies. Therefore schemes that reveal these statistics (distribution or frequency) are vulnerable to COAs. The data frequency can be revealed when the encryption scheme encrypts plaintexts to the same cyphertexts using the same key every time; schemes that do this are usually referred to as Deterministic Encryption (DET) schemes; such schemes will be considered in further details in Sub-section 3.4.1.

Knowing Plaintext Attack (KPA): A KPA is where the attacker has access to a limited number of pairs of plaintext and their corresponding cyphertexts.

Chosen Plaintext Attack (CPA): Using a CPA the attacker is able to choose a number of plaintexts to be encrypted and consequently have access to the resulting cyphertexts. This attack is most likely to occur in the case of asymmetric encryption schemes were the public key used to encrypt the data is publicly announced and anyone may use it. Schemes that produce the same cyphertext every time when the same plaintext value is encrypted using the public key are also vulnerable to a Dictionary Attack (DA). In this case the attacker creates a dictionary by simply encrypting all potential plaintext values using the scheme public key. Therefore, the inclusion of noise is an essential part of any encryption scheme designed to safeguard against CPAs and DAs.

Model Inversion Attack (MIA): A MIA is when the attacker has access to a classification model (for example an ANN or a decision tree classifier) and directs prediction queries at the model with the intention of acquiring information concerning the model's behaviour beyond simply the prediction results. In this attack,

and as noted in [114], the attacker will exploit the query predictions to reveal confidential aspects of the data originally used to train the model. Therefore, when the data classification model generation algorithms are delegated to a TPDM, it is important to preserve the privacy of the weights and biases of the developed model from the TPDM and QOs. More importantly, the query launch should be controllable by the party or parties who own the model.

2.6 Summary

This chapter has presented the necessary background concerning PPDM that underpins the work presented in the following chapters of this thesis. The chapter commenced by defining data privacy and determining what the term "private" constitutes. The privacy preservation requirements in the context of the two scenarios considered in this thesis, the single data owner scenario and the multiple data owners scenario, were presented. The requirements highlighted the dilemma of protecting privacy while at the same time maintaining the accuracy and the efficiency of any data mining activity. A range of existing PPDM techniques were reviewed. Some of these techniques, such as SMPC and secret sharing, were only relevant to collaborative data mining whilst the other techniques could be extended to the collaborative data mining context, although both will have some consequent computation and communication overhead. In the single data owner scenario data privacy can be preserved using data anonymisation, perturbation and data encryption using an appropriate homomorphic encryption. The chapter was concluded with a discussion of the assumed participant behaviours in the context of the work presented in this thesis, and the set of potential attacks that can be instigated in the context of PPDM. The following chapter presents some fundamentals and preliminaries concerning cryptography with an emphasis on schemes relevant to PPDM.

Chapter 3

Cryptography Fundamentals and Preliminaries

3.1 Introduction

As noted in Chapter 1 the research described in this thesis aimed to use Homomorphic Encryption (HE), coupled with the concept of distance matrices, as a technique to address the PPDM problem in the context of data clustering and classification. This chapter will thus provide the necessary background concerning cryptography with an emphasis on HE and Property Preserving Encryption (PPE) Schemes. HE and PPE schemes, in general, are designed to be used with respect to long-term data storage and computation outsourcing of sensitive data to a third party (such as CSPs). The data owners encrypt their data prior to uploading it to a third party tasked with manipulate the encrypted data, using the properties of the encryption scheme, without decryption. The remainder of this chapter is constructed as follows. Section 3.2 presents a comprehensive background to cryptography and HE. The section includes consideration of basic cryptographic terminologies, HE schemes and their mathematical properties, an overview of HE schemes and a discussion of the limitations associated with HE as relevant to the development and implementation of data mining algorithms. The chapter then continues, in Section 3.3, with material concerning two HE schemes adopted with respect to the proposed solutions presented later in Chapters 4 to 9: (i) Liu's fully HE scheme [53] and (ii) the Paillier HE scheme [54]. This is followed by a review of different PPE schemes as potential solutions to third party data mining. Finally, the chapter is concluded with a summary in Section 3.5.

3.2 Homomorphic Encryption Fundamentals

HE is an emerging encryption technique which allows *certain computations* to be performed on cyphertexts without decryption. The results of these operations, once decrypted, should match the result of some form of operation performed using the original unencrypted data. This means, effectively, that we can perform computation (algebraic operations) without having access to the original data and thus preserve data privacy. This is not the case in traditional encryption techniques, where the result of operations involving encrypted data has no meaningful interpretation. However, HE shares many of the key concepts, terms and notation with the traditional encryptions. Therefore, this section will present some fundamental background to cryptography with an emphasis on HE. The Section is organised as follows. Sub-section 3.2.1 presents the fundamental

principles of encryption. This is followed, in Sub-section 3.2.2, with the fundamental principles of HE schemes. The nature of HE schemes is then discussed in further detail in Sub-section 3.2.3. The sub-section includes consideration of three categories of scheme, their characteristics and homomorphic mathematical properties: (i) Partial Homomorphic Encryption, (ii) SomeWhat Homomorphic Encryption and (iii) Fully Homomorphic Encryption. The limitations of HE schemes which are pertinent to the implementation of PPDM are discussed in Sub-section 3.2.4 along with solutions that have been proposed in the literature.

3.2.1 Background to Encryption

Encryption is the process of encoding data, so that it is no longer in its original form, in such a way that it cannot be de-encoded by unauthorised parties. It is a well-known security primitive, that substantially guarantees data privacy preservation, which is widely adopted with respect to many application domains [17, 115]. Examples include Electronic Health Records [116], wireless sensor networks [22], and smart cities and Internet of Things [117, 118]. The unencrypted data value v is referred to as a *plaintext* value, whilst the encrypted equivalent e is referred to as a *cyphertext* value. The plaintext belongs to what is known as the *message space* ($\mathcal{M}$), whilst generated cyphertexts belongs to what is known as the *cyphertext space* ($\mathcal{C}$). Each encryption scheme defines its own message space and cyphertext space. The message space is required to comprise numeric data such as: integers, as in the case of [54, 119–125]; real values, as in the case of [53, 126]; or binary data as in the case of [127–135]. Categorical data needs to be translated into an appropriate numeric form. The cyphertext space depends on the encryption algorithm used.

The encryption (`Encrypt`) and decryption (`Decrypt`) functions utilise scheme keys to map plaintext from the message space ($\mathcal{M}$) into cyphertext in the cyphertext space ($\mathcal{C}$) and recover the value back from cyphertexts respectively. The encryption/decryption is conducted using what are known as encryption keys. There are two classes of encryption schemes categorised according to nature of the encryption keys used: *asymmetric* and *symmetric* encryption schemes [136]. The *asymmetric* (or public key) schemes use different keys for encrypting and decrypting the data referred to as the Public Key (PK) and the Secret Key (SK) respectively. The *symmetric* (or private key) schemes use the same key (the secret key SK) for the encryption and decryption. The schemes that belong to the latter category need to keep their key private whilst the former category of scheme can widely distribute their PK without compromising security since the public key provides no information about the secret key which is required for decryption. Examples of asymmetric HE schemes can be found in [54, 120, 121, 127–130, 137, 138], whilst examples of symmetric HE schemes can be found in [53, 119, 122, 126, 139].

Using the encryption key, PK in the asymmetric scheme and the SK in the symmetric scheme, the `Encrypt` function maps a plaintext value v to a cyphertext e. An encryption scheme that produces different cyphertexts for the same plaintext value each time the same plaintext is encrypted using the same key is referred to as *probabilistic* scheme whilst an encryption scheme that produce the same cyphertext on each occasion is referred to as a *deterministic* scheme. Many HE schemes are probabilistic in nature. Examples can be found in [53, 54, 119–122, 126–130, 137–139]; whilst there are few HE schemes that can be considered to be deterministic [123, 140]. Probabilistic schemes featured *semantic security*, which makes the scheme preferable in terms of PPDM. Semantic security means that the knowledge of a cyphertext does not provide any useful information concerning the plaintext with respect to a hypothetical adversary [17]. In

other words, an adversary would be unable to recover plaintext values from their corresponding cyphertexts. More formally, given cyphertext e and plaintexts v_1 and v_2 the adversary cannot guess, with a higher probability than $\frac{1}{2}$, whether e is an encryption of plaintext v_1 or v_2. Conversely, the `Decrypt` function decodes the cyphertext to the original plaintext using the scheme secret key (SK).

3.2.2 Homomorphic Encryption: Definition and Basic Properties

As already noted, the phrase "Homomorphic Encryption" refers to encryption techniques which allows certain computations to take place directly over cyphertexts; that is, without the need to first decrypt the cyphertext. The concept of HE has its roots in the work of Rivest et al. [124], although the phrase "privacy homomorphism" was used. The first true additive HE scheme is that presented in [141], referred to as R-additive privacy homomorphism, in which only the addition of at most R plaintexts was allowed. Formally HE schemes can be defined as in Definition 3.1.

Definition 3.1 (*Homomorphic Encryption*)**.** An encryption scheme is said to be homomorphic for some operation $\circ \in \mathcal{F}_{\mathcal{M}}$, where $\mathcal{F}_{\mathcal{M}}$ is a set of operations that can be applied to the message space (such as addition), if there exist a corresponding operation $\diamond \in \mathcal{F}_{\mathcal{C}}$, where $\mathcal{F}_{\mathcal{C}}$ is the set of operations in cyphertext space, if the following property is satisfied:

$$\texttt{Decrypt}\Big(SK, \texttt{Encrypt}(PK, v_1) \diamond \texttt{Encrypt}(PK, v_2)\Big) = v_1 \circ v_2 \tag{3.1}$$

This is illustrated in Figure 3.1 which shows an HE message space, containing three plaintext values, together with the associated cyphertext spaces holding corresponding cyphertext values. The operation ($\circ$) defined for the message space equates to an alternative operation ($\diamond$) in the cyphertext space. As depicted in the figure, cyphertext e_3 is generated by applying a homomorphic operation ($\diamond$) to the cyphertexts e_1 and e_2. The cyphertext e_3, when decrypted, will result in v_3, which will equate to the value produced when applying the plaintext operation ($\circ$) to the plaintexts v_1 and v_2. It should also be noted here that the cyphertext that is the results of a homomorphic operation should retain semantic security (as described early in Sub-section 3.2.1).

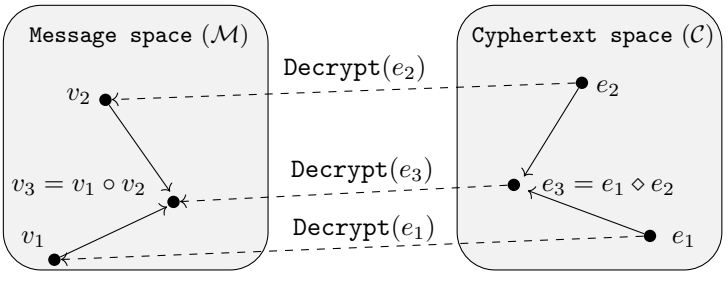


FIGURE 3.1: The homomorphic property: the result of applying homomorphic operation $\diamond$ to cyphertexts e_1 and e_2, to give e_3, will match the result of applying operation $\circ$ to plaintexts v_1 and v_2 to give v_3

In practice, each scheme defines their set of permitted operations, $\mathcal{F}_{\mathcal{M}}$ and $\mathcal{F}_{\mathcal{C}}$, without the restriction that the operations should necessarily corresponds. For example, Liu's scheme [53] defines $\mathcal{F}_{\mathcal{M}} = \{+, \times\}$ and $\mathcal{F}_{\mathcal{C}} = \{+, \times\}$ in such a way that addition on cyphertexts will correspond to the addition of plaintexts, and multiplication on cyphertexts will correspond to the multiplication of plaintexts. However, in the case of Paillier encryption [54], which is homomorphic only for addition, $\mathcal{F}_{\mathcal{M}} = \{+\}$ and $\mathcal{F}_{\mathcal{C}} = \{\times\}$ are

defined; thus applying multiplication on cyphertexts corresponds to applying addition to plaintexts. Liu's and Paillier schemes are both used with respect to solutions presented later in this thesis and are therefore considered in further detail in Section 3.3 below. For the purpose of distinguishing between the two sets of operations, $\mathcal{F}_{\mathcal{M}}$ and $\mathcal{F}_{\mathcal{C}}$, the work presented in this thesis will use the standard symbols $+$ and $\times$ for plaintext addition and multiplication, and the symbols $\oplus$ and $\otimes$ for cyphertext addition and multiplication. From a practical perspective, the operations $\oplus$ and $\otimes$ are typically more complicated to implement than the standard arithmetic operations. For instance there implementation may involve polynomial addition, or multiplication when cyphertexts are polynomial in nature [53, 142], or matrix addition and multiplication when cyphertexts are matrices [126, 143].

It should also be noted here that in the case of probabilistic HE schemes, as opposed to deterministic HE schemes, the homomorphic properties do not commute due to the probabilistic feature used in these schemes to achieve semantic security. This is because the same value presented twice will encrypt to two different cyphertexts with high probability. In general:

$$\texttt{Encrypt}(PK, v_1) \diamond \texttt{Encrypt}(PK, v_2) \neq \texttt{Encrypt}(PK, v_1 \circ v_2) \tag{3.2}$$

3.2.3 Overview of Homomorphic Encryption Schemes

HE schemes can support addition and/or multiplication over cyphertexts. Depending on the nature of the supported homomorphic operations and the number of operations, HE schemes can be categories into: (i) Partial Homomorphic Encryption (PHE) schemes, (ii) SomeWhat Homomorphic Encryption (SWHE) schemes and (iii) Fully Homomorphic Encryption (FHE) schemes. Each will be discussed in further detail in the following three sub-sections; Sub-sections 3.2.3.1 to 3.2.3.3.

3.2.3.1 Partial Homomorphic Encryption

Schemes that are categorised as PHE schemes support addition or multiplication on cyphertexts, but not both. These schemes can be further divide on the basis of the operation they support as Additive PHE and Multiplicative PHE schemes. A survey of the literature indicates that most PHE schemes adopt an asymmetric encryption approach, and that most PHE schemes are probabilistic. Exceptions are the Rivest, Shamir and Adleman (RSA) without padding [140] and Modified RSA Encryption Algorithm (MREA) PHE schemes [123], which are deterministic schemes. Table 3.1 lists some of the more common PHE schemes labelled in terms of their scheme properties, the nature of their message space ($\mathcal{M}$), the homomorphic operations they support and the "hardness" of the associated security assumptions. The security assumption refers to the adopted mathematical structure and is used to estimate their security level in a formal manner. The security assumptions are based on well-identified mathematical problems which are hard to solve in general, but easy to solve when a trapdoor (a secret key) is known. The security assumptions listed in the table are: Semantic Security (SS), Quadratic Residuosity (QR), Discrete Logarithm (DL) problem, Composite Residuosity (CR), Integer Factorization (IF) and e' the Root (eR). The majority of schemes reported in Table 3.1 are categorised as Additive schemes whilst only RSA without padding and Elgamal are Multiplicative. In terms of performance, the RSA, Elgamal, and Paillier schemes have been shown to be efficient [144]. The GM scheme [127] is defined on a binary message space ($\mathbb{F}_2$) and encrypts one bit at a time. Therefore, when using the GM scheme, values to be encrypted need to first be converted to their binary form and every

bit in the resulting binary string encrypted individually. Therefore, in this scheme the homomorphic addition over cyphertexts are performed as a bitwise exclusive-or (XOR) operation. The remaining five schemes listed in the table [54, 120, 121, 123, 140] are defined over a positive integer message space ($\mathbb{Z}_N^+$) in such a way that the message space of the values that may be encrypted range from 0 to $N-1$ where N is the RSA modulus. This will be illustrated further in Sub-section 3.3.2 using the Paillier scheme used with respect to the work presented later in this thesis.

The performance efficiency associated with PHE makes it appealing for many real-world privacy preserving applications. PHE schemes have been widely applied in electronic voting protocols [145], biometric applications [146, 147] and MIT CryptDB[1] [148]. However, the limitation of PHE schemes is the limited range of operations available. This is sometimes solved by adopting multiple PHE schemes, one for each operation required. However, this has limited PHE schemes from being accepted as solutions for outsourced data analysis given that the data mining algorithms require operations that are not all supported by PHE schemes (as will be discussed in further detail in the next sub-section; Sub-section 3.2.4).

Table 3.1: A Survey of different partially homomorphic encryption schemes

Scheme	Symmetric	Asymmetric	Probabilistic	Deterministic	$\mathcal{M}$	$\mathcal{F}_\mathcal{M}$	$\mathcal{F}_\mathcal{C}$	Security Assumption					
								SS	QR	DL	CR	IF	eR
CEG [121]		√	√		$\mathbb{Z}_N^+$	$+$	$\otimes$	√		√			
Elgamal [120]		√	√		$\mathbb{Z}_N^+$	$\times$	$\otimes$	√		√			
GM [127]		√	√		$\mathbb{F}_2$	$+$	XOR	√	√				
MREA [123]		√		√	$\mathbb{Z}_N^+$	$+$	$\otimes$				√	√	
Paillier [54]		√	√		$\mathbb{Z}_N^+$	$+$	$\otimes$	√	√		√		
RSA [140]		√		√	$\mathbb{Z}_N^+$	$\times$	$\otimes$					√	√

3.2.3.2 SomeWhat Homomorphic Encryption

A few researchers have attempted to improve the versatility of HE schemes through, to some extent, permitting both addition and multiplication. The result was what became known as SWHE schemes which supported both homomorphic addition and multiplication, but limited in the number of times the operations can be performed before the cyphertext noise grows too large and correct decryption becomes impossible. The noise originates from the probabilistic encryption process where some noise is added during the encryption and removed during the decryption. This noise grows with successive homomorphic operations and eventually makes it impossible to decrypt the results. Because of this limitation, where by any proposed scheme is bounded by cyphertext noise, SWHE schemes cannot have an arbitrary (unlimited) number of operation applications; hence the name "somewhat" HE.

There are a number of examples of SWHE schemes that support unlimited addition but only one multiplication over cyphertexts, these include BGN [137], Smart and Vercauteren (SV) [138] and Polly Cracker with Noise [149]. In other cases the precise

[1]CryptDB is a system designed by MIT to execute enterprise queries regarding encrypted data in a MySQL database.

number of multiplications depends on the scheme parameters as in the case of [125, 129]. It is not always the total number of multiplications which is the limiting factor, rather the *depth* of multiplication. For example, $x_1 \times x_2 \times x_3$ has a multiplicative depth of 2, whereas $(x_1 \times x_2) + (x_3 \times x_4) + \cdots + (x_{n-1} \times x_n)$ has multiplicative depth of 1. As already mentioned, the exact depth for a SWHE scheme is dependent on the scheme itself and the parameters chosen, which commonly involves a trade-off between speed, security and/or memory requirements against multiplication depth.

3.2.3.3 Fully Homomorphic Encryption

The FHE schemes allow an unlimited number of additions and multiplications to be performed on cyphertexts without losing the ability to correctly decrypt the results. The first FHE scheme was introduced by Gentry [130] and constructed by coupling an existing SWHE scheme with a noise management technique called *bootstrapping*. The preliminary idea was to refresh the SWHE cyphertexts after each multiplication so as to bring the noise level down to that of a freshly encrypted cyphertext and thus permit an arbitrary number of homomorphic multiplications given that the adopted SWHE scheme already supported an arbitrary number of additions. Many FHE schemes that have been proposed subsequently have followed the same fundamental idea. The disadvantage of these schemes is that the adopted noise management techniques tend to be complex. A technical review of a number of FHE schemes, in terms of their encryption characteristics, message space ($\mathcal{M}$) and adopted noise management technique, is given in Table 3.2. Note that the listed schemes are all probabilistic and mostly asymmetric schemes. From the table it can be seen that the noise management techniques include: *bootstrapping*, *modulus switching*, *scale invariant* and *flattening*. However, the performance overhead associated with these techniques remains a major obstacle in the face of the practical implementation of FHE schemes. Each of the noise management techniques included in Table 3.2 is discussed in some further detail in the remainder of this sub-section.

The *bootstrapping* noise management technique has access to the secret key that is encrypted as a portion of the public key and is associated with asymmetric FHE schemes. The bootstrapping process refreshes the cyphertexts by homomorphically computing the decryption function using an encrypted secret key. Although bootstrapping methods allow the performance of an arbitrary number of multiplication operations, its main drawback is the complexity of the homomorphic decryption, making it impractical for many applications. Improving the efficiency of bootstrapping, as first proposed by Gentry [130], has received much attention, see for example [150–152], which has resulted in an enhanced asymptotic performance [152, 153]. Despite all these attempts, bootstrapping is far from being practical for PPDM due to the requirement to repeatedly apply the bootstrapping process given that many data mining algorithms, such as Artificial Neural Network (ANN), require many data multiplication operations. The performance issue associated with bootstrapping has instigated the emergence of more lightweight techniques as discussed below.

The *modulus switching* noise management technique does not fully refresh a cyphertext (as in the case of bootstrapping), but successfully limits the noise growth in the cyphertext during homomorphic computations. Using a technique similar to the "dimension reduction" mechanism proposed later in this thesis, the magnitude of the noise can be reduced without knowing any information about the secret key as in the case of bootstrapping. Instead, the process only needs to know the cyphertext size bound in order to transform a cyphertext, e modulo q into a different cyphertext modulo p without sacrificing the correctness of the decryption procedure (so that the modulo is

TABLE 3.2: A survey of different fully homomorphic encryption schemes

Scheme	Symmetric	Asymmetric	Probabilistic	Deterministic	$\mathcal{M}$	Noise Management				
						Bootstrapping	Modulus Switching	Scale Invariant	Flattening	Noise free
BGV [125]		√	√		$\mathbb{Z}_N^+$		√			
BLLN [132]		√	√		$\mathbb{F}_2$			√		
Brakerski's [131]		√	√		$\mathbb{F}_2$			√		
BV-LWE [129]		√	√		$\mathbb{F}_2$		√			
DGHV [128]		√	√		$\mathbb{F}_2$	√				
F-NTRU [134]		√	√		$\mathbb{F}_2$				√	
Gentry [130]		√	√		$\mathbb{F}_2$	√				
GSW [133]		√	√		$\mathbb{F}_2$				√	
KH [122]	√		√		$\mathbb{Z}_N^+$					√
Liu's (2015) [119]	√		√		$\mathbb{Z}_N^+$					√
Liu's (2013) [53]	√		√		$\mathbb{R}$					√
LW [126]	√		√		$\mathbb{R}$					√
Niu15 [135]		√	√		$\mathbb{F}_2$					√
WANG [139]	√		√		$\mathbb{F}_q$					√

switched from q to p). However, as a result, this technique has a small cyphertext size as compared to the bootstrapping technique, which may be a security concern.

The *scale invariant* technique was introduced in [131]. The schemes that adopt the scale invariant technique encode the plaintexts differently from other FHE schemes in such a way that the noise, caused by the homomorphic multiplications, increases by a fixed factor independent of the magnitude of the noise in the cyphertexts. In other schemes the homomorphic multiplication is essentially sensitive, and each time the multiplication is performed the amount of noise is squared. In scale invariant based schemes, a new cyphertext for a given plaintext value can be generated using one of the plaintext cyphers by multiplying the cyphertext with an appropriate "scalar"; a modulus switching technique is used to manage the amount of noise in generated cyphertexts. However, the limitation of the scale invariant technique is that a more complex rounding operation is required for homomorphic multiplication [154].

The *flattening* noise management technique was introduced in [143] and is based on the modulus switching technique. However, it is only useful when the cyphertext is presented in matrix form and the encryption key is presented as a vector. The flattening technique uses a transformation to modify the vectors, which results in a better bound on noise growth.

Given the foregoing, most FHE schemes are founded on the seminal work by Gentry [130] and hence FHE schemes are constructed in two steps: (i) design of a SWHE scheme and (ii) adoption of a noise management techniques to transform the proposed SWHE scheme into a FHE scheme. There is a need for more efficacious FHE schemes that achieve FHE directly without the need for a SWHE precursor. Recent work is directed

at constructing noise-free FHE schemes based on what is referred to as the "number theoretic approach" [53, 119, 122, 126, 135, 139].

3.2.4 Homomorphic Encryption Limitations

In this sub-section, an overview of the most significant limitations of HE schemes, as relevant to the development and implementation of secure data mining algorithms to support PPDM, is presented. These limitations vary slightly between encryption schemes, however, a large number of these are common to most HEs. The work presented in Chapters 4 to 9 of this thesis is, in part, directed at mitigating against some of the HE limitations presented here, especially to limit data owner participation. In the following the broad limitations in HE are discussed and their possible mitigation, as proposed in the literature, presented.

1. **Message space:** Most HE schemes restrict the message space to either binary ($\mathcal{M} = \mathbb{F}_2 = \{0, 1\}$) or positive integer ($\mathcal{M} \in \mathbb{Z}^+$). This means that the direct encryption of categorical values and real number values is not supported. With respect to categorical values the solution is to assign an integer ordinal numbering of some kind. With respect to real numeric values the solution that is frequently adopted is to use rational approximations by encrypting the numerator and denominator separately and using the rules of arithmetic for fractions (although this is very inefficient). A more efficient transformation technique is presented in [155] designed to represent real values within the practical limitation of HE schemes. The transformation is based on the IEEE standard for floating-point arithmetic (IEEE 754) [156]. However, it is still an expensive process and impractical for many instances of PPDM problem. A more straightforward solution is to approximate the value to the nearest integer value, however, this will clearly effect the effectiveness of the data mining. This solution is insufficient in the context of some data mining algorithms, such as ANNs, which required network weights to be real values of less than 1. The most effective and efficient solution to date was presented in [157], the solution required the predefinition of the number of decimal places to be retained in the plaintext value to be encrypted. This value is selected according to the desired accuracy level, say a. The plaintext value to be encrypted is first multiplied by 10^a and then rounded to the nearest integer. The result will be then encrypted using any scheme defined over $\mathbb{Z}$. When the cyphertext needs to be decrypted it is first decrypted and then divided by 10^a to arrive at the correct plaintext value.

   Referring back to Table 3.2 only two schemes operate over a real number message space ($\mathcal{M} \in \mathbb{R}$), Liu's FHE scheme [53] and the LW scheme [126]; these schemes therefore permit direct real value encryption. The LW scheme is not as efficient as Liu's scheme. However, Liu's scheme features substantial inflation of cyphertexts after each multiplication rendering the scheme to be insufficiently scalable for some instances of PPDM as will be discussed in Sub-section 3.3.1.

2. **No division:** Homomorphic division, where the two operands are cyphertexts, is currently not supported; in other words there is no FHE scheme for which there exists a homomorphic operation $\diamond$ such that $\texttt{Decrypt}(e_1 \diamond e_2) = \frac{v_1}{v_2}$. In some cases, it is possible to avoid division altogether by rescaling the algorithms appropriately to give results proportional to the true results. Examples of this approach are given in [158].

3. **Non-polynomial function:** Evaluating non-polynomial functions, for example e^x or $\sqrt{x}$, is problematic in practice when x is encrypted. Iterative methods [159] and lookup tables [160] can be used as an alternative to approximate the value of some particular non-linear functions. Polynomial approximations methods such as Taylor series expansions [159] is an alternative solution, however it is unreliable as the approximation needs to be done using a high degree polynomial for accurate results to be obtained, this in turn increases the amount of noise and the size of cyphertexts.

4. **Bounding cyphertext noise:** As already noted, using FHE schemes, as the number of operations performed on the cyphertexts increases the noise component also increases. Once the noise has crossed a certain threshold, a cyphertexts can no longer be decrypted correctly and the data is lost. Noise management mechanisms were considered in Sub-section 3.2.3.3 where it was noted that FHE schemes adopt computationally intensive mechanisms to re-encrypt (or refresh) cyphertexts without exposing the underlining plaintexts values in the process. Four such mechanisms were outlined in Sub-section 3.2.3.3: (i) bootstrapping [128, 130], (ii) modulus switching [125, 129], (iii) scale invariant [131, 132] and (iv) flattening [133, 134]. These mechanisms produce another cyphertext for the same plaintext value, but with less noise so that more additions and multiplications can be performed without losing the ability to decrypt the outcomes. However, in many cases these mechanisms introduce performance limitations. A recent innovation is the *noise-free* encryption scheme which avoids the requirement of adopting noise management mechanisms by allowing arbitrary large quantities of noise without losing the ability of decrypting the cyphertexts. Examples of these schemes can be found in [53, 119, 122, 126, 135, 139, 161, 162].

5. **No comparison:** Comparison operators, such as tests of equality and inequality ($<$, $>$, $\leqslant$, $\geqslant$, $\neq$ and $=$) cannot be performed on HE cyphertexts. HE schemes tend to be probabilistic, only deterministic schemes can support equality checking ($=$ and $\neq$). The most significant obstacle here is with respect to algorithms that require the evaluation of data comparisons or logical conditions to decide what action to perform next. This is a particular feature of the majority of data mining algorithms. For example, for some data clustering algorithms, the decision to add a data record to an existing cluster or create a new cluster requires comparison operations. One mitigating proposal is recourse to data owners to perform these operations, or to resort to SMPC comparison protocols such as Yao's Millionaires Problem Protocol (YMPP) [30] or Cachin's scheme [104]. Both feature an extensive computation and communication overhead. The inability of HE schemes to realise comparison operations is the most significant limitation of these schemes with respect to PPDM.

The above limitations raise numerous difficulties for homomorphic computation, in particular with respect to the implementation of statistical and data mining algorithms and, as emphasised later in this thesis (Chapters 4 to 9) represent a significant limitation of the application of HE schemes in the context of PPDM. Arguably, the most substantial of these limitations is the absence of comparison operations which are typically required by most data clustering and classification applications, and to calculate the non-linear polynomial functions that are required, for example, in the case of ANN training.

3.3 Homomorphic Encryption Schemes Examples

The work presented in this thesis has adopted two existing HE schemes that feature: (i) a sufficient level of security, (iii) efficiency in term of execution time and (iii) provided adequate homomorphic properties which could facilitate a wide range of data mining algorithms. These are the Liu's FHE scheme defined in the patent specification [53] and Paillier PHE [54]. The former encryption scheme has been adopted for encrypting the outsourced data in proposed solutions presented in Chapters 4 to 7 and is also considered in term of scalability in Chapter 9. The Paillier scheme has been utilised to facilitate the functionality of the Multi-Users Order Preserving Encryption (MUOPE) scheme proposed in Chapter 6. Therefore, in the following two sub-sections, Sub-sections 3.3.1 and 3.3.2, Liu's FHE scheme and the Paillier PHE scheme are comprehensively presented. The sub-sections will consider the scheme key generation process, encryption and decryption functions, scheme homomorphic properties and scheme security. In addition, the schemes are also considered in terms of their computational aspects, especially Liu's FHE schemes as it is one of the FHE schemes that feature inflation in the number of sub-cyphertexts as homomorphic multiplications are applied. The inflation problem, associated with Liu's homomorphic multiplication, is addressed in this thesis by introducing a novel scheme that modifies Liu's original scheme, referred to as the Modified Liu's Scheme (MLS) and presented in Chapter 9. The MLS provides a potential to reduce the number of sub-cyphertexts back to their original size using a "dimensionality reduction" algorithm and the concept of trapdoor.

3.3.1 Liu's Fully Homomorphic Encryption Scheme

The Liu's scheme [53] is a "modern" FHE scheme, in the sense that: (i) it does not require any noise management technique and (ii) the scheme message space accommodates the representation of real numbers $\mathbb{R}$. In the following sub-sections, Liu's scheme is presented in detail, covering: (i) the key generation process, encryption and decryption in Sub-section 3.3.1.1; (ii) the scheme's homomorphic properties in Sub-section 3.3.1.2; (iii) the level of security provided by the scheme in Sub-section 3.3.1.3; and (iv) details on the computational costs, and the associated dependency on the scheme's parameters, in Sub-section 3.3.1.4.

3.3.1.1 Key, Encryption and Decryption

Liu's scheme is symmetric scheme that uses the same key for encryption and decryption purposes. The secret key SK is a list; $SK(m) = \Big[(k_1, s_1, t_1), \ldots, (k_m, s_m, t_m)\Big]$ where k_i, s_i and t_i are real numbers. The scheme requires that $SK(m)$ satisfies the following two conditions: (i) $m \geqslant 3$ and (ii) $k_m + s_m + t_m \neq 0$. Given $SK(m)$, the scheme encrypts a plaintext value v to a set of m sub-cyphertexts; $E = \{e_1, \ldots, e_m\}$. The pseudo code for the data encryption is as given in Algorithm 1. The algorithm starts by generating m non-zero random numbers, R= $\{r_1, \ldots, r_m\}$ (line 2). For the purpose of security, the value of the random numbers should be sufficiently large. In line 3, the cyphertext E is dimensioned as a set of m elements; E= $\{e_1, \ldots, e_m\}$. The cyphertext set elements are calculated as per the equations given in lines 4 to 8. The first and last elements in the sub-cyphertexts set are calculated in a manner different to that used for the middle sub-cyphertexts as clearly shown in Algorithm 1. The Encryption algorithm will exit with the cyphertext E (line 9).

Algorithm 1 Liu's scheme encryption algorithm

1: **procedure** ENCRYPT($v, SK(m)$)
2: `Uniformly generate` m `large random numbers` $R = \{r_1, ..., r_m\}$
3: `Declare` E `as a set of` m `elements`
4: $e_1 = k_1 \times t_1 \times v + s_1 \times r_m + k_1 \times (r_1 - r_{m-1})$
5: **for** $i = 2$ to $m - 1$ **do**
6: $e_i = k_i \times t_i \times v + s_i \times r_m + k_i \times (r_i - r_{i-1})$
7: **end for**
8: $e_m = (k_m + s_m + t_m) \times r_m$
9: **Exit** with E
10: **end procedure**

Given a set of sub-cyphertexts $E = \{e_1, \ldots, e_m\}$ and the scheme secret key $SK(m)$, Algorithm 2 will decrypt E and return the value of the original plaintext v as per the equations given in lines 2 to 4.

Algorithm 2 Liu's scheme decryption algorithm

1: **procedure** DECRYPT($E, SK(m)$)
2: $t = \sum_{i=1}^{m-1} t_i$
3: $s = \frac{e_m}{(k_m + s_m + t_m)}$
4: $v = \frac{\left(\sum_{i=1}^{m-1} (e_i - s \times s_i)/k_i\right)}{t}$
5: **Exit** with v
6: **end procedure**

3.3.1.2 Homomorphic Properties

Liu's scheme supports the addition and multiplication of cyphertexts which, as previously noted, are indicated using the notation $\oplus$ and $\otimes$ respectively. The homomorphic addition for Liu cyphertexts, $E_1 = \{e_{1_1}, \ldots, e_{1_m}\}$ and $E_2 = \{e_{2_1}, \ldots, e_{2_m}\}$ that encrypt v_1 and v_2 respectively, are implemented as sub-cyphertexts additions. Each sub-cyphertexts in E_1 will be added to the corresponding sub-cyphertexts in E_2 as per Equation 3.3. Homomorphic multiplication is implemented by determining the outer product of the two cyphertexts to be multiplied as also given in Equation 3.3. Therefore, for one multiplication the number of sub-cyphertexts is increased from m to m^2 and continues to exponentially increase with each homomorphic multiplication operation.

$$\begin{aligned} E_1 \oplus E_2 = &\quad \{e_{1_1} + e_{2_1}, \ldots, e_{1_m} + e_{2_m}\} && = v_1 + v_2 \\ E_1 \otimes E_2 = &\quad \{e_{1_1} \times e_{2_1}, \ldots, e_{1_1} \times e_{2_m}, \ldots e_{1_m} \times e_{2_1}, \ldots, e_{1_m} \times e_{2_m}\} && = v_1 \times v_2 \end{aligned} \tag{3.3}$$

The scheme also allows cyphertexts multiplication by a plaintext value c; indicated using the notation $\circledast$ and implemented as given in Equation 3.4. By extension, cyphertexts subtraction ($\ominus$) can be implemented by concatenating the operations $\circledast$ and $\oplus$ as shown in Equation 3.4. As already noted, cyphertext multiplication changes the number of sub-cyphertexts; however the homomorphic operations $\oplus$ and $\ominus$ require the two operand cyphertexts to be of the same length. As a result, when $\oplus$ or $\ominus$ is required to be conducted with operands that have different sub-cyphertexts sizes, the cyphertext

with the lower number of sub-cyphertexts is iteratively homomorphically multiplied with $\texttt{Encrypt}(1, SK(m))$ till the appropriate number of sub-cyphertexts are arrived at. For example, to evaluate $v_1 + v_1 \times v_2 + v_1 \times v_2 \times v_3$ using Liu's FHE scheme the operands need to be of the same length (same number of sub-cyphertexts). This means, the operands v_1 and $v_1 \times v_2$ need to be multiply with $\texttt{Encrypt}(1, SK(m))$ twice and once respectively. This is demonstrated in Equation 3.5 where E_1, E_2 and E_3 are the encrypted equivalents of v_1, v_2 and v_3 respectively.

$$\begin{aligned} c \circledast E_1 = &\quad \{c \times e_{1_1}, \ldots, c \times e_{1_m}\} \\ E_1 \ominus E_2 = &\quad E_1 \oplus (-1 \circledast E_2) \end{aligned} \tag{3.4}$$

$$\begin{aligned} v_1 + v_1 \times v_2 + v_1 \times v_2 \times v_3 \equiv &\quad E_1 \otimes \texttt{Encrypt}(1, SK(m)) \otimes \texttt{Encrypt}(1, SK(m)) \\ &\oplus E_1 \otimes E_2 \otimes \texttt{Encrypt}(1, SK(m)) \\ &\oplus E_1 \otimes E_2 \otimes E_3 \end{aligned} \tag{3.5}$$

3.3.1.3 Security

Liu's FHE scheme is a probabilistic scheme in that it produces different cyphertexts for the same plaintext on each occasion, even when the same secret key is used. This is achieved by the usage of a random set R in Liu's encryption algorithm (Algorithm 1). Liu's scheme is semantically secure when the size of the plaintext message space $\mathcal{M}$ is bounded, not allowing arbitrarily large plaintexts. The semantic security, as already noted in Sub-section 3.2.1, can be proven by demonstrating that attackers will be unable to guess, with higher probability than $\frac{1}{2}$, whether cyphertexts E is an encryption of plaintext v_1 or v_2 [163]. Using Liu's encryption function, and given that the plaintexts are bounded and the random number set R=$\{r_1, \ldots, r_m\}$ is arbitrarily large, a Liu's scheme cyphertext E will be dominated by the nature of R. Hence, the probability of distinguishing whether E encrypts v_1 or v_2 is asymptotically equal to distinguishing the value of the random numbers used to encrypt E. If the random number has p decimal digits, then, the probability of distinguishing a particular value of a random numbers is $O(\frac{1}{10^p})$ due to the uniformity of sampling used to draw from the random set R (Algorithm 1). Therefore, the probability of guessing correctly is negligible given that $\frac{1}{10^p} < \frac{1}{2}$.

3.3.1.4 Computational Aspects: Cyphertext Inflation

The number of sub-cyphertexts m, and the number of homomorphic multiplications, will have an impact on the computational cost and memory resources required when using Liu's scheme. Therefore, in this sub-section, some important computational aspects of the Liu's FHE scheme are presented. The sub-section provides details on the processing time for: (i) key generation, (ii) data encryption and decryption; and (iii) the conducting of homomorphic operations ($\oplus$, $\otimes$ and $\circledast$) for different values of m and different dataset sizes (number of attributes). Liu's FHE key generation is a one-time process that increases in time with m. Experiments, not presented here, demonstrated that Liu's secret key $SK(m)$ can be generated in negligible time. For example, when the number of sub-cyphertexts $m = 3$, 9 and 15 the secret key $SK(m)$ was generated in $0.85ms$, $0.87ms$ and $0.89ms$, respectively. The experiments reported in Figure 3.2 show that, the processing time for all homomorphic operations increases linearly with the size of the data and the value of m. Encryption and decryption are one-time operations and so contribute relatively little to the overall runtime. Homomorphic addition, $\oplus$, and

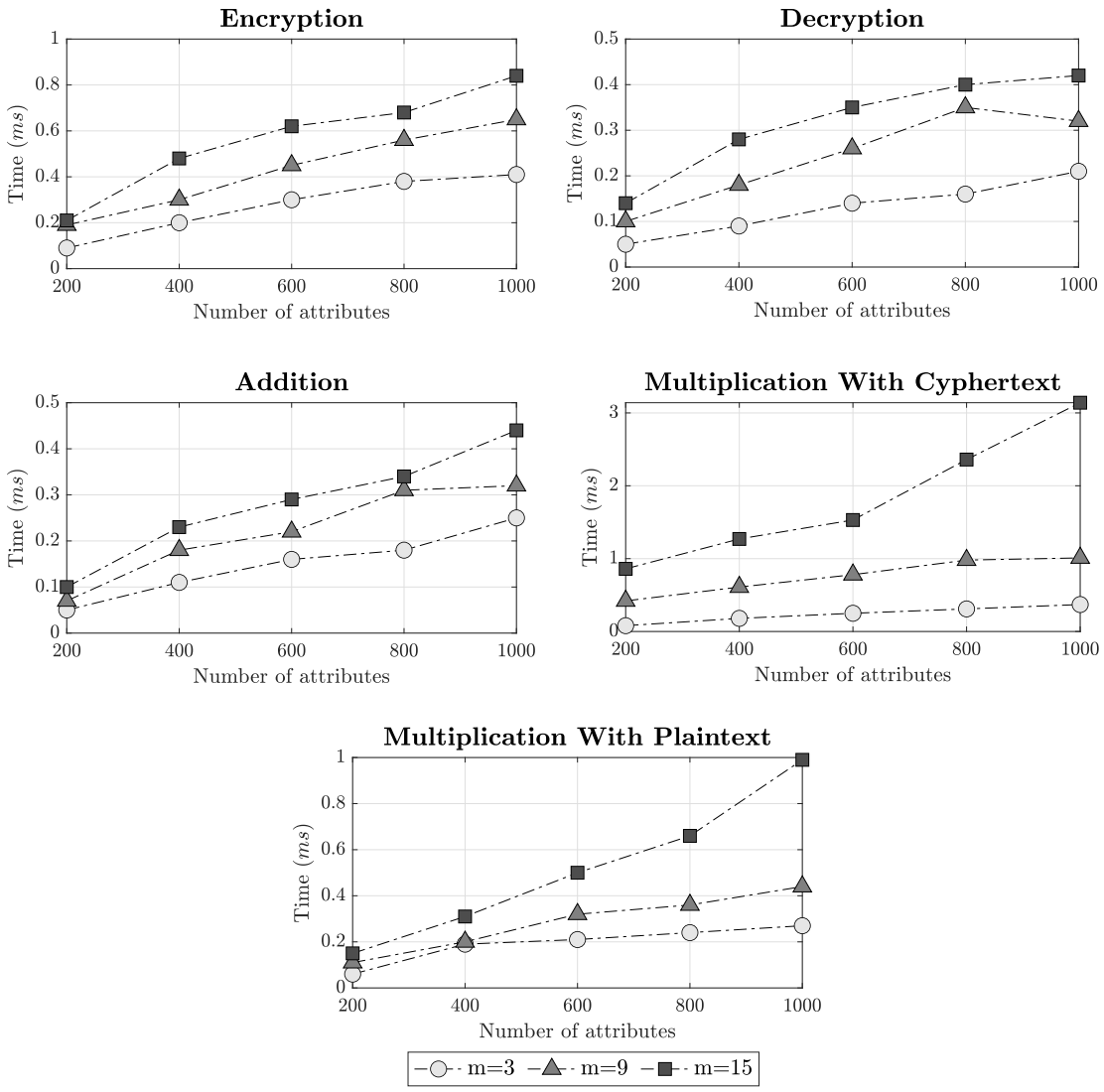


FIGURE 3.2: The computational cost of the homomorphic operations supported by Liu's FHE scheme for different numbers of attributes in a record: [**top left**] `Encrypt`(v); [**top right**] `Decrypt`(E); [**middle left**] addition: $E_1 \oplus E_2$; [**middle right**] two cyphertexts multiplication: $E_1 \otimes E_2$ and [**bottom**] cyphertext multiplication with plaintext $c \circledast E_1$. The given runtime values are averages over 10 iterations

multiplications, $\otimes$ and $\circledast$, are clearly much slower than the corresponding regular plaintext arithmetic operations. Homomorphic multiplication ($\otimes$) is more expensive than cypher addition and multiplying a cyphertext with a plaintext value as evidenced in the experimental results reported in Figure 3.2. The complexity of each homomorphic operation increases with m as well as the number of attributes featured in data records.

The high computational costs are a reflection of the increase in size when data is encrypted. Liu's scheme homomorphic multiplication, as described in Sub-section 3.3.1.2, performs cyphertexts multiplication by determining the outer product of the two cyphertexts to be multiplied. Therefore, for one multiplication the cyphertext size, and consequently the required memory, is increased from m to m^2 and continues to exponentially increase with each homomorphic multiplication operation ($\otimes$). In this thesis this issue is referred to as cyphertext inflation. Figure 3.3 shows the degree of inflation, in the form of a 3D histogram, for a range of values of m, $m = \{3, 9, 15, 21, 27, 33\}$, when

multiplying with other cyphertexts. In the figure, the vertical axis indicates the number of sub-cyphertexts resulting from homomorphic multiplication, while the horizontal axises represent the number of sub-cyphertexts involved in the multiplication. For the work presented in this thesis the value of m was fixed at $m = 3$ because it guarantees the best performance as shown in the experiments reported in this sub-section.

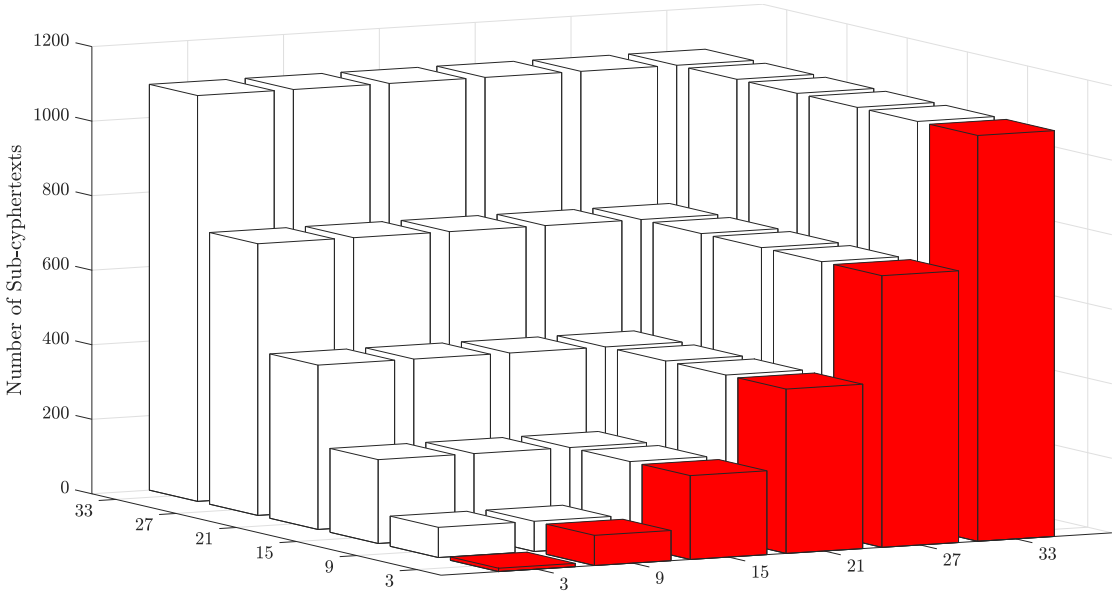


FIGURE 3.3: 3D histogram showing the increase in the number of sub-cyphertexts as a result of multiplying cyphertexts using homomorphic multiplication ($\otimes$) with different values of m

3.3.2 Paillier Partial Homomorphic Encryption Scheme

The second HE scheme adopted with respect to the work presented in this thesis is the Paillier scheme [54]. This is a probabilistic, Additive HE scheme, which relies on the Decisional Composite Residuosity (DCR) assumption. The scheme is defined over the positive integer message space ($\mathcal{M}= \mathbb{Z}_N^+$) in such a way that the values that may be encrypted range from 0 to $N - 1$ where N is the RSA modulus. The following, Sub-sections give details of the scheme. Sub-section 3.3.2.1 considers scheme key generation, encryption and decryption. Sub-section 3.3.2.2 presents the homomorphic properties of the Paillier PHE scheme. The Sub-section 3.3.2.3 reviews the security of the scheme. The section is concluded, Sub-section 3.3.2.4, with details on the computational costs, and the associated dependency on the scheme's parameters.

3.3.2.1 Key, Encryption and Decryption

The Paillier public key is generated by randomly selecting two different prime numbers (p and q), with k bit-length, where k is a scheme security parameter. In the literature [164] it is recommended that the value of k be either 2048 or 3072 bits. The Rivest-Shamir-Adleman (RSA) modulus (N) is then calculated by multiply the randomly selected two prime numbers, $N = p \times q$. The value of N is then considered as a scheme public key. The secret key is given by (λ, μ) which is calculated as given in Equation 3.6 where LCM is a *Least Common Multiple* function, L is a function defined as $L(x) = \frac{x-1}{N}$ and g is a non-zero integer of order divisible by N. The value of g should be small for performance reasons, $g = 2$ was used with respect to the work presented in this thesis.

$$\begin{aligned} \lambda = &\quad LCM(p-1, q-1) \\ \mu = &\quad (L(g^{\lambda} \pmod{N^2}))^{-1} \pmod{N} \end{aligned} \tag{3.6}$$

The Paillier scheme will encrypt the plaintext value v to cyphertext e using equation:

$$e = g^v r^N \pmod{N^2} \tag{3.7}$$

where r is a random number drawn from the range (1, $N-1$) such that $GCD(r,N)= 1$ where GCD is the Greatest Common Divisors. The decryption function decodes e to the original plaintext value v as per equation 3.8.

$$v = L\Big(e^{\lambda} \pmod{N^2}\Big)\mu \pmod{N} \tag{3.8}$$

3.3.2.2 Homomorphic Properties

The Paillier PHE scheme has an additive homomorphic feature that maps plaintext addition (+) to cypher multiplication ($\otimes$) as per Equation 3.9, where $v_1, v_2 \in \mathbb{Z}_N^+$ are encrypted using Paillier (Equation 3.7) to give e_1 and e_2 respectively. Equation 3.10 demonstrates this mapping.

$$e_1 \otimes e_2 \pmod{N^2} = v_1 + v_2 \tag{3.9}$$

$$\begin{aligned} \texttt{Encrypt}(v_1) \otimes \texttt{Encrypt}(v_2) &= \Big(g^{v_1} r_1^N \pmod{N^2}\Big) \times \Big(g^{v_2} r_2^N \pmod{N^2}\Big) \\ &= g^{v_1+v_2}(r_1 r_2)^N \pmod{N^2} \\ &= \texttt{Encrypt}(v_1 + v_2) \end{aligned} \tag{3.10}$$

In addition to the additive homomorphic property, Pailliler PHE has some additional homomorphic properties, which allow multiplication with plaintexts. A plaintext value c can be multiplied with a cyphertext e, that encrypts a plaintext value v, as in Equation 3.11. This feature can be used to support cyphertexts subtraction, as in the case of Liu's FHE scheme, by multiplying a cyphertexts by -1 and then adding the result with another cyphertext.

$$c \circledast e = e^c \pmod{N^2} = c \times v \tag{3.11}$$

3.3.2.3 Security

The Paillier encryption scheme [54], as noted above, is a probabilistic encryption scheme that has been demonstrated to be semantically secure if the DCR assumption holds. The probabilistic property is incorporated by using a random value r in the encryption equation (Equation 3.7). The DCR assumption states that, given a composite integer N and an integer $x \in \mathbb{Z}$, it is computationally hard to decide whether there is a $y \in \mathbb{Z}$ such that $x \equiv y^N \pmod{N^2}$. This is related to the "hardness" to factorise N, when N is the product of two large primes (p and q). This makes the probability of distinguishing cyphertexts hard, and thus Paillier is demonstratively semantically secure.

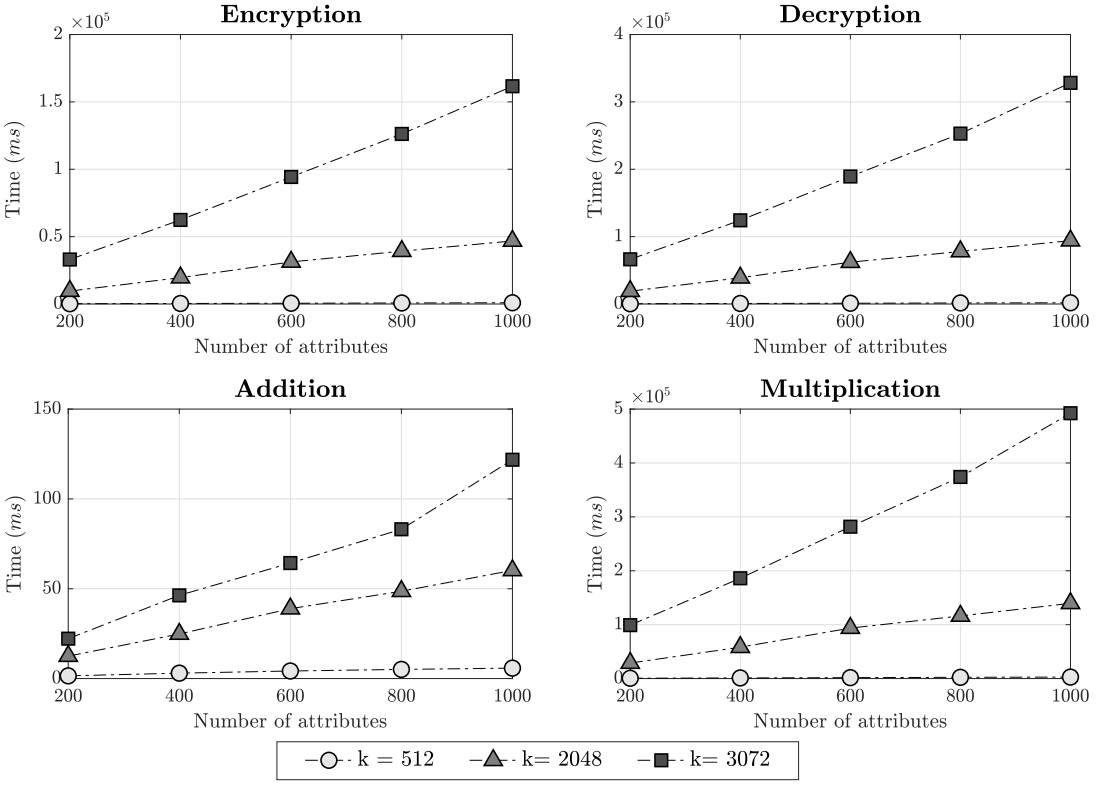


FIGURE 3.4: The computational cost of Paillier PHE scheme homomorphic operations for different numbers of attributes in a record: **[top left]** `Encrypt`(v) **[top right]** `Decrypt`(e) **[bottom left]** addition: $e_1 \otimes e_2$; **[bottom right]** multiplication: $c \circledast e$. The given runtime values are averages over 10 iterations

3.3.2.4 Computational Aspects

The performance (efficiency) of the Paillier PHE scheme and its HE mathematical properties relies on the selected security parameter k. Recall that the value of k represents the bit-length of the two prime numbers, p and q, used to generate the public key N. Therefore, in this sub-section, some computational aspects of Paillier PHE scheme are presented. The sub-section provides details on the processing time for: (i) data encryption and decryption; and (ii) the homomorphic operations ($\otimes$, $\circledast$) for different value of the security parameter (k) and different dataset sizes (number of attributes). The experiments conducted considers three different values of k; $k = \{512, 2048, 3072\}$. The results are shown in Figure 3.4. The figure shows that the processing time for all homomorphic operations increases linearly with the size of the data and k. However, the data encryption and decryption are one-time operations that do not introduce a significant overhead to the overall runtime. The homomorphic operations, $\otimes$ and $\circledast$, are clearly much slower than the corresponding regular plaintext arithmetic operations, although the multiplication $\circledast$ is more expensive than the cypher addition as evidenced in the experimental results reported in Figure 3.4.

3.4 Property Preserving Encryption

There are several encryption schemes that preserve plaintext properties over cyphertexts. These are generically referred to as Property Preserving Encryption (PPE) schemes [165]. The HE schemes considered in the foregoing section are a special form of PPE

scheme [165], schemes that preserve arithmetic operations. However, there are other properties that may be preserved using PPE schemes. Examples of such schemes include: (i) Deterministic Encryption (DET), (ii) Distance Preserving Encryption (DPE), (iii) Asymmetric Scalar Product Encryption (ASPE) and (iv) Order Preserving Encryption (OPE); examples that could all provide potential operations required for PPDM. These schemes have different performance and security characteristics as they permit a range of different operations [18]. In the following sub-sections the above listed PPE schemes are considered in further detail, Sub-section 3.4.1 to 3.4.4. Each sub-section describes the nature of the relevant schemes along with the associated potential security threats. OPE is described in further detail than the others because it is of particular relevance with respect to the work presented in this thesis as will become clear later in the thesis.

3.4.1 Deterministic Encryption

The Deterministic Encryption (DET), as noted above, is a type of encryption whereby plaintext values are encoded to the same cyphertext each time they are presented using a prescribed encryption key. Therefore, it allows for equality checking over encrypted data. Formally, the cyphertexts e_1 and e_2, generated using a DET scheme, can be directly compared using $e_1 == e_2$ or $e_1 \neq e_2$ without having access to the original plaintext values. From the literature, only a few DET schemes also have homomorphic properties, two examples are: (i) the Rivest-Shamir-Adleman (RSA) without padding scheme [140] which allows homomorphic multiplication as well as equality checking, and (ii) the Modified RSA Encryption Algorithm (MREA) scheme [123] which permits addition mapped as multiplication over cyphertexts as well as equality checking. DET schemes have been widely adopted in many application domains, especially with respect to secure outsourced database storage and secure SQL query resolution over encrypted data, as in the case of the CryptDB project [148].

However, DET schemes have serious security vulnerabilities. DET schemes leak a considerable amount of information. Simple frequency analysis or dictionary attack can often be used to attack such schemes (the reader may wish to refer back to Sub-section 2.5.2 where the nature of dictionary attacks was described). The *frequency analysis attack* exploits the absence of randomness during the encryption process whereby the same cyphertext is produced when the same plaintext value is encrypted multiple times. Therefore, with basic information about the message space (for example the frequency of a particular attribute value), attacks can match the frequencies of cyphertexts with the frequencies of plaintexts data. The *dictionary attack* is typically applied to semantically insecure schemes where the attacker has access to the public key. In this case, the attacker can create their own dictionary by simply encrypting all potential plaintexts using the scheme's public key. This vulnerability renders DET schemes to be inappropriate for PPDM and by extension DMaaS.

3.4.2 Distance Recoverable Encryption

Distance Recoverable Encryption (DRE) schemes provide a straightforward solution for many data clustering and classification algorithms that required distance comparison. Given an encryption function `Encrypt` and key Key, if $\texttt{Encrypt}(v_1, Key)$ is the encryption of data value v_1 and $\texttt{Encrypt}(v_2, Key)$ is the encryption of data value v_2, the scheme is denoted as a DRE scheme if and only if there exists a computational procedure $\mathcal{F}$ such that:

$$\mathcal{F}\Big(\texttt{Encrypt}(v_1, Key), \texttt{Encrypt}(v_2, Key)\Big) = v_1 - v_2 \tag{3.12}$$

The scheme preserved distances between any two encrypted records, and thus the calculated distance over encrypted data records are the same as that between the corresponding original records. Distance Preserving Transformation (DPT) is an example of DRE, which has been adopted in the context of secure k-Means data clustering and Chameleon hierarchical data clustering in [166]. However, as shown in [167, 168], DRE schemes have serious security vulnerabilities if some of the plaintext data values are known to an attacker, or if the mapping between some plaintexts and cyphertexts is known; the Known-Plaintext Attack (KPA) and the Chosen-Plaintext Attack (CPA). Using either of these attacks the attacker can recover the original data values with very high confidence via knowledge of the mutual distances between data records together with the probability distribution from which they are drawn. Therefore, any data transformation/encryption that preserve the "actual or real distances" between data records may be vulnerable to the disclosure of private information [169]. Therefore, the DRE schemes were deemed not sufficiently secure for the purpose of PPDM.

3.4.3 Asymmetric Scalar Product Encryption

The weakness of DRE comes from the fact that the attacker is able to recover "real" distances from the encrypted data. The Asymmetric Scalar Product Encryption (ASPE) schemes address this by allowing distance comparison between two data records in such a way that the real distance cannot be recovered. ASPE has been used in [23, 167, 170] to implement k-Nearest Neighbour (kNN) classification. Using an ASPE scheme the two values to be compared are encrypted using two distinct matrices. One is encrypted using what is known as an "invertible matrix" and the other using an "inverted matrix", thus avoiding an attack based on distance preservation between unencrypted data and encrypted data, hence avoiding distance recovery [167]. More formally, given an encryption function `Encrypt` and key Key the ASPE scheme allows the secure determination of whether value v_3 is nearer to value v_1 or v_2 by directly comparing the cyphertexts: $\texttt{d}(\texttt{Encrypt}(v_1, Key)$ and $\texttt{Encrypt}(v_3, Key))$, and $\texttt{d}(\texttt{Encrypt}(v_2, Key)$ and $\texttt{Encrypt}(v_3, Key))$. However, the requirement of using two encrypted versions of the data renders the scheme impractical for large scale PPDM.

3.4.4 Order Preserving Encryption

The Order Preserving Encryption (OPE) approach ensures that the numerical order of plaintexts persists in the generated cyphertexts. Therefore, when plaintexts $v_1 < v_2$ the encryption scheme preserves the relation $\texttt{Encrypt}(v_1) < \texttt{Encrypt}(v_2)$. This feature allows comparison operations to be directly applied over the encrypted data, without decryption, in the same way as on plaintexts using operations such as: $=$, $\neq$, $<$, $>$, $\leqslant$ and $\geqslant$. Therefore, OPE has been adopted in many real applications such as CryptDB [148] developed by the Massachusetts Institute of Technology (MIT), Encrypted BigQuery [171, 172] developed by Google, and other solutions directed at secure database query resolution [51, 148, 173–177]. There has been a significant amount of work on OPE schemes [178–181]. In the remainder of this sub-section a review of some of the most well known OPE schemes is presented, together with their respective efficiency and security attributes.

The first OPE scheme that provided a rigorous level of security was the Boldyreva, Chenette, Lee and O'Neill (BCLO) scheme presented in [178]. The scheme defined a

Random Order-Preserving Function (ROPF) to be used for generating order preserving cyphers. Each cyphertext assignment was made according to the output of their Hyper-Geometric Distribution (HGD) sampling algorithm run on appropriate parameters for each plaintext. However, the scheme was inefficient due to the HGD sampling. Experiments, conducted on an iMac running under the macOS High Sierra operating system with 8GB of RAM using the BCLO Java implementation available at [182], demonstrated that a dataset with 452 records and 279 attributes required 8 hours to encrypt. Due to its functional requirements, the BCLO OPE scheme (as in the case of most OPE schemes) is defined to be deterministic, so as to allow for data equality comparison. This inherent deterministic property, as in the case of other DET schemes, results in leakage of a lot of information which renders the scheme to be vulnerable to the deterministic attacks presented early in this section (Sub-section 3.4.1). The research presented in [179] was directed at improving the performance of ROPF by reducing the number of HGD sampling invocations in the BCLO scheme. However, in [183, 184] it was shown that ROPF, as used in the BCLO scheme, exposed more than the order of the data values; at least half of the plaintext bits were also revealed. Therefore, BCLO schemes cannot satisfy the security requirements expected for PPDM. BCLO schemes are not secure against CPAs and are vulnerable to simple statistical analysis which may reveal confidential data.

The mutable Order Preserving Encryption (mOPE) proposed in [180] was designed to be used by TPDMs and addressed the threat of CPAs. The mOPE scheme changes the generated cyphertexts, using a small number of pre-encrypted cyphertexts, whenever an already-encrypted plaintext value is presented again. The scheme works by constructing a balanced search tree (an AVL tree) containing all of the plaintext values encrypted using a DET scheme. The order-preserving encoding of a value *is the path from the root to that plaintext value in the search tree.* Tree paths are denoted using a binary encoding where 0 denotes the left child edge and 1 the right child edge. Thus, if x is less than y, the path to x will be to the left of y (as the binary encodings used increase from left to right in the tree). Each time a value is encrypted it is inserted into the tree at an appropriate tree node using an interactive protocol with the data owner. Dynamic Order Preserving Encryption (DOPE), presented in [181], is founded on the mOPE scheme presented in [180]. The DOPE scheme modified the mOPE scheme by introducing a trusted proxy between the data owner and the TPDM to manage the interactive protocol for inserting new encrypted data records.

Generally speaking, the above mentioned OPE schemes are all deterministic encryption schemes (one-to-one mapping). An adversary who is either aware of the message space or gathers statistical data concerning the message space (data distribution and frequency) is capable of building a mapping between the actual and the encrypted values. The leakage associated with the usage of DET OPE schemes discussed in [185] and techniques for minimising such leakage have been suggested. One of these technique is to introduce *randomisation* to the OPE encryption function, and trade some of its functionality (equivalence checking) to enhance security. This idea was implemented in [51, 186] where a one-to-many order preserving mapping function was introduced. In [51], the secret key consists of two integers, a and b, and the encryption of a value v is $a \times v + b + \epsilon$, for some randomly chosen noise $\epsilon \in [0, a-1]$, small enough to preserve the ordering. However, the a and b variables (the secret key) are the same across all the encrypted data and this makes the scheme vulnerable to statistical attacks as demonstrated in [187]. An alternative OPE scheme is presented later in this thesis that address the disadvantages associated with the above OPE schemes, thus rendering it suitable for PPDM and by extension DMaaS.

3.5 Summary

This chapter has presented the cryptographic background concerning HE schemes, and some PPE schemes, that are relevant to PPDM. The chapter commenced by introducing the basic cryptography terminology. The concept of HE was discussed, and an overview of a range of different HE schemes and their properties was presented. The limitations of HE schemes, pertinent to the research presented in this thesis, were highlighted. The presented HE limitations included consideration of the techniques that may be adopted to addressing each limitation. Two HE schemes that feature later in this thesis were presented in detail, and their security limitations and computational complexity considered with respect to the central research question of the thesis. The following chapter introduces the first proposed solution to PPDM, founded on the idea of HE and with a focus on reducing the data owner participation (compared to alternative solutions) when delegating data clustering to a TPDM.

Chapter 4

Updatable Distance Matrices

4.1 Introduction

In the previous chapter, a number of preliminaries relevant to cryptography and PPE schemes were presented. As established, there is no scheme that supports all the operations required in the context of PPDM and by extension DMaaS. The solutions that have been proposed to date, and that were reported in Chapter 2, either require a significant element of data owner participation or incorporate unreasonable security assumptions that require the delegation of the secret key to a third party or the splitting of the key among multiple non-colluding parties using the technique of secret sharing. These solutions also have significant disadvantages concerning efficiency; in many cases because of the required communication and computation overhead to securely fulfil the unsupported operations.

In this chapter, the concept of Updatable Distance Matrices (UDMs) is presented that dramatically reduces the data owner participation and thus the communication and computational overhead which is a feature of previous solutions. The focus is data clustering using a TPDM because clustering is arguably the simplest data mining application to implement over encrypted data, a number of examples taken from the literature were cited in the previous chapter [19–22, 33–37]. A Secure k-Means (Sk-Means) data clustering algorithm, founded on the UDM concept, is therefore also proposed which operates over encrypted data. This algorithm is the first of a number of secure data mining algorithms proposed in this thesis designed to outsource data mining tasks to a TPDM with only a limited data owner participation. In the introduction to this thesis two kinds of secure DMaaS scenarios were identified, the single data owner scenario and the multiple data owners scenario. The work presented in this chapter is directed at the single data owner scenario, the multiple data owners scenario will be considered in later chapters. The reader may wish to refer back to Sub-section 2.3.1 where the data privacy preservation requirements for the single data owner scenario were presented.

The main idea underpinning the UDM was to provide the TPDM with a mechanism to accurately compare cyphertexts without involving the data owner. A further feature of the UDM is that it can be readily updated as new cyphertexts are generated using the HE properties of the scheme. The updating process allows the data comparison to be performed between newly generated cyphertexts and the outsourced data. The updating process uses the concept of a Shift Matrix which requires very limited data owner participation, substantially less than that required in the case of comparator approaches [19–22, 33–37]. In the case of the proposed Sk-Means the TPDM still needs to have access to the original data to assign records to clusters. The data privacy in this case is preserved using data encryption applied at the attribute level. Liu's FHE scheme

[53], previously presented in Chapter 3, was used for this purpose. The intuition was to modify k-Means to work with the UDM concept and replace the mathematical operations with the HE properties of Liu's FHE scheme. The evaluation criteria were data clustering accuracy, clustering efficiency and security. For the proposed Sk-Means data clustering algorithm to be said to be operating correctly it should produce comparable or similar clustering results to those produced in the case of the standard k-Means algorithm using unencrypted data.

The rest of this chapter is structured as follows. Section 4.2 provides a detailed description of the proposed UDM mechanism. This is followed, in Section 4.3, by presentation of the proposed Secure k-Means (Sk-Means) data clustering algorithm designed to operate using encrypted data and founded on the UDM concept. In Section 4.4 the results of an extensive evaluation concerning the UDM concept and the proposed Sk-Means algorithm, using benchmark datasets from the UCI machine learning repository, are presented. Finally, Section 4.5 provides a summary of the material presented in this chapter.

4.2 Updatable Distance Matrices

This section introduces the Updatable Distance Matrices (UDMs). The section provides details concerning the nature of UDMs, their utilisation in the context of k-Means secure clustering and how such matrices are updated when new clustering centroids are generated (using the properties of Liu's FHE scheme). This section thus comprises two sub-sections. The first, Sub-section 4.2.1, is concerned with the nature of UDMs and their usage in the context of k-Means clustering; and the second, Sub-section 4.2.2, with the updating process that uses the concept of a Shift Matrix (SM). Procedural details concerning the practical generation of UDMs and the proposed secure Sk-Means algorithm are given in the following section, Section 4.3.

4.2.1 The Updatable Distance Matrices

Data encryption using HE schemes, as in the case of standard encryption schemes, typically involves translating from plaintexts form to cyphertexts form in a way that any data ordering that might have been a feature of the plaintext data is not transferred to the generated cyphertexts. Therefore data comparisons over the cyphertexts cannot be performed. Data clustering, as in the case of many data mining algorithms, requires substantial data comparison. To address this issue many proposed solutions [19, 20, 35–37], where the secret key is not shared (which raises a security concern), require that the data comparisons be performed by the data owner either by: (i) the data owner decrypting the cyphertexts, comparing the results and then sending the results of the comparisons back to the TPDM; or (ii) in the case of the multiple data owner scenario, by using a SMPC comparison protocol such as YMPP or Cachin's scheme [30, 104]. The UDM concept was designed to allow the TPDM to compare encrypted data records without recourse to data owners or complex SMPC comparison protocols. Although, as will become clear, in the case of the proposed Sk-Means algorithm data owner participation is still required to update the UDM once new cluster centroids have been generated as the Sk-Means algorithm progresses; however, as will also become clear, this data owner participation is minimal in comparison with existing solutions.

A UDM is a 3D matrix that holds (unencrypted) distances between the attribute values in each record with the corresponding attributes values in every other record. Therefore, the first and the second dimension of a UDM correspond to the number of

records in the dataset and the third dimension corresponds to the number of attributes featured in the data. More formally, given a dataset $D = \{r_1, \ldots, r_n\}$ where each record r_x features a set of values that correspond to a set of attributes $A = \{r_{x,1}, \ldots, r_{x,a}\}$; the UDM associated with the dataset D will measure $n \times n \times a$. The matrix UDM ($\mathbf{U}$) will comprise a set of elements $u_{x,y,z}$ where x and y reference record identifiers and z an attribute identifier. The value in each element $u_{x,y,z}$ is a plaintext (unencrypted) value calculated by the data owner, as $r_{x,z} - r_{y,z}$. The element $u_{x,y,z}$ thus expresses the difference between the value of attribute z in record x and the value of the same attribute in record y. A UDM ($\mathbf{U}$) has the form shown in Equation 4.1 (the third dimension represented as a list). It is symmetric about the leading diagonal so only the values for the lower (or the upper) 3D triangle of the matrix needs to be considered.

$$\mathbf{U} = \begin{bmatrix} (u_{1,1,1}, \cdots, u_{1,1,a}) & (u_{1,2,1}, \cdots, u_{1,2,a}) & \cdots & (u_{1,n,1}, \cdots, u_{1,n,a}) \\ \vdots & \vdots & \ddots & \vdots \\ (u_{n,1,1}, \cdots, u_{n,1,a}) & (u_{n,2,1}, \cdots, u_{n,2,a}) & \cdots & (u_{n,n,1}, \cdots, u_{n,n,a}) \end{bmatrix} \tag{4.1}$$

A UDM $\mathbf{U}$ and the encrypted dataset D', generated by the data owner, can collectively provide a solution to data clustering. They allow the TPDM to cluster the encrypted data records. The idea is illustrated in this chapter using the k-Means clustering approach. This will operate as follows. On the first iteration the first k encrypted records in D' will be selected by the TPDM to represent the first iteration cluster centroids $Cent'_1 = \{cent'_{1_1}, \ldots, cent'_{1_k}\}$. The remaining encrypted data records in D' are added to the nearest cluster by comparing the distances between each record and the cluster centroids. The content of $\mathbf{U}$ is used for this purpose. As already noted the first dimension of $\mathbf{U}$ represents the data record identifiers, the second the centroids and the third the attribute identifiers. On each occasion that the data owner causes the k-Means clustering to be executed the data owner must specify the number of cluster k, only the first k elements in the second dimension of $\mathbf{U}$ are then used, the remaining can be ignored. Thus if a maximum value for k is known in advance the size of the UDM can be limited. As in the case of standard k-Means, the data records that have been added to cluster in the first iteration are used to calculate the next iteration centroids. The HE properties of Liu's FHE scheme, used to encrypt D (presented in Chapter 3), are used to re-calculate encrypted cluster centroids to arrive at new set of centroids $Cent'_2 = \{cent'_{2_1}, \ldots, cent'_{2_k}\}$. On the next iteration, the TPDM needs to re-determine the cluster content by comparing each encrypted record in D' with the new set of centroids $Cent'_2$. This cannot be achieved directly as Liu's FHE scheme does not support data comparison and $\mathbf{U}$ holds the distances with respect to the first iteration centroids. One solution would be to return $\mathbf{U}$ to the data owner along with the newly calculated cluster centroids for an update. However, this solution will introduce a substantial communication and computational overhead. Instead, a novel updating process is proposed that allows $\mathbf{U}$ to be updated, regardless of the dataset size, with only very limited data owner participation as presented in the following sub-section.

4.2.2 Updating Process

The updating process aims to effectively and efficiently update a UDM in the context of the Sk-Means algorithm introduced in the preceding sub-section. Algorithm 3 shows the pseudo code for the updating process. The inputs are the UDM $\mathbf{U}$ to be updated, the ith iteration clusters centroids ($Cent'_i$) and the $i+1$th iteration cluster centroids ($Cent'_{i+1}$). The first step is for the TPDM to calculate what is referred to as a *Shift Matrix* S' (line

2) that represents the distances between cluster centroids of two consecutive iterations, $Cent'_i$ and $Cent'_{i+1}$. The distances are calculated using the cyphertext subtraction ($\ominus$) property of Liu's FHE scheme. More formally a shift matrix is a 2D matrix of the form (k is the k-Means clustering parameter):

$$S' = \begin{bmatrix} s'_{1,1} & s'_{1,2} & \cdots & s'_{1,a} \\ \vdots & \vdots & \ddots & \vdots \\ s'_{k,1} & s'_{k,2} & \cdots & s'_{k,a} \end{bmatrix}$$

Each element $s'_{x,y}$ holds the difference ("offset" or "shift" value) between the yth attribute in centroid $cent'_{i_x} \in Cent'_i$ and the yth attribute in centroid $cent'_{i+1_x} \in Cent'_{i+1}$; thus the difference between the iteration i centroids $Cent'_i$ and newly calculated iteration $i+1$ centroids $Cent'_{i+1}$. The values are encrypted using Liu's FHE scheme. The calculated shift matrix S' is then sent to the data owner for decryption to arrive at the plaintext shift matrix S (line 3); this is the only data owner participation that is required in the proposed Sk-Means algorithm. The matrix S, once received by the TPDM, is then used to update $\mathbf{U}$ (line 4) using Equation 4.2. Recall that k-Means uses only the first k columns in $\mathbf{U}$ and thus the updated process is only directed at these k columns. Equation 4.3 demonstrates the value held in element $u_{x,y}$ before and after applying the updating process to prove the correctness of the updating process. The first line in the equation shows the initial value held in $u_{x,y}$ that represents the distance between record r_x and the yth centroid of the current clustering iteration. The second line of the equation shows the value of shift matrix element s_y added during the updating process. The updated $u_{x,y}$ element is shown in the last line of the equation. The updated UDM element will hold the distances between data records and new clustering iteration centroids. The algorithm exits (in line 5) with the UDM $\mathbf{U}$ that has been updated and is ready for the next iteration of the algorithm and a decrypted shift matrix S.

$$\begin{bmatrix} (u_{1,1,1} + s_{1,1}, \cdots, u_{1,1,a} + s_{1,a}) & \cdots & (u_{1,k,1} + s_{k,1}, \cdots, u_{1,k,a} + s_{k,a}) \\ \vdots & \ddots & \vdots \\ (u_{n,1,1} + s_{1,1}, \cdots, u_{n,1,a} + s_{1,a}) & \cdots & (u_{n,k,1} + s_{k,1}, \cdots, u_{n,k,a} + s_{k,a}) \end{bmatrix} \tag{4.2}$$

$$\begin{aligned} u_{x,y} &= r_x - cent_{i_y} \\ u_{x,y} + s_y &= r_x - cent_{i_y} + (cent_{i_y} - cent_{i+1_y}) \\ u_{x,y} &= r_x - cent_{i+1_y} \end{aligned} \tag{4.3}$$

Algorithm 3 UDM updating process

1: **procedure** UPDATEUDM($\mathbf{U}$,$Cent'_i$,$Cent'_{i+1}$)
2: $S' = Cent'_i \ominus Cent'_{i+1}$
3: $S = S'$ decrypted by data owner (Algorithm 2 in Chapter 3)
4: $\mathbf{U} = \mathbf{U} + S$ ▷ Equation 4.2
5: **Exit** with $\mathbf{U}$ and S
6: **end procedure**

4.3 Secure Data Clustering

This section presents the secure data clustering process founded on the concept of UDMs. The process has two parts, a data preparation part and a clustering part. The first is

conducted by the data owner and is detailed in Sub-section 4.3.1, while the second is conducted by the TPDM and is detailed in Sub-section 4.3.2.

4.3.1 Data Preparation for Outsourcing

This section presents the preparation step, conducted by data owners, to securely outsource their data to the TPDM who provides DMaaS. The data preparation comprises: (i) data preprocessing, (ii) data encryption and (iii) UDM calculation. The pseudo code for the data outsourcing process is given in Algorithm 4. The inputs are the raw dataset to be outsourced $RawD$, the number of attributes a that features in $RawD$ and Liu's FHE scheme parameter m as presented in Sub-section 3.3.1. During the data preprocessing the data owner casts any categorical attributes values to appropriate discrete counterparts to arrive at a preprocessed dataset D (line 2). The Liu's scheme secret key $SK(m)$ is then generated by the data owner following the process presented earlier in Chapter 3. The reader may wish to refer back to Sub-section 3.3.1 where Liu's FHE scheme was presented. The empty dataset D' is then created in line 4 to hold the encrypted dataset. The preprocessed dataset D is then encrypted attribute-wise in lines 5 to 7 using Algorithm 1 given in Chapter 3. The next preparation step is to calculate the UDM $\mathbf{U}$ (lines 8 to 15). Only the lower 3D triangle is calculated in this algorithm. The UDM calculation commences by dimensioning the desired matrix $\mathbf{U}$ in line 8 and then populating it (lines 9 to 15); recall from Sub-section 4.2.1 that $r_{x,z}$ is the value of the zth attribute in the feature set describing the xth record in the dataset, while $r_{y,z}$ is the value of the zth attribute in the feature set describing the yth record in the dataset. The output from the data preparation step is the encrypted dataset D' and the UDM $\mathbf{U}$, ready to send to the TPDM.

Algorithm 4 Data preparation for outsourcing process when using the UDM concept

```
1:  procedure OutsourceData(RawD,a,m)
2:      D = Dataset RawD converted to numeric dataset where necessary
3:      SK(m) = Liu's scheme secret key generated as in Sub-section 3.3.1
4:      D' = ∅
5:      for all r ∈ D do
6:          D' = D' ∪ Encrypt(r, SK(m))                    ▷ Algorithm 1
7:      end for
8:      U = Empty UDM dimensioned according to |D|, |D| and a
9:      for x = 1 to x = |D| do
10:         for y = 1 to y = x do
11:             for z = 1 to z = a do
12:                 u_{x,y,z} = r_{x,z} − r_{y,z} (u_{x,y,z} ∈ U)
13:             end for
14:         end for
15:     end for
16:     Exit with D' and U
17: end procedure
```

4.3.2 Secure k-Means

The Secure k-Means (Sk-Means) data clustering algorithm that operates using the concept of UDMs over encrypted data is given by the pseudo code presented in Algorithm 5. Most of the algorithm is executed by the TPDM following a process very similar to

standard k-Means as described in [46]. The inputs are: (i) the encrypted dataset D', (ii) the UDM $\mathbf{U}$ and (iii) the desired number of clusters k to be produced using the k-Means as prescribed by the data owner. The output will be a set of clusters C. The algorithm commences by dimensioning the set of cluster $C = \{c_1, \ldots, c_k\}$ (line 2). The first k records in the encrypted dataset D' are then assigned to cluster C one per cluster in line 3. The iteration counter, i, is then initialised (line 4). The set of current iteration cluster centroids $Cent'_i = \{cent'_{i_1}, \ldots, cent'_{i_k}\}$ is then defined in line 5 and initialised with the first k encrypted records in D'. The remaining encrypted data records are then assigned to clusters using a call to the *populateClusters* sub-procedure (line 6) give in lines 18 to 28. The *populateClusters* sub-procedure inputs are: (i) the record number rid in D' where the process needs to start from, (ii) the UDM $\mathbf{U}$, (iii) the set of cluster so far C, (iv) the encrypted dataset D' and (v) the set of current iteration cluster centroids ($Cent'_i$). The *populateClusters* sub-procedure uses $\mathbf{U}$ to determine the similarity between the cluster centroids in $Cent'_i$ and the remaining records in D' starting with record r'_{rid} and continuing to record $r_{|D'|}$. Equation 4.4 is used to derive the distance between r'_x and r'_y (where $r'_x \in D'$, and $r'_y \in Cent'_i$ $(1 \leqslant y \leqslant k)$.

$$sim(\mathbf{U}, r'_x, r'_y) = \sum_{z=1}^{z=a} |\mathbf{u}_{x,y,z}| \tag{4.4}$$

Once all records have been assigned to a cluster the clusters centroids are re-calculated (line 7) using the *CalculateCentroids* sub-procedure given at the end of the algorithm (lines 29 to 44). The encrypted records belonging to the same cluster (c_j) are used to calculate the cluster centroid (lines 33 to 41). The values of each attribute for the records in a cluster are added using homomorphic addition, $\oplus$ (line 36) and then multiplied by $(\frac{1}{|c_j|})$ using homomorphic multiplication $\circledast$, where $|c_j|$ is the number of records in cluster c_j, to arrive at a set of new cluster centroids (line 40). The *CalculateCentroids* sub-procedure exits with a new set of cluster centroids assigned in line 7 to $Cent'_{i+1}$. The *UpdateUDM* is called in line 8 as presented earlier in Algorithm 3. Next we enter into a loop (lines 9 to 15), which repeats until the shift matrix (S) holds only zero values; the cluster centroids are the same for two consecutive iterations. Note that (not shown in the algorithm) when updating $\mathbf{U}$ we only need to update the first k records in the second dimension representing cluster centroids. The loop commences by creating a new empty cluster set C (line 10). With the new centroids ($Cent'_{i+1}$) the algorithm again assigns all records to C using the populateClusters sub-procedure (line 11) in the same manner as before. In line 12 the iteration counter (i) is updated. The algorithm continues in this manner until the termination condition is arrived at.

4.4 Experimental Results and Evaluation

This section reports on the experimental analysis conducted to evaluate the performance of the UDM concept in the context of the proposed Sk-Means data clustering algorithm presented in this chapter. The objectives of the evaluation were to consider the following criteria:

1. **Data owner participation:** The amount and complexity of the required data owner participation.
2. **Efficiency:** The efficiency of the proposed Sk-Means data clustering, which operates over encrypted data and utilised the UDM, compared to the standard algorithm which operates over unencrypted data.

Algorithm 5 Secure k-Means (Sk-Means) clustering algorithm

procedure Sk-MEANS(D', $\mathbf{U}$, k)
  C = Set of k empty clusters
  Assign the first k records in D' to C (one per cluster)
  $i = 1$
  $Cent'_i$ = Set of first k records in D' (the k cluster centroids)
  $C = populateClusters(k+1, \mathbf{U}, C, D', Cent'_i)$
  $Cent'_{i+1}$ = CalculateCentroids(C,k)
  $\mathbf{U}, S$ = UpdateUDM($\mathbf{U}, Cent'_i, Cent'_{i+1}$) ▷ Algorithm 3
  while $S \neq 0$ **do**
    C = Set of k empty clusters
    $C = populateClusters(1, \mathbf{U}, C, D', Cent'_{i+1})$
    $i = i + 1$
    $Cent'_{i+1}$ = CalculateCentroids(C,k)
    $\mathbf{U}, S$ = UpdateUDM($\mathbf{U}, Cent'_i, Cent'_{i+1}$)
  end while
  Exit with C
end procedure
procedure POPULATECLUSTERS(rid,$\mathbf{U}$,C,D',$Cent'$)
  $id = null$
  for $x = rid$ to $x = |D'|$ **do**
    for $y = 1$ to $y = |C|$ **do**
      $sim = sim(\mathbf{U}, r'_x, cent'_y)$ where $r'_x \in D'$ and $cent'_y \in Cent'$ (Equation 4.4)
      id = cluster identifier with lowest sim value so far
    end for
    $c_{id} = c_{id} \cup r'_x$ ($c_{id} \in C$ and $r'_x \in D'$)
  end for
  Exit with C
end procedure
procedure CALCULATECENTROIDS(C, k)
  $Cent'$ set of k empty centroids
  for $j = 1$ to $j = k$ **do**
    $Average'$ set of a elements initialised by 0
    for $i = 1$ to $i = |c_j|$ **do**
      Rec' = Get(c_j,i) where Rec' is the ith record in c_j
      for $q = 1$ to $q = a$ **do**
        $average'_q = average'_q \oplus rec'_q$
      end for
    end for
    for $q = 1$ to $q = a$ **do**
      $cent'_{j,q} = average'_q \circledast \frac{1}{|c_j|}$
    end for
  end for
  Exit with $Cent'$
end procedure

3. **Accuracy:** The accuracy (correctness) of the resulting cluster configuration produced by the Sk-Means.

4. **Security:** The level of security provided by the proposed solution and the encryption scheme adopted.

The proposed solution was implemented using the Java object oriented programming language. For the evaluation fifteen "classification" benchmark datasets from the University of California Irvine (UCI) machine learning repository [50] were used. Some statistical and descriptive information concerning these datasets are given in Table 4.1. The datasets have integer, real and categorical attributes. Note that the number of classes for each dataset (column 4 in the table) was used as the value for k in the k-Means clustering. The experiments were run using an iMac (3.8 GHz Intel Core i5) running under the macOS High Sierra operating system with 8GB of RAM. The results obtained, in the context of the above objectives are presented in the following four sub-sections, Sub-Sections 4.4.1 to 4.4.4, respectively.

TABLE 4.1: Statistical information for the datasets used to evaluate the UDM concept and Sk-Means

No. UCI Dataset	Num. Records (n)	Num. Attributes (a)	Num. Clusters ($\|C\|$)	Attributes Description
1. Arrhythmia	452	279	16	Categorical, int. & real
2. Banknote Auth.	1372	4	2	Real
3. Blood Trans.	748	4	2	Integer
4. Breast Cancer	198	33	2	Real
5. Breast Tissue	106	9	6	Integer & real
6. Chronic Kidney	400	24	2	Categorical, int. & real
7. Dermatology	366	34	6	Categorical & int.
8. Ecoli	336	8	8	Categorical & real
9. Indian Liv. Pat.	583	10	2	Integer & real
10. Iris	150	4	3	Categorical & real
11. Libras Mov.	360	90	15	Real
12. Lung Cancer	32	56	3	Integer
13. Parkinsons	195	22	2	Categorical & real
14. Pima Disease	768	8	2	Integer & real
15. Seeds	210	7	3	Integer & real

4.4.1 Data Owner Participation

This sub-section considers the results for evaluating the amount and the complexity of data owner participation when utilising the proposed Sk-Means data clustering and the UDM concept. In the proposed solution the data owner will participate in: (i) preparing the data for outsourcing as per Algorithm 4 and (ii) decrypting the shift matrix on each Sk-Means iteration. The data preparation process comprises: (i) data encryption (*Data Enc.*) and (ii) UDM Calculation (*UDM Cal.*). Table 4.2 shows the recorded runtimes associated with these sub-processes using the UCI datasets, and the total time required for data preparation. The table shows that the overall time (column 4) required to prepare the data is negligible, even for the largest dataset, *Arrhythmia*, runtime values of $9.80ms$ and $230.2ms$ were recorded to encrypt the dataset and calculate the UDM. The complexity of the UDM generation can be calculated as being of order $O(\frac{n\times(n+1)}{2} \times a)$;

TABLE 4.2: Time required by data owner to prepare data for outsourcing

UCI Dataset	Data Enc. (ms)	UDM Cal. (ms)	Total UDM App. (ms)
Arrhythmia	9.80	230.2	240.00
Banknote Auth.	1.85	388.0	389.85
Blood Trans.	4.04	62.0	66.04
Breast Cancer	1.70	11.0	12.70
Breast Tissue	0.40	3.0	3.40
Chronic Kidney	2.65	29.0	31.65
Dermatology	1.80	40.2	42.00
Ecoli	0.98	8.8	9.78
Indian Liv. Pat.	1.30	41.8	43.10
Iris	0.20	2.0	2.20
Libras Mov.	4.80	49.8	54.60
Lung Cancer	0.59	1.0	1.59
Parkinsons	1.90	6.3	8.20
Pima Disease	1.44	47.0	48.44
Seeds	0.48	5.0	5.48

if we know that value for k in advance this can be reduced to $O((\frac{k\times(2n-k+1)}{2})\times a)$. Note that in the UDM only the upper or lower 3D triangle needs to be calculated.

The data owner also participated in the updating of the UDM while the Sk-Means data clustering was undertaken by the TPDM. The updating process, as already noted in Sub-section 4.2.2, requires the data owner to decrypt the shift matrix after each Sk-Means iteration. The shift matrix measures $k \times a$, a very small matrix that requires negligible time for decryption. However, in the experiments reported in this thesis, the data owner and TPDM were both hosted on the same machine. In a real life single data owner scenario the data owner and the TPDM will be hosted on different machines and thus there will be a "message passing" overhead. The time for message passing would add to the overall runtime, although this is again not anticipated to add a significant further overhead. The runtime for decrypting the shift matrix was evaluated by considering the effect of the number of attributes featured in the dataset D' and the effect of the number of iterations on the execution time. The results are presented in Figure 4.1 which shows two plots. The first is the runtime against the number of attributes and the second is the runtime against the number of iterations required by the Sk-Means to arrive at the clustering configuration. The two plots show that, as anticipated, the runtime increased with the number of attributes and number of iterations.

4.4.2 Secure Clustering Efficiency

The utilisation of the HE properties of Liu's scheme clearly introduce a computational overhead with respect to the same operations conducted using plaintext data. This sub-section reports on an analysis of the overhead introduced by utilisation of the HE properties in the proposed Sk-Means algorithm. The overhead is measured in terms of the runtime required by the TPDM to cluster the encrypted dataset (this will also include the time required for decrypting the shift matrices on each Sk-Means iteration). The runtime required will be compared with the runtime to clustering the datasets using the standard k-Means algorithm. The results are shown in columns 2 and 5 of Table 4.3 (note that the runtimes for Sk-Means does not include any data owner preparation

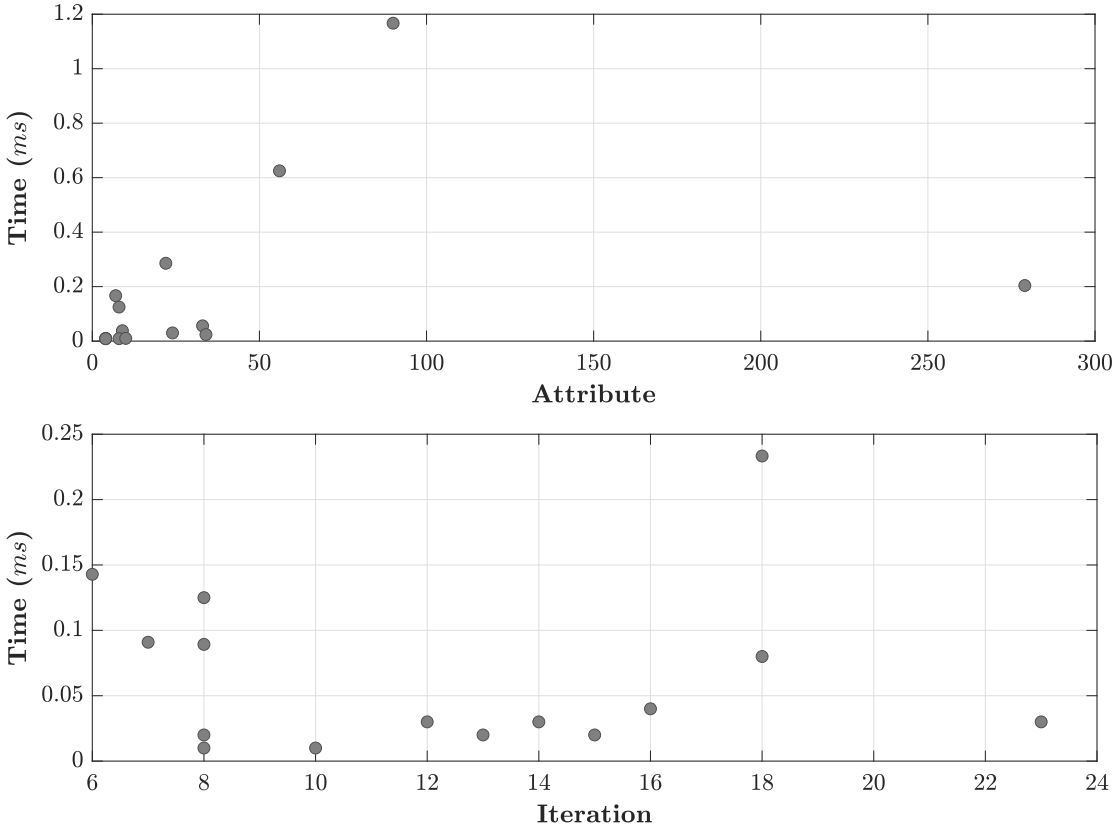


FIGURE 4.1: Time required (ms) by data owner to decrypt the shift matrix between two centroids for different numbers of attributes and different iterations

time). The runtimes required by the proposed Sk-Means data clustering were slightly higher than the standard algorithm but inspection of the table indicates that this was not significantly so.

4.4.3 Secure Clustering Accuracy

Clustering accuracy was measured by comparing the generated cluster configurations obtained using Sk-Means with those obtained using standard k-Means when using the same value for k. For it to be argued that the Sk-Means algorithm is operating correctly/accurately it should produce comparable cluster configurations with that produced using standard k-Means with the same value for k. The measure used to compare the two clustering configurations was the established Silhouette Coefficient ($Sil.\ Coef.$) [188]. The value for $Sil.\ Coef.$ was calculated as per Equation 4.5 where: (i) r_{i_j} is the jth record in cluster c_i, (ii) a_{i_j} is the average distance of the record from all other records within c_i and (iii) b_{i_j} is the minimum average distance of the record from all the records included in every other cluster. Thus $a(x_j)$ is a measure of the *cohesiveness* of the individual clusters, whilst $b(x_j)$ is a measure of the *separation* between clusters. The value of the $Sil.\ Coef.$ can thus vary between -1 to 1, where the closer the coefficient is to 1 the better the clustering configuration. In the context of the work presented in this thesis the intention was not to achieve a "best" clustering configuration but to compare the configurations for the purpose of establishing the accuracy of the proposed solution. The results are presented in Table 4.3.

TABLE 4.3: Execution time for secure k-Means clustering

UCI Dataset	Standard k-Means			Secure k-Means		
	Runtime (ms)	Num. Iter.	Sil. Coef.	Runtime (ms)	Num. Iter.	Sil. Coef.
Arrhythmia	38	10	0.602	215	10	0.602
Banknote Auth.	58	16	0.207	87	16	0.207
Blood Trans.	19	12	0.370	27	12	0.370
Breast Cancer	11	8	0.020	31	8	0.020
Breast Tissue	19	18	0.787	26	18	0.787
Chronic Kidney	14	8	0.009	29	8	0.009
Dermatology	14	15	0.801	35	15	0.801
Ecoli	6	23	0.628	39	23	0.628
Indian Liv. Pat.	12	13	0.169	18	13	0.169
Iris	9	14	0.836	14	14	0.836
Libras Mov.	26	18	0.590	125	18	0.590
Lung Cancer	3	8	0.645	17	8	0.645
Parkinsons	5	7	0.079	11	7	0.079
Pima Disease	19	8	0.000	30	8	0.000
Seeds	9	6	0.706	10	6	0.706

$$Sil.\ Coef. = \frac{\sum_{i=1}^{i=k} \frac{\sum_{j=1}^{j=|c_i|} Sil(r_{i_j})}{|c_i|}}{k}$$
$$Sil(r_{i_j}) = \frac{b(x_{i_j}) - a(x_{i_j})}{max(a(x_{i_j}), b(x_{i_j}))} \tag{4.5}$$

Inspection of Table 4.3, columns 4 and 7, shows that identical $Sil.\ Coef.$ values were obtained in all cases, indicating that the final clustering configuration produced using Sk-Means were the same as those produced using standard k-Means. Therefore it can be argued that the encryption scheme used, the proposed UDM concept and the proposed updating process using shift matrices do not adversely affect the quality of the produced cluster configurations. Table 4.3 also shows the number of iterations required using standard k-Means and the proposed Sk-Means algorithms (columns 3 and 6), the number of required iterations was the same with respect to each dataset. Thus, the proposed mechanism does not only produce the same clustering configuration, it is also arrived at in the same number of iteration.

4.4.4 Secure Clustering Security

The security of the proposed solution was evaluated by identifying the potential attacks that can be directed to breach the outsourced data privacy when using the UDM concept. The most likely attacks are those that can be launched by adversaries who have access to the encrypted dataset and/or the UDM, and by an adversary who is also a TPDM who is able to send the Shift Matrices (SMs) for decryption. As defined earlier in Chapter 2, the TPDM is considered to be a semi-honest party and thus a passive adversary who will follow the Sk-Means data clustering algorithm, but in the meanwhile may try to learn additional information by analysing the intermediate results and messages exchanged during algorithm execution. The adversaries are thus able to launch: (i) COAs against the encrypted datasets, (ii) utilises the elements of UDM (that are in plaintext) by

reveres engineering the elements to reveal statistical aspects of original data and (iii) develop dictionary attacks using the updating process.

With the reference to the first category of attack, COAs, the dataset in the proposed solution is encrypted using Liu's FHE scheme which features semantic security as presented earlier in Sub-section 3.3.1.3. In semantically secure encryption schemes an adversary who has access to cyphertexts cannot obtain any partial information about the corresponding plaintext values. Thus, examination of cyphertexts by external adversaries or by the TPDM will yield nothing about the corresponding plaintext values [17]. Therefore, COAs cannot be successful against Liu's FHE cyphertexts. Possible successful attacks are those directed at the UDM and making use of the updating process. The UDM elements are exchanged in plaintext and represent (a very large) set of linear equations, therefore attackers may be able to reverse engineer this set of equations so as to obtain the statistical distribution of the distance values contained in the UDM. The UDM updating process requires the sending of a HE encrypted shift matrix to the data owner for decryption, this may allow the TPDM to develop knowledge about the HE cyphertext and their corresponding plaintext values. This is not limited to Knowing Plaintext Attacks (KPAs), it can take the form of a Dictionary Attack (DA) whereby the adversary generates new cyphertexts and then derive their plaintext equivalent values using the HE properties and the knowledge obtained when the shift matrix cyphertext is decrypted.

4.5 Summary

This chapter has presented the concept of the Updatable Distance Matrix (UDM) and its usage in the context of Secure k-Means (Sk-Means) data clustering. The proposed algorithm operated over encrypted data without necessitating decryption, the UDM was used to determine the data similarity between data records and cluster centroids. The UDM could be updated using the proposed updating process utilising the idea of a shift matrix and required only very limited data owner participation. The proposed solution was presented in detail and its performance was evaluated using UCI datasets. The significance of the UDM was that it dramatically reduced the data owner participation from $k \times n \times i$ to $k \times a \times i$ where i is the iteration count. It should also be noted that use of a shift matrix, and the associated data owner participation, can be avoided if alternative forms of clustering were considered that avoid the need for cluster centroid recalculation. However, the UDM has significant security issue in that it is sent in plaintext form and thus its usage entails significant security concerns as it is vulnerable to reverse engineering attack, and susceptible to Known Plaintext Attacks (KPAs) and Dictionary Attack (DA). In the following chapter mechanisms whereby these security concerns may be addressed are considered using the idea of a Cryptographic Ensemble that utilises a HE and an OPE scheme to provide a comprehensive solution to the PPDM problem.

Chapter 5

Encrypted Updatable Distance Matrices

5.1 Introduction

The previous chapter presented the concept of Updatable Distance Matrices (UDMs). The fundamental idea was, instead of working with the actual dataset D, to use a proxy for the data, more specifically the distances between each attribute in D and every other attribute in D. The utility of UDMs was illustrated in the context of secure k-Means clustering; the Sk-Means algorithm was proposed. However, UDMs had celar application with respect to any clustering mechanism that involved comparison of records, such as Nearest Neighbour Clustering (NNC) [48]. The original dataset D was still required to determine the changing cluster centroids used in Sk-Means clustering. To maintain security it was proposed that the data be encrypted using a suitable HE scheme that supports centroid calculation; the proposed Sk-Means algorithm used Liu's FHE scheme. The advantage offered was that secure clustering could be conducted in a manner that dramatically reduced the required data owner participation compared to previously proposed solutions to realised secure k-Means clustering. Using Sk-Means most of the processing was delegated to the TPDM. The FHE mathematical properties of the adopted scheme (Liu's scheme) were used to manipulate D' without decryption; while the UDM concept was used, on each Sk-Means iteration, to determine the similarity between the encrypted data records and encrypted clusters centroids. However it was observed that the UDM was essentially a set of linear equations exchanged in plaintext form, and that this might be used to reverse engineer statistical aspects concerning the original dataset.

This chapter considers a solution to the observed disadvantages associated with the UDM approach described in the previous chapter. The main idea presented is to encrypt the UDM so that any statistical information concerning data frequency and distribution can no longer be derived. So that the UDM can still be used as described in the foregoing chapter the encryption scheme needs to support comparison operations. Liu's FHE scheme, used previously to encrypt D, does not support comparison, for this to be supported an OPE scheme is required. However, OPE schemes do not support the arithmetic functions required to calculate centroids (assuming k-Means clustering). A Cryptographic Ensemble, comprised of two encryption schemes, was therefore envisaged. The first to encrypt the dataset D to arrive at D' and the second to encrypt the UDM to arrive at an Encrypted UDM (EUDM). For the first it was decided to use Liu's FHE scheme [53], because this was a well-established scheme and because it was used with respect to the Sk-Means algorithm presented in the previous chapter. With respect

to the second no suitable "off-the-shelf" OPE scheme was available [51, 173, 178, 180, 181] hence a bespoke Order Preserving Encryption (OPE) scheme was designed, the Frequency and Distribution Hiding OPE (FDH-OPE) scheme, to facilitate the required UDM operation. The FDH-OPE scheme is one of the contributions of the thesis. The dataset D is encrypted using an HE scheme, Liu's scheme is suggested, and the UDM is encrypted using the proposed OPE scheme. Together the two schemes collectively provided the required privacy preservation.

To evaluate the proposed EUDM concept, coupled with the idea of a Cryptographic Ensemble, a secure variation of k-Means [46] clustering was implemented, the Double Blind Secure k-Means (DBSk-Means) algorithm. A disadvantage of the EUDM is that it requires a substantial amount of storage resource. An alternative variation of EUDMs, Encrypted Distance Matrices (EDMs), was proposed which required less storage but did not support the updating required to support secure k-Means clustering (a feature of which is repeated re-calculation of cluster centroids). The EDM concept was therefore illustrated using a Double Blind Secure Nearest Neighbour Clustering (DBSNNC) approach. Both algorithms are referred to as "double blind" algorithms because the TPDM worked with two encryption schemes, the Cryptographic Ensemble referred to earlier. In the case of k-Means use of EUDMs still required data owner participation, however this was limited to translating (flipping) a small set of cyphertexts from one encrypted form to another.

The rest of the chapter is organised in a similar manner to the previous chapter that described the UDM and Sk-Means data clustering approach. Section 5.2 presents the proposed FDH-OPE scheme. Section 5.3 presents the idea of Encrypted Updatable Distance Matrices (EUDMs) and Encrypted Distance Matrices (EDMs); which is followed, in Section 5.4, with descriptions of the DBSk-Means and DBSNNC algorithms. A worked example illustrating how the Cryptographic Ensemble is used in the context of the EUDM and DBSk-Means is given in Section 5.5. Section 5.6 presents an extensive evaluation of the Cryptographic Ensemble idea, EUDMs and EDMs. A summary of the work considered in this chapter, and a review of the main findings, is given in Section 5.7.

5.2 Frequency and Distribution Hiding Order Preserving Encryption

As noted in the above introduction the fundamental idea presented in this chapter, to address the disadvantages of the UDM approach presented in the foregoing chapter, was to use a Cryptographic Ensemble comprised of two encryption schemes. In the context of work presented in this thesis, the Cryptographic Ensemble used: (i) the established Liu's FHE scheme and (ii) the proposed FDH-OPE scheme. The latter is one of the main contributions of this thesis and is therefore considered in detail in this section.

The proposed Frequency and Distribution Hiding Order Preserving Encryption (FDH-OPE) scheme is an amalgamation of two existing OPE schemes, the nonlinear order preserving scheme proposed in [51] and the scheme of Zheli et al. [52]. The former was used to hide the data frequency in the generated cyphertexts and the latter to hide the data distribution. Note that revealing the data distribution in generated cyphertexts raises the potential of disclosing statistical features concerning the plaintext data. Encrypting data so that the data distribution is hidden requires knowledge of the distribution within the plaintext data, the plaintext intervals where the data density is high, and then generating the cyphertexts in such a way that high density plaintext intervals are

dispersed over large cyphertext intervals. The frequency of data is hidden by adopting a probabilistic approach to generating cyphertext so that different cyphertext values are generated for the same plaintext value when using the same encryption key.

The FDH-OPE scheme utilises two functions: (i) `KeyGen` to generate the encryption keys and (ii) `Encrypt` to encrypt plaintext values (using the generated encryption keys). There is no `Decrypt` function as the adopted encryption function is a one way function that randomly maps plaintexts to order preserving cyphertexts, there is no requirement for decryption.

The `KeyGen` component consists of several steps that cumulatively generate a secret key (SK) to be used by the `Encrypt` function. The first step is for the data owner to determine the "interval" of the message space $\mathcal{M} = [l, h)$ and the expanded "interval" of the cyphertext space $\mathcal{C} = [l', h')$ in such a way that $|\mathcal{M}| \ll |\mathcal{C}|$ and the l, l', and h, h', are the minimum and maximum interval boundaries for the message and cyphertext spaces respectively (see Figure 5.1). For the FDH-OPE scheme to operate correctly with respect to work presented in this thesis, the generated cyphertexts should preserve the "sign" so that positive plaintext values will always have positive cyphertexts and the negative plaintext values will always have negative cyphertexts. This requirement is essential to maintain correctness when updating UDMs (as will become clear in the next section, Section 5.5). The second step is the data distribution hiding process that itself comprises two steps, *message space splitting* and *non-linear cyphertext space expansion* which operate as follows:

Message space splitting: The data owner randomly splits the message space interval $\mathcal{M}$ into t consecutive intervals; $\mathcal{M} = \{m_1, \ldots, m_t\}$, where t is a random number. The length of each intervals is determined randomly by deciding the minimum and maximum interval boundaries in such a way that $\mathcal{M} = \cup_{i=1}^{i=t} m_i = \cup_{i=1}^{i=t} [l_i, h_i) = [l, h)$ and $[l_i, h_i) \cap [l_j, h_j) = \phi$ for all $i \neq j$ (Figure 5.1). The data density for each interval is then calculated as $Dens = \{dens_1, \ldots, dens_t\}$ where $dens_i$ is the density of data in message space m_i. The data density in this case refers to the number of plaintext values in the data to be encrypted that fall in a given message space interval.

Non-linear cyphertext space expansion: The data owner then splits the cyphertext space $\mathcal{C}$ into t intervals; $\mathcal{C} = \{c_1, \ldots, c_t\}$. So that the data distribution is hidden, the length of each cyphertext space interval c_i is determined according to the density of the data stored in $dens_i \in Dens$. Thus message space intervals with high data density will have large corresponding cyphertext space intervals. In other words if $dens_i > dens_j$ then $|c_i| > |c_j|$. The length of each cyphertext interval is determined by selecting an appropriate minimum and maximum interval boundaries in such a way that; $\mathcal{C} = \cup_{i=1}^{i=t} c_i = \cup_{i=1}^{i=t} [l'_i, h'_i) = [l', h')$ and $[l'_i, h'_i) \cap [l'_j, h'_j) = \phi$ for all $i \neq j$ (Figure 5.1).

On completion of the `KeyGen` process the calculated message space and cyphertext space interval boundaries will be the FDH-OPE secret keys.

The data frequency is hidden using a "one-to-many" encryption function that maps $v \in m_i$ to an OPE equivalent cyphertext $v' \in c_i$. Algorithm 6 gives the pseudo code for the encryption function (`Encrypt`). The inputs are the plaintext value to be encrypted v and the data sensitivity $sens$. The data sensitivity is defined in [51] as a minimum distance between the plaintext values in the dataset to be encrypted. More formally, given a set of all dataset attributes values V the sensitivity of V is the smallest difference value in the set; $|v_1 - v_2|$ $v_1 \in V$ and $v_2 \in V$ $v_1 \neq v_2$. The sensitivity is usually specific to a dataset. For example, if the attribute value in a dataset can only be even numbers, then

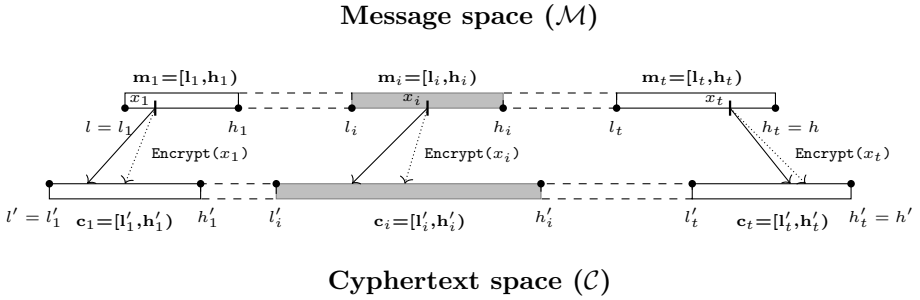


FIGURE 5.1: Message and cyphertext space splitting

the minimum sensitivity will be 2. Given a dataset of the salaries of staff in a company that takes the format $d_1d_2d_3.d_4d_5$, where d_i is a digit, the minimum sensitivity will be 0.01. That is, the smallest possible salary difference between two staff members in the company is 0.01. The `Encrypt` algorithm commences by determining the message space interval ID, i, within which $|v|$ is contained by calling the IntervalID procedure (line 2). The interval boundaries (the keys) of the ith message and cyphertext space are then retrieved in lines 3 and 4 by calling the Boundary and Boundary$'$ procedures. The values of the interval boundaries, and data sensitivity, are then used to calculate the interval scale value, $scale_i$, and the sample random value δ_i as per the equations given in lines 5 and 6. The $scale_i$ and δ_i values are then used in line 7 to calculate the cyphertext v'. The value of δ_i represents the amount of noise added to the cyphertext while maintaining the data ordering. To hide the distribution of the generated cyphertext, especially with respect to high density intervals so that cyphertexts are spread across the cyphertext space interval, the required value of δ_i may need to be high. In the proposed FDH-OPE scheme, the value of δ_i is determined individually for each interval so that longer intervals with a larger $scale_i$ value will consequently have a larger δ_i value than in the case of shorter intervals. The algorithm will exit (line 11) with cyphertext v'. As the encryption function requires the preservation of the plaintext's "sign", any plaintexts will be post-processed in terms of their absolute value. For negative plaintext values the resulting cyphertexts will multiplied by -1 (lines 8 to 10). Adding the random value δ_i to generated cyphertexts will guarantee that identical attribute values will not always have the same encryption.

Algorithm 6 FDH-OPE encryption algorithm

1: **procedure** ENCRYPT($v, sens$)
2: $i \leftarrow \text{IntervalID}\big(|v|\big)$
3: $l_i, h_i \leftarrow \text{Boundary}(i)$
4: $l'_i, h'_i \leftarrow \text{Boundary}'(i)$
5: $scale_i = \frac{l'_i - h'_i}{l_i - h_i}$
6: $\delta_i = \text{Random}(0, sens \times scale_i)$
7: $v' = l'_i + scale_i \times \big(|v| - l_i\big) + \delta_i$
8: **if** $v < 0$ **then**
9: $v' = v' \times -1$
10: **end if**
11: **Exit** with v'
12: **end procedure**

5.3 Encrypted UDMs and Encrypted Distance Matrices

This section presents the twin concepts Encrypted Updatable Distance Matrices (EUDMs) and Encrypted Distance Matrices (EDMs). The first is the encrypted equivalent of the UDM presented in Chapter 4. The second addresses the storage disadvantage associated with EUDM but with the disadvantage that the updateability feature is lost. Both are discussed in detail in this section. The section is divided into two sub-sections. Sub-section 5.3.1 describes the EUDM concept, whilst the idea of EDMs is presented in Sub-section 5.3.2.

Note that in the remainder of this chapter, to differentiate between Cryptographic Ensemble cyphertexts, the value x encrypted using Liu's FHE scheme is denoted as $[x']$ while the value x encrypted using the proposed FDH-OPE scheme is denoted by $[[x']]$.

5.3.1 Encrypted Updateable Distance Matrices

An EUDM, as in the case of the UDM, is a 3D matrix where the first and the second dimensions correspond to the number of records n in a given dataset D, and the third dimension corresponds to the number of attributes a featured in D. However, an EUDM in contrast to the UDM, holds the (encrypted) distances between attributes values in each record with the corresponding attributes values in every other record; as opposed to the unencrypted distances. The encryption scheme used to this end is the FDH-OPE scheme presented in Section 5.2. The form of a specific EUDM, $[[EU]]$, is given in Equation 5.1 where the first two elements in the suffix of a matrix element are the relevant record numbers and the third the relevant attribute number. An EUDM is generated in two steps: (i) UDM calculation and (ii) UDM encryption to arrive at the Encrypted UDM (EUDM). The first was detailed in Section 4.2 of Chapter 4. Given $[[EU]]$, each element $[[eu_{x,y,z}]] \in [[EU]]$ is calculated as per Equation 5.2 and then encrypted using the FDH-OPE scheme; the relevant pseudo code is given in Algorithm 6.

Using an EUDM the TPDM will thus have access to the "*order*" of the distances but not the original distance values. From Equation 5.1 and Equation 5.2 it can be observed, as in the case of UDMs, that an EUDM is symmetric about the leading diagonal and hence only the lower (or upper) "triangle" of the matrix needs to be calculated. The dataset D is encrypted using Liu's FHE scheme and hence the TPDM will have access only to encrypted data records. Two encryption schemes are therefore used, the Cryptographic Ensemble referred to earlier.

$$[[EU]] = \begin{bmatrix} \left([[eu_{1,1,1}]], \cdots, [[eu_{1,1,a}]]\right) & \cdots & \left([[eu_{1,n,1}]], \cdots, [[eu_{1,n,a}]]\right) \\ \vdots & \ddots & \vdots \\ \left([[eu_{n,1,1}]], \cdots, [[eu_{n,1,a}]]\right) & \cdots & \left([[eu_{n,n,1}]], \cdots, [[eu_{n,n,a}]]\right) \end{bmatrix} \tag{5.1}$$

$$eu_{x,y,z} = r_{x,z} - r_{y,z} \tag{5.2}$$

A feature of UDMs, described in the previous chapter, was that they can be updated; this is also a requirement for EUDMs. The EUDM updating process, assuming a secure k-Means algorithm, is given in Algorithm 7. The inputs are: the EUDM $[[EU]]$ to be updated, the ith iteration cluster centroids $[Cent'_i]$ and the $i+1$th iteration centroids $[Cent'_{i+1}]$. The updating process starts by generating an encrypted shift matrix $[S']$ (line 2). The matrix $[S']$ holds the difference between the k-Means iteration i and $i+1$ centroids and hence it measures $k \times a$ as shown in Equation 5.3. $[S']$ is calculated

using the mathematical properties of Liu's FHE scheme, thus to update $[[EU]]$ an order preserving cypher is required. Therefore, the content of $[S']$ needs to be "flipped", from the HE format to the FDH-OPE format, to give $[[S']]$. This is done in lines 3 and 4 by the data owner. The TPDM can then use the encrypted shift matrix $[[S']]$ to update $[[EU]]$ (line 5). The algorithm exits (line 11) with the updated EUDM and a Boolean variable *terminate* that is set to "true" when the shift matrix S is comprised entirely of zeros; in other words when the k-Means achieves stable centroids in two continuous iterations, and "false" otherwise (as per lines 6 to 10).

$$[S'] = \begin{bmatrix} [s'_{1,1}] & [s'_{1,2}] & \cdots & [s'_{1,a}] \\ \vdots & \vdots & \ddots & \vdots \\ [s'_{k,1}] & [s'_{k,2}] & \cdots & [s'_{k,a}] \end{bmatrix} \tag{5.3}$$

Algorithm 7 EUDM updating process

1: **procedure** UPDATEENCRYPTEDUDM($[[EU]]$,$[Cent'_i]$,$[Cent'_{i+1}]$)
2: $[S'] = [Cent'_i] \ominus [Cent'_{i+1}]$
3: S = the data owner will decrypt $[S']$
4: $[[S']]$ = the data owner will re-encrypt the elements using the FDH-OPE key
5: $[[EU]]$=$[[EU]]$+$[[S']]$
6: **if** $S == 0$ **then**
7: $terminate$ = true
8: **else**
9: $terminate$ = false
10: **end if**
11: **Exit** with $[[EU]]$, $terminate$
12: **end procedure**

5.3.2 Encrypted Distance Matrices

An EUDM requires considerable storage, this storage can be reduced by turning the 3D EUDM into a 2D Encrypted Distance Matrix (EDM). The distinction is that an EUDM measures $n \times n \times a$ whilst an EDM measures $n \times n$, a reduction in size by a factor of a. An alternative way of expressing the distinction is that an EUDM holds distances at the attribute level whilst an EDM holds distances at the record level. The proposed EDM, although serving to reduce the storage requirement, therefore entails the trade-off that its application in the context of data clustering will be limited to secure clustering techniques that do not entail new cyphertext generation as in the case of secure k-Means clustering which entails centroid re-calculation. This means, at least in the context in which distance matrices have been used as reported in this thesis so far, that EDMs do not require updating. The effort required on behalf of a data owner to update an EDM, should this be a requirement, is such that from a practical perspective EDMs are not updatable.

The nature of a specific EDM, $[[ED]]$, is formally defined in Equation 5.4. The first and the second dimensions corresponding to the number of data records, n, in the encrypted dataset $[D']$. The EDM is generated in two steps: (i) DM calculation and (ii) DM encryption to arrive at an Encrypted DM (EDM). The value (distance) for each EDM element $[[ed_{x,y}]]$ is calculated as per Equation 5.5, this value is then encrypted, in step 2, using the proposed FDH-OPE. As in the case of UDMs and EUDMs, an EDM

is symmetric about the leading diagonal, so only the values for the lower (or the upper) 2D triangular of the matrix needs to be calculated.

$$[[ED]] = \begin{bmatrix} [[ed_{1,1}]] & [[ed_{1,2}]] & \cdots & [[ed_{1,n}]] \\ \vdots & \vdots & \ddots & \vdots \\ [[ed_{n,1}]] & [[ed_{n,2}]] & \cdots & [[ed_{n,n}]] \end{bmatrix} \tag{5.4}$$

$$ed_{x,y} = \sum_{i=1}^{i=a} (|r_{x,i} - r_{y,i}|) \tag{5.5}$$

5.4 Secure Data Clustering Using EUDM and EDM

This section presents the proposed secure data clustering processes designed to utilise the EUDM and EDM concepts presented above, as well as the idea of a Cryptographic Ensemble where more than one encryption technique are incorporated into a single approach so that PPDM can be realised. The proposed process is generic in nature, and not dedicated to any single specific data clustering algorithm; it is directed at providing a general solution for secure data clustering where the adopted technique requires distance comparisons. It is suggested that this encompasses all current data clustering algorithms in that, by definition, clustering entails the grouping of examples that are similar, similarity is always defined according to some kind of distance measurement. However, to illustrate the utility of the proposed process two specific algorithms that encapsulate the process are used; the Double Blind Secure k-Means (DBSk-Means) algorithm which is founded on the use of EUDMs and the Double Blind Secure Nearest Neighbour Clustering (DBSNNC) which is founded on the use of EDMs. The phrase "double blind" is used because the Cryptographic Ensemble features two encryption schemes.

The remainder of this section is organised as follows. Sub-section 5.4.1 presents the data preparation process required when adopting the proposed Cryptographic Ensemble and EUDM/EDM approach. The next two sub-sections, Sub-section 5.4.2 and Sub-section 5.4.3, present respectively the Double Blind Secure k-Means (DBSk-Means) algorithm founded on the EUDM concept and the Double Blind Secure NNC (DBSNNC) algorithm founded on the EDM concept. The following section, Section 5.5, presents worked examples of the DBSk-Means algorithm using EUDM.

5.4.1 Data Preparation for Outsourcing

This section presents the process conducted by a data owner to prepare a dataset for outsourcing to a TPDM, who provides DMaaS, using the proposed EUDM/EDM approach. The pseudo code for the data preparation process is presented in Algorithm 8. The inputs are the raw dataset $RawD$, the number of attributes a featured in $RawD$ and the number of Liu's scheme sub-cyphertexts m (as presented in Sub-section 3.3.1). Similar to the UDM approach, the first step is to process the raw data by casting the categorical attributes into a discrete integer value form to prepare the data for encryption (line 2). The secret key list, $SK(m)$, required using Liu's FHE scheme, is then generated in line 3. The processed dataset is then encrypted, using Liu's FHE scheme, to give $[D']$ using Algorithm 1 (lines 4 to 7). Dependent on the adopted approach, the algorithm then starts the generation of the desired matrix; either EUDM or EDM. The desired matrix is generated in two steps: (i) matrix calculation (line 8) and (ii) matrix encryption (line 15). The matrices (EU and ED) are calculated by calling the

sub-procedure *CalculateUDM* or sub-procedure *CalculateDM* as appropriate, given at the end of the algorithm (lines 18 to 28 and lines 29 to 37, respectively). The matrix calculation algorithms commence by dimensioning the desired matrix (line 19 and line 30 in the context of EUDM and EDM respectively). Recall that, the EUDM is a 3D matrix where the first and the second dimensions corresponding to the number of data records feature in $RawD$, and the third dimension is the number of attributes a. The EDM is 2D matrix where the two dimensions are correlated to the number of data records feature in $RawD$. Recall also that the value $r_{x,z}$ in lines 23 and 33 is the value of the zth attribute in the feature vector describing the xth record in the dataset, while $r_{y,z}$ is the value of the zth attribute in the feature vector describing the yth record. The matrix encryption commences by generating the FDH-OPE key (lines 9 to 14) as discussed in Section 5.2. The first step is for the data owner to decide the message space and cyphertext space interval boundaries (lines 9 and 10). The data owner will then randomly split the message space intervals (line 11, where t is a random number). The data density is then calculated in line 12, for each message space interval, by calling the *DensityCalculation* procedure. According to the data density in the desired calculated matrix, EU or ED, the cyphertext space is split in line 13 to result in a set of cyphertext intervals. The data sensitivity is then calculated in line 14 (as described in Section 5.2). The key generated is then used to encrypt the desired matrix elements in line 15. The output from the data preparation process is the encrypted dataset $[D']$ and an EUDM $[[EU]]$ or EDM $[[ED]]$, ready to be sent to the TPDM.

5.4.2 Double Blind Secure k-Means Clustering

This sub-section presents the Double Blind Secure k-Means (DBSk-Means) clustering algorithm used to illustrate the utility of the EUDM concept. The pseudo code for the DBSk-Means algorithm is given in Algorithm 9. The inputs are an encrypted dataset $[D']$, an EUDM $[[EU]]$ (because updating the matrix is required when new centroids have been calculated) and the desired number of clusters k. The operation of DBSk-Means is very similar to Sk-Means as presented earlier in Chapter 4. The differences are that: (i) an EUDM is used instead of a UDM and (ii) the EUDM and SM are encrypted. On each k-Means iteration the SM is returned by the data owner to the TPDM in encrypted form (using the proposed FDH-OPE scheme). Therefore, the TPDM has access to the "order of distance" instead of original distance values as in the case of the UDM approach. The data similarity between cluster centroids $[cent'_y]$ and encrypted data record $[r'_x]$ is determined using the EUDM as per Equation 5.6.

$$sim([[EU]], [r'_x], [cent'_y]) = \sum_{z=1}^{z=a} |eu_{x,y,z}| \tag{5.6}$$

5.4.3 Double Blind Secure Nearest Neighbour Clustering

This sub-section presents the Double Blind Secure Nearest Neighbour Clustering (DBSNNC) algorithm used to illustrate the utility of the EDM concept. Recall that an EDM is not updatable (at least from a practical perspective), but offers the advantage of a reduced storage requirement compared with the EUDM. The DBSNNC process is conducted by a TPDM following a manner similar to that used for standard NNC [48]. The algorithm does not feature any need to generate new cyphertexts and hence an EDM is well suited. The pseudo code for the DBSNNC algorithm is presented in Algorithm 10. The inputs are the encrypted dataset $[D']$, an EDM $[[ED]]$ (previously generated by

Algorithm 8 Data outsourcing process required when using the EUDM or EDM concept

procedure OUTSOURCEDATA($RawD$,a,m)
  D = Dataset $RawD$ converted to numeric form where necessary
  $SK(m)$ = Liu's scheme secret key generated as in Sub-section 3.3.1
  $[D'] = \emptyset$
  for all $r \in D$ **do**
    $[D'] = [D'] \cup$ Encrypt$(r, SK(m))$ ▷ Algorithm 1
  end for
  Matrix Calculation:
    EUDM: EU = CalculateUDM(D,a)
    EDM: ED = CalculateDM(D,a)
  $l, h \longleftarrow$ Decide the message space interval boundaries
  $l', h' \longleftarrow$ Decide the cyphertexts space interval boundaries
  $MessageIntervals[]$ = MessageSpaceSplitting(t)
  Density Calculation:
    EUDM: $Dens[]$ = DensityCalculation($MessageIntervals[]$,EU)
    EDM: $Dens[]$ = DensityCalculation($MessageIntervals[]$,ED)
  $CypherIntervals[]$ = CypherSpaceExpansion($Dens$)
  $sens$= Calculate data sensitivity using selected matrix EU or ED
  Matrix Encryption: ▷ Algorithm 6
    $[[EU]] = [[EU]] \cup$ Encrypt(eu, $sens$)) ($eu \in EU$)
    $[[ED]] = [[ED]] \cup$ Encrypt(ed, $sens$)) ($ed \in ED$)
  Exit with $[D']$ and $[[EU]]$ or $[[ED]]$
end procedure
procedure CALCULATEUDM(D,a)
  EU = Empty EUDM dimensioned according to $|D|$, $|D|$ and a
  for $x = 1$ to $x = |D|$ **do**
    for $y = 1$ to $y = x$ **do**
      for $z = 1$ to $z = a$ **do**
        $eu_{x,y,z} = r_{x,z} - r_{y,z}$ ($eu_{x,y,z} \in EU$) ▷ Equation 5.2
      end for
    end for
  end for
  Exit with EU
end procedure
procedure CALCULATEDM(D,a)
  ED = Empty EDM dimensioned according to $|D|$ and $|D|$
  for $x = 1$ to $x = |D|$ **do**
    for $y = 1$ to $y = x$ **do**
      $ed_{x,y} = \sum_{z=1}^{z=a} |r_{x,z} - r_{y,z}|$ ($ed_{x,y} \in ED$) ▷ Equation 5.5
    end for
  end for
  Exit with ED
end procedure

Algorithm 9 Double Blind Sk-Means (DBSk-Means) clustering algorithm

1: **procedure** DBSk-Means($[D']$, $[[EU]]$, k)
2: $[C]$ = Set of k empty clusters
3: Select the first k records in $[D']$ and assign to $[C]$ (one per cluster)
4: $i = 1$
5: $[Cent'_i]$ = Set of first k records in $[D']$
6: $[C]$ = populateClusters($k+1$, $[[EU]]$, $[C]$, $[D']$, $[Cent'_i]$)
7: $[Cent'_{i+1}]$ = CalculateCentroids($[C]$,k) ▷ Sub-procedure in Algorithm 5
8: $[[EU]]$, terminate = UpdateEncryptedUDM($[[EU]]$,$[Cent'_i]$, $[Cent'_{i+1}]$) ▷ Algorithm 7
9: **while** ! *terminate* **do**
10: $[C]$ = Set of k empty clusters
11: $[C]$ = populateClusters(1, $[[EU]]$,$[C]$,$[D']$,$[Cent'_{i+1}]$)
12: $i = i + 1$
13: $[Cent'_{i+1}]$ = CalculateCentroids($[C]$,k)
14: $[[EU]]$, *terminate* = UpdateEncryptedUDM($[[EU]]$, $[Cent'_i]$, $[Cent'_{i+1}]$)
15: **end while**
16: **Exit** with $[C]$
17: **end procedure**
18: **procedure** populateClusters(rid,$[[EU]]$,$[C]$,$[D']$,$[Cent']$)
19: $id = null$
20: **for** $x = rid$ to $x = |[D']|$ **do**
21: **for** $y = 1$ to $y = |[C]|$ **do**
22: $[[similarity]] = sim([[EU]], [r'_x], [cent'_y])$ where $[r'_x] \in [D']$ and $[cent'_y] \in [Cent']$ (Equation 5.6)
23: id = cluster identifier with lowest *similarity* value so far
24: **end for**
25: $[c_{id}] = [c_{id}] \cup [r'_x]$ ($[c_{id}] \in [C]$ and $[r_x]' \in [D']$)
26: **end for**
27: **Exit** with $[C]$
28: **end procedure**

data owner) and the desired NNC threshold $[[\sigma']]$. The dataset is encrypted using the Liu's FHE scheme whilst the EDM and σ are encrypted using the FDH-OPE scheme. The algorithm commences by creating an empty set of clusters $[C]$ that will hold records encrypted using Liu's scheme (line 2). The first record in $[D']$ is then used to create cluster $[C_1]$ which is then added to the set of clusters $[C]$ (lines 3 and 4). The number of generated clusters so far is set to 1 (line 5). A loop is then entered (lines 6 to 16) that iteratively clusters the remaining records in $[D']$. The loop commences by finding a record $[r'_m]$ whose distance from $[r'_i]$ is the smallest. To this end, the EDM is used to determine the similarity as per Equation 5.7 where $[r'_i]$ and $[r'_m]$ are two encrypted records. Note that the value calculated in Equation 5.7 is the "order of the distance" (or encrypted distance) not a plaintext distance value. The calculated value is then compared with the encrypted threshold value $[[\sigma']]$ (line 8) and in the case when the value is less than or equal to $[[\sigma']]$ the record $[r'_i]$ is added to cluster $[C_x]$ where the record $[r'_m]$ is contained (lines 9 and 10), otherwise a new cluster is created (lines 12 to 14). The algorithm will continue until all records in $[D']$ are assigned to clusters and exits with a cluster configuration $[C]$.

Algorithm 10 Double Blind Secure Nearest Neighbour Clustering (DBSNNC)

```
1: procedure DBSNNC([D'], [[ED]], [[σ']])
2:     [C] = Empty set of clusters
3:     [C_1] = {[r'_1]}
4:     [C] = [C] ∪ [C_1]
5:     k = 1
6:     for i = 2 to i = |[D']| do
7:         Find [r'_m] in some cluster C_x where the sim([[ED]], [r'_i], [r'_m]) is the smallest
8:         if sim([[ED]], [r'_i], [r'_m]) ⩽ [[σ']] and [r'_m] is founded then
9:             [C_x] = [C_x] ∪ [r'_i]
10:            Update the cluster set [C_x] in the set of clusters [C]
11:        else
12:            k = k + 1
13:            [C_k] = {[r'_i]}
14:            [C] = [C] ∪ [C_k]
15:        end if
16:    end for
17:    Exit with [C]
18: end procedure
```

$$sim([[ED]], [r'_i], [r'_m]) = ed_{i,m} \tag{5.7}$$

5.5 Worked Example Using An EUDM and The DBSk-Means Algorithm

To assist in the understanding of the EUDM concept presented in this chapter, in the context of the data preparation for outsourcing process described in Sub-section 5.4.1 and DBSk-Means introduced in Sub-section 5.4.2, a worked example is presented in this section using the sample dataset D given in Table 5.1. The dataset D comprises five records and two attributes; $D = \{r_1, r_2, r_3, r_4, r_5\}$ and $r_i = \{r_{i,1}, r_{i,2}\}$. The outsourcing process, applied to this data, is illustrated in Sub-section 5.5.1, whilst the DBSk-Means process is illustrated in Sub-section 5.5.2. The operation of EDMs is not illustrated here but it operates in a similar, although not appropriate for k-Means clustering.

Table 5.1: Worked example dataset D

0.73	8.84
49.93	34.44
0.57	65.04
62.15	32.29
59.47	36.04

5.5.1 Data Preparation for Outsourcing Worked Example

As noted above, the data outsourcing process commences by translating the raw data into a suitable format and then generating a UDM which is then encrypted to give an EUDM. Recall that a UDM is a 3D matrix that holds distances between the attribute

values in the records. For the given example the UDM will measure $5 \times 5 \times 2$. Each element in the UDM is calculated using Equation 5.2, given in Section 5.3. Recall that a UDM is symmetric about the leading diagonal; hence only the elements (distance values) for the lower (or upper) triangle need to be calculated. In this worked example, the lower triangle is calculated. The resulting unencrypted UDM, U, is then as shown below (the third dimension is indicated using value pairs):

$$U = \begin{bmatrix} (0.00, 0.00) & & & & \\ (49.20, 25.60) & (0.00, 0.00) & & & \\ (-0.16, 56.20) & (-49.36, 30.60) & (0.00, 0.00) & & \\ (61.42, 23.45) & (12.22, -2.15) & (61.58, -32.75) & (0.00, 0.00) & \\ (58.74, 27.20) & (9.54, 1.60) & (58.90, -29.0) & (-2.68, 3.75) & (0.00, 0.00) \end{bmatrix}$$

Thus, the value for element $u_{2,1,1}$ is $r_{2,1} - r_{1,1} = 49.93 - 0.73 = 49.20$, that for element $u_{2,1,2}$ is $r_{2,2} - r_{1,2} = 34.44 - 8.84 = 25.60$, and so on. The elements in the leading diagonal are all zeros as they hold the distance between the same data records in D. For example, the value for element $u_{3,3,1}$ is $r_{3,1} - r_{3,1} = 0.57 - 0.57 = 0.0$.

Once calculated a UDM is encrypted, using the proposed FDH-OPE scheme, to give an EUDM. The first step to generate the FDH-OPE secret key is to determine the intervals of the boundaries (message space interval $\mathcal{M}$ and cyphertexts space interval $\mathcal{C}$) and then randomly split the message space and non-linearly split the cyphertext space to determine the interval boundaries of $\mathcal{M}$ and $\mathcal{C}$ respectively. For simplicity, small message space and cyphertexts space will be used. The message space is set to $\mathcal{M} = [-70, 70)$ and cyphertext space is set to $\mathcal{C} = [-1026, 1026)$. Thus, the message space maximal interval is 70 and the minimal interval is -70, whereas the cyphertext space maximal interval is 1026 and the minimal interval is -1026. This means that the maximum value that can be encrypted using this FDH-OPE scheme is 70 and the minimum value that can be encrypted is -70. The maximum cyphertexts generated by the FDH-OPE scheme encryption process is 1026 and the minimum is -1026. The message spaces will be split randomly, for example into *four* intervals ($t = 4$).

As noted earlier in Section 5.2, for the EUDM to operate correctly with respect to k-Means clustering it is required that the "sign" of each distance value is preserved. This is a crucial feature for the correctness of the updating process when the shift matrix is added to the EUDM to be used for the next iteration. Therefore, in the encryption function, Algorithm 6, any plaintexts will be processed in terms of their absolute value and then the generated cyphertext multiplied by -1 in the case of negative values. This means the message space splitting is only required to consider the positive intervals $\mathcal{M} = [0, 70)$. However, for the density calculation process, required for expanding the cyphertext space, negative values to be encrypted contribute to the density count of the interval within which their absolute equivalent value contains. For the example the size for the *four* message space will be defined as follows $m_1 = [0, 16)$, $m_2 = [16, 26)$, $m_3 = [26, 39)$ and $m_4 = [39, 70)$ as shown in Figure 5.2 (the value above each interval is the attribute value density for that interval). For the purpose of not revealing the cyphertexts of zeros, the leading diagonal values, which are all zeros, are not encrypted and are thus not involved in the density calculation. For example, the density of data within the interval $[0, 16)$ is calculated by counting the attribute values in U that belongs to this interval (-0.16, 12.22, -2.15, 9.54, 1.60, -2.68 and 3.75); hence the $dens_1 = 7$. The corresponding cyphertext space will then be split into *four* intervals such that the length of each interval is determined according to the density of the data in the corresponding message space interval. Consequently, dense message space intervals will correspond to large cyphertext space intervals. To this end, Equation 5.8 was used to

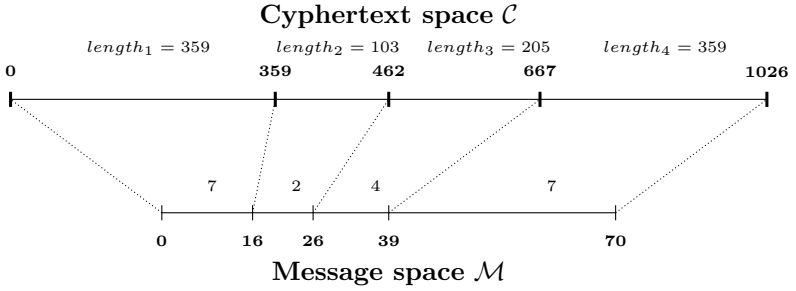


FIGURE 5.2: Message and cyphertext space splitting example

determine the length of the cyphertext space intervals, where: (i) CIS is the Cypher Interval Size, the maximum interval boundary of the cyphertext space (1026), and (ii) $ratio_i$ is the interval density in terms of the number of values in the interval divided by the total number of values in U, 20 in this case (we only need to consider the 20 elements in the lower triangle).

$$\begin{aligned} interval\ length_i = &\quad round(CIS \times ratio_i) &&= round(1026 \times ratio_i) \\ ratio_i = &\quad \tfrac{interval\ density}{|U|} &&= \tfrac{dens_i}{20} \end{aligned} \tag{5.8}$$

In the first message space interval, m_1 the density, $dens_1$, is 7; thus $ratio_1 = \frac{7}{20} = 0.35$ and consequently the first cyphertext $interval\ length_1 = 1026 \times 0.35 = 359.1 \approx 359$. The maximal interval will then be $l'_1 + 359 = 0 + 359 = 359$ (note that l'_1 is the minimal interval boundary of the first cyphertext space interval that equals l' for the first interval). For the next intervals the $l'_i = h'_{i-1}$. The calculation is repeated for each interval so as to give a final result. The cyphertext intervals $c_1 = [0, 359), c_2 = [359, 462), c_3 = [462, 667), c_4 = [667, 1026)$ will produce the result shown in Figure 5.2. From the figure it can clearly be seen that intervals with high density have longer cyphertext intervals than intervals with low density. The UDM elements are then encrypted using Algorithm 6. For example, $u_{2,1,2} = 25.60$ will be encrypted as follows:

$$\begin{aligned} v = &\quad 25.60 \Longrightarrow |25.60| = 25.60 &&\in m_2 \\ scale_2 = &\quad \tfrac{l'_2 - h'_2}{l_2 - h_2} = \tfrac{359-462}{16-26} &&= 10.3 \\ [[v']] = &\quad l'_2 + scale_2 \times \Big(|v| - l_2\Big) &&= 359 + 10.3\Big(|25.60| - 16\Big) = 457.88 \\ [[v']] = &\quad 457.88 + \delta_2 &&0 \leqslant \delta_2 \leqslant (10.3 \times 0.01)\ 0.01\ is\ data\ sensitivity \\ [[v']] = &\quad 457.88 + 0.103 = 457.983 \quad if \quad \delta_2 = 0.103 \end{aligned}$$

Recall that in the above the value of δ_2 is equivalent to $scale_2 \times sens$, where $sens$ is the sensitivity value calculated as described in Section 5.2 and $scale_i$ is calculated for each interval as shown in Algorithm 6. As a consequence of the above, the EUDM $[[EU]]$ will be:

$$[[EU]] = \begin{bmatrix} ([[eu_{1,1,1}]],[[eu_{1,1,2}]]) & \cdots & ([[eu_{1,5,1}]],[[eu_{1,5,2}]]) \\ \vdots & \ddots & \vdots \\ ([[eu_{5,1,1}]],[[eu_{5,1,2}]]) & \cdots & ([[eu_{5,5,1}]],[[eu_{5,5,2}]]) \end{bmatrix} =$$

$$\begin{bmatrix} (0.0,0.0) & & & & \\ (785.23,457.98) & (0.0,0.0) & & & \\ (-3.81,866.29) & (-787.08,534.70) & (0.0,0.0) & & \\ (926.74,435.84) & (274.44,-48.47) & (928.59,-568.61) & (0.0,0.0) & \\ (895.71,481.08) & (214.30,36.13) & (897.56,-509.47) & (-60.36,84.37) & (0.0,0.0) \end{bmatrix}$$

The dataset D also needs to be encrypted using Liu's FHE scheme following the pseudo code given in Algorithm 1. Assume that the private secret key list, generated by the data owner, is $SK(m) = SK(3) = \Big[(k_1,s_1,t_1),(k_2,s_2,t_2),(k_3,s_3,t_3)\Big] = \Big[(3.2,2.7,0),(9.1,3.1,1.5),(3.6,7.9,0)\Big]$ and that the following three random numbers are generated for the encryption $r_1 = 5689.23, r_2 = -8523.87$ and $r_3 = 24231.47$. In practice different random numbers will be generated each time the encryption function is invoked and the values of the secret key SK will be very large; however, for illustrative purposes the same random variables will be used to encrypt the entire dataset D and a small value for the secret key selected. Recall that, when using Liu's FHE scheme the plaintext value v will be encrypted to a cyphertext E that comprises m sub-cyphertexts. In the considered example 3 cyphertexts were used $E = \{e_1, e_2, e_3\}$. Consider attribute value $r_{1,2} = 8.84$ as an example, $v = 8.84$ will be encrypted as follows:

$$\begin{aligned} e_1 &= k_1 \times t_1 \times v + s_1 \times r_3 + k_1 \times (r_1 - r_2) \\ &= 3.2 \times 0 \times 8.84 + 2.7 \times 24231.47 + 3.2 \times (5689.23 + 8523.87) \\ &= 110906.889 \\ e_2 &= k_2 \times t_2 \times v + s_2 \times r_3 + k_2 \times (r_2 - r_1) \\ &= 9.1 \times 1.5 \times 8.84 + 3.1 \times 24231.47 + 9.1 \times (-8523.87 - 5689.23) \\ &= -54100.9870 \\ e_3 &= (k_3 + s_3 + t_3) \times r_3 \\ &= (3.6 + 7.9 + 0) \times 24231.47 \\ &= 278661.905 \end{aligned}$$

The encrypted value will then be $(110906.889, -54100.9870, 278661.905)$. The encrypted dataset will be as shown in Table 5.2. Note that e_1 and e_3 are always the same because we use the same random numbers. However, this will not be the case in practice. The data owner is now in a position to send the encrypted dataset $[D']$, and the associated EUDM $[[EU]]$, to the TPDM.

Table 5.2: Worked example encrypted dataset $[D']$

$(110906.889, -54211.6885, 278661.905)$	$(110906.889, -54100.9870, 278661.905)$
$(110906.889, -53540.1085, 278661.905)$	$(110906.889, -53751.5470, 278661.905)$
$(110906.889, -54213.8725, 278661.905)$	$(110906.889, -53333.8570, 278661.905)$
$(110906.889, -53373.3055, 278661.905)$	$(110906.889, -53780.8945, 278661.905)$
$(110906.889, -53409.8875, 278661.905)$	$(110906.889, -53729.7070, 278661.905)$

5.5.2 Double Blind Sk-Means Worked Example

Data clustering, using DBSk-Means, is conducted by the TPDM on receiving a request from the data owner specifying the number of clusters k. Following Algorithm 9, and assuming $k = 2$, the first step is to select the first two encrypted records in $[D']$ as cluster centroids where $cent'_{1_1}$ is the first centroid for the first DBSk-Means iteration and $cent'_{1_2}$ is the second centroid for the first DBSk-Means iteration, thus:

$$\begin{aligned}[cent'_{1_1}] &= [(110906.889, -54211.6885, 278661.905),\\ &\quad (110906.889, -54100.9870, 278661.905)]\\ [cent'_{1_2}] &= [(110906.889, -53540.1085, 278661.905),\\ &\quad (110906.889, -53751.5470, 278661.905)]\end{aligned}$$

The remaining encrypted records in $[D']$ are added to the clusters according to their similarity with the centroids using $[[EU]]$. Only the first two columns in $[[EU]]$ are used; the number of cluster centroids. Therefore, the following sub-matrix from $[[EU]]$ is used to determine data similarity where the rows are records and the columns the cluster centroids:

$$\begin{bmatrix} (0.0, 0.0) & (-785.23, -457.98)\\ (785.23, 457.98) & (0.0, 0.0)\\ (-3.81, 866.29) & (-787.08, 534.70)\\ (926.74, 435.84) & (274.44, -48.47)\\ (895.71, 481.08) & (214.30, 36.13) \end{bmatrix}$$

For illustrative purpose, all sub-matrix elements are calculated. In practice, the values for elements in the upper triangle can be obtained by simply reversing the sign of corresponding elements in the lower triangle. The element $eu_{1,2,1}$ is then assigned using the value in element $eu_{2,1,1}$ but with a reversed sign. Using the above sub-matrix the similarity between each xth record and each ith centroid is determined using equation 5.9. For example, the similarity between the *third* record and $first$ centroid is $|[[-3.81]]| + |[[866.29]]| = [[870.1]]$ whilst the similarity with the *second* centroid is $|[[-787.08]]| + |[[534.70]]| = [[1321.78]]$. Therefore, the third record will be assigned to the first cluster. This can be verified by conducting the same calculation using the plaintext equivalent values where the third record is r_3 =(0.57,65.04) and the plaintexts centroids are $cent_{1_1}$ =(0.73,8.84), $cent_{1_2}$ =(49.93,34.44). Hence $r_3 - cent_{1_1}$=$(|0.57 - 0.73|, |65.04 - 8.84|) = (0.16 + 56.2) = 56.36$ and $r_3 - cent_{1_2}$= $(|0.57 - 49.93|, |65.04 - 34.44| = (49.36 + 30.6) = 79.96$; thus the third record will also be assigned to the first cluster in the unencrypted case. The remaining data records will be assigned to the nearest cluster following the same process. Therefore, $[r'_4]$ and $[r'_5]$ will be assigned to the second cluster because $[r'_4] - cent'_{1_1} = [[1362.58]]$ and $[r'_4] - cent'_{1_2} = [[322.91]]$ whilst $[r'_5] - cent'_{1_1} = [[1376.79]]$ and $[r'_5] - cent'_{1_2} = [[250.43]]$.

$$sim([[EU]], [r'_x], [cent'_i]) = \sum_{z=1}^{z=2} |[[eu]]_{x,i,z}| \tag{5.9}$$

On completion of the first iteration the following cluster configuration will have been arrived at: $[C_1] = \{[r'_1], [r'_3]\}$ and $[C_2] = \{[r'_2], [r'_4], [r'_5]\}$. The cluster centroids, for the next iteration $Cent'_2 = \{cent'_{2_1}, cent'_{2_2}\}$, will be recalculated as the *means* of the relevant record attribute values. For example, the centroid of the first cluster will be calculated as follows:

$$\begin{aligned}[cent'_{2_{1,1}}] &= (\tfrac{110906.889+110906.889}{2}, \tfrac{-54211.6885-54213.8725}{2}, \tfrac{278661.905+278661.905}{2}) \\ &= (110906.889, -54212.7805, 278661.905)\end{aligned}$$

and:

$$\begin{aligned}[cent'_{2_{1,2}}] &= (\tfrac{110906.889+110906.889}{2}, \tfrac{-54100.9870-53333.8570}{2}, \tfrac{278661.905+278661.905}{2}) \\ &= (110906.889, -53717.422, 278661.905)\end{aligned}$$

The centroid of the second cluster will be calculated in a similar manner:

$$\begin{aligned}[cent'_{2_{2,1}}] &= (110906.889, -53441.1005, 278661.905) \\ [cent'_{2_{2,2}}] &= (110906.889, -53754.0495, 278661.905)\end{aligned}$$

The correctness of the centroids can be established by comparing the decrypted centroid values with calculating plaintext centroid values. The first cluster comprises r_1 and r_3; therefore the plaintext attribute values for the centroid will be:

$$\begin{aligned}cent_{2_{1,1}} &= \tfrac{0.73+0.57}{2} &&= 0.65 \\ cent_{2_{1,2}} &= \tfrac{8.84+65.04}{2} &&= 36.94\end{aligned}$$

The second cluster comprises r_2, r_4 and r_5; therefore the plaintext attribute values for the centroid will be:

$$\begin{aligned}cent_{2_{2,1}} &= \tfrac{49.93+62.15+59.47}{3} &&= 57.183 \\ cent_{2_{2,2}} &= \tfrac{34.44+32.29+36.04}{3} &&= 34.257\end{aligned}$$

Decrypting the HE centroids, using Algorithm 2, gives the same values:

$$\begin{aligned}t &= 0 + 1.5 = 1.5 \\ s &= \tfrac{278661.905}{(3.6+7.9+0)} = 24231.47 \\ cent'_{2_{1,1}} &= (\tfrac{(110906.889-24231.47\times 2.7)}{3.2} + \tfrac{(-54212.7805-24231.47\times 3.1)}{9.1}) \quad /1.5 = 0.65 \\ cent'_{2_{1,2}} &= (\tfrac{(110906.889-24231.47\times 2.7)}{3.2} + \tfrac{(-53717.422-24231.4\times 3.1)}{9.1}) \quad /1.5 = 36.94 \\ cent'_{2_{2,1}} &= (\tfrac{(110906.889-24231.47\times 2.7)}{3.2} + \tfrac{(-53441.1005-24231.47\times 3.1)}{9.1}) \quad /1.5 = 57.183 \\ cent'_{2_{2,2}} &= (\tfrac{(110906.889-24231.47\times 2.7)}{3.2} + \tfrac{(-53754.0495-24231.4\times 3.1)}{9.1}) \quad /1.5 = 34.257\end{aligned} \tag{5.10}$$

Table 5.3: First and second iteration centroids

$[cent'_{1_1}]$	$[(110906.889, -54211.6885, 278661.905), (110906.889, -54100.9870, 278661.905)]$
$[cent'_{1_2}]$	$[(110906.889, -53540.1085, 278661.905\ ), (110906.889, -53751.5470, 278661.905)]$
$[cent'_{2_1}]$	$[(\ 110906.889, -54212.7805, 278661.905), (110906.889, -53717.422, 278661.905)]$
$[cent'_{2_2}]$	$[(110906.889, -53441.1005, 278661.905), (110906.889, -53754.0495, 278661.905)]$

The next stage is to update $[[EU]]$ using a shift matrix calculated from the differences between the initial centroids allocated at start-up, and those determined at the end of

the first iteration (ready for the start of iteration 2). Table 5.3 shows the first and the second sets of centroids, $[Cent'_1]$ and $[Cent'_2]$, for the two clusters. The Shift Matrix $[S']$, encrypted using Liu's FHE scheme, calculated by the TPDM as $[Cent'_1] \ominus [Cent'_2]$, is as follows:

$$[S'] = \begin{bmatrix} (0, 1.092, 0) & (0, -383.565, 0) \\ (0, -99.01, 0) & (0, 2.5025, 0) \end{bmatrix}$$

The corresponding plaintext values, decrypted by the data owner, using Algorithm 2, are then:

$$S = \begin{bmatrix} 0.08 & -28.1 \\ -7.253 & 0.183 \end{bmatrix}$$

The values re-encrypted, using the proposed FDH-OPE scheme, and sent to the TPDM by the data owner, will then be as follows:

$$[[S']] = \begin{bmatrix} [[s'_{1,1}]] & [[s'_{1,2}]] \\ [[s'_{2,1}]] & [[s'_{2,2}]] \end{bmatrix} = \begin{bmatrix} 2.019 & -495.275 \\ -162.982 & 4.331 \end{bmatrix}$$

The re-encrypted $[[S']]$ is then used to update $[[EU]]$:

$$[[EU]] = \begin{bmatrix} ([[eu_{1,1,1}]] + [[s'_{1,1}]], [[eu_{1,1,2}]] + [[s'_{1,2}]]) & ([[eu_{1,2,1}]] + [[s'_{2,1}]], [[eu_{1,2,2}]] + [[s'_{2,2}]]) \\ ([[eu_{2,1,1}]] + [[s'_{1,1}]], [[eu_{2,1,2}]] + [[s'_{1,2}]]) & ([[eu_{2,2,1}]] + [[s'_{2,1}]], [[eu_{2,2,2}]] + [[s'_{2,2}]]) \\ ([[eu_{3,1,1}]] + [[s'_{1,1}]], [[eu_{3,1,2}]] + [[s'_{1,2}]]) & ([[eu_{3,2,1}]] + [[s'_{2,1}]], [[eu_{3,2,2}]] + [[s'_{2,2}]]) \\ ([[eu_{4,1,1}]] + [[s'_{1,1}]], [[eu_{4,1,2}]] + [[s'_{1,2}]]) & ([[eu_{4,2,1}]] + [[s'_{2,1}]], [[eu_{4,2,2}]] + [[s'_{2,2}]]) \\ ([[eu_{5,1,1}]] + [[s'_{1,1}]], [[eu_{5,1,2}]] + [[s'_{1,2}]]) & ([[eu_{5,2,1}]] + [[s'_{2,1}]], [[eu_{5,2,2}]] + [[s'_{2,2}]]) \end{bmatrix}$$

The result will be:

$$= \begin{bmatrix} (0.0 + 2.019, 0.0 - 495.275) & (-785.23 - 162.982, -457.98 + 4.331) \\ (785.23 + 2.019, 457.98 - 495.275) & (0.0 - 162.982, 0.0 + 4.331) \\ (-3.81 + 2.019, 866.29 - 495.275) & (-787.08 - 162.982, 534.70 + 4.331) \\ (926.74 + 2.019, 435.84 - 495.275) & (274.44 - 162.982, -48.47 + 4.331) \\ (895.71 + 2.019, 481.08 - 495.275) & (214.30 - 162.982, 36.13 + 4.331) \end{bmatrix}$$

$$= \begin{bmatrix} (2.019, -495.275) & (-948.212, -453.649) \\ (787.249, -37.295) & (-162.982, 4.331) \\ (-1.791, 371.015) & (-950.062, 539.031) \\ (928.759, -59.435) & (111.458, -44.139) \\ (897.729, -14.195) & (51.318, 40.461) \end{bmatrix}$$

The next iteration can now commence. Each record is again assigned to a cluster and the similarity with the updated cluster centroids determined using Equation 5.9. For example, comparing record $[r'_1]$ with $cent'_{2_1}$ and $cent'_{2_2}$ gives respective similarities of [[497.294]] and [[1401.861]]; consequently, $[r'_1]$ is assigned to the first cluster. To check the correctness of this assignment we can compare with the plaintext result. Recall that, the plaintext centroids are given in equation 5.10; in which case $cent_{2_1} - r_1 = (|0.65 - 0.73|, |36.94 - 8.84|) = (0.08 + 28.1) = 28.18$ and $cent_{2_2} - r_1 = (|57.183 - 0.73|, |34.257 - 8.84|) = (56.453 + 25.417) = 81.87$; this confirms the assignment of r_1 to the first cluster. The process will continue in this manner, with subsequent iterations, until the cluster centroids stabilise.

5.6 Experimental Results and Evaluation

This section reports on the experimental analysis conducted to evaluate the operation of the proposed EUDM/EDM mechanisms, the idea of the Cryptographic Ensemble and the secure data clustering algorithm using DBSk-Means and DBSNNC. For the evaluation the same UCI datasets used with respect to the evaluation reported in Chapter 4 were used. The objectives of the evaluation were:

1. To evaluate the Cryptographic Ensemble and the concept of EUDMs/EDMs in terms of four evaluation criteria: (i) extent of data owner participation, (ii) clustering efficiency, (iii) comparative clustering accuracy and (iv) security.
2. To compare the operation of EUDMs with the concept of UDMs presented in Chapter 4 in terms of the complexity of data owner participation, efficiency, accuracy and security (comparison of the operation of EDMs with UDMs was deemed inappropriate because they used different clustering techniques).

As in the case of the evaluation presented in Chapter 4 the number of classes in a dataset were used to indicate the desired number of clusters to be produced by DBSk-Means (the k value). The DBSNNC parameter value (σ) was randomly selected from a sequence of experiments (not reported here) conducted using different values for σ although in practice the σ value would be selected by the data owner. The outcomes from the experiments related to the four criteria associated with the first objective are presented and discussed in Sub-sections 5.6.1 to 5.6.4, while the outcomes from the experiments related to the second objective are presented in Sub-section 5.6.5.

5.6.1 Data Owner Participation

The first criteria used to evaluate the Cryptographic Ensemble idea and the concept of EUDMs/EDMs was to analyse the processing time required by the data owner to prepare the dataset for the TPDM and to participate whilst the data clustering algorithm was in progress. Table 5.4 presents the runtime values obtained to prepare the selected UCI datasets when utilising both EUDMs and EDMs. Recall that, when using EUDMs the data preparation comprises: (i) data encryption (*Data Enc.*), (ii) UDM calculation (*UDM Cal.*) and (iii) UDM encryption (*UDM Enc.*), to arrive at an EUDM; whilst when using EDMs the preparation process comprises: (i) data encryption (*Data Enc.*), (ii) DM calculation (*DM Cal.*) and (iii) DM encryption (*DM Enc.*), to arrive at an EDM. The *Data Enc.* and *UDM Cal.* sub-process are the same as those used with respect to UDMs as described in the previous chapter, Chapter 4, because an EUDM is an extension of a UDM. For completeness, the runtimes for *Data Enc.* and *UDM Cal.* (that were previously presented in Table 4.2) are included in columns 2 and 3 of Table 5.4.

From the table, it can be observed that the overall runtime required to prepare data when using an EUDM was higher than the overall runtime required to prepare data using an EDM, as indicated by inspection of columns 5 and 8. Note that the runtimes using EUDMs is given in Seconds (*Sec.*); whilst that for EDMs is given in milli-seconds (ms). The required runtime associated with the UDM Cal. and DM Cal. sub-processes, as shown in columns 3 and 6, is negligible; the UDM/DM can be calculated in a matter of milli-second, although the time required to calculate the UDM is larger. More formally, the time complexity for calculating an initial UDM is in the order of $O(||[D']|| \times ||[D']|| \times a)$; if we know the value for k in advance this can be reduced to $O(||[D']|| \times k \times a)$. The complexity of the DM calculation, in turn, is given by $O(||[D']|| \times ||[D']||)$. The times to

TABLE 5.4: Time required by the data owner to prepare data for outsourcing. The reported results are an average of ten runs

No. UCI Dataset	Data Enc. (ms)	EUDM			EDM		
		UDM Cal. (ms)	UDM Enc. (Sec)	Total App. (Sec)	DM Cal. (ms)	DM Enc. (ms)	Total App. (ms)
1. Arrhythmia	9.80	230.20	45.90	46.13	47.70	168.80	216.50
2. Banknote Auth.	1.85	388.00	9.00	9.39	27.30	1485.40	1512.70
3. Blood Trans.	4.04	62.00	1.90	1.96	11.30	252.70	264.00
4. Breast Cancer	1.70	11.00	1.50	1.51	5.67	58.10	63.77
5. Breast Tissue	0.40	3.00	0.20	0.20	1.70	34.40	36.10
6. Chronic Kidney	2.65	29.00	3.80	3.83	8.30	144.60	152.90
7. Dermatology	1.80	40.20	3.40	3.44	10.30	102.70	113.00
8. Ecoli	0.98	8.80	0.70	0.71	4.23	121.20	125.43
9. Indian Liv. Pat.	1.30	41.80	3.10	3.14	9.20	239.10	248.30
10. Iris	0.20	2.00	0.20	0.20	1.90	50.90	52.80
11. Libras Mov.	4.80	49.80	10.40	10.45	13.40	123.50	136.90
12. Lung Cancer	0.59	1.00	0.10	0.10	1.00	11.90	12.90
13. Parkinsons	1.90	6.30	1.10	1.11	3.90	61.30	65.20
14. Pima Disease	1.44	47.00	4.10	4.15	12.50	422.40	434.90
15. Seeds	0.48	5.00	0.50	0.51	2.70	70.00	72.70

encrypt a UDM or DM, to arrive at an EUDM or EDM, are given in columns 4 and 7 of Table 5.4. The time to encrypt a DM is much lower than that for a UDM; note that the time for UDM encryption is given in seconds whilst the time for DM encryption is given in milli-seconds. Although in both cases the encryption takes understandably longer than the other preparation processes, it can be argued that the preparation time does not present a significant overhead.

In addition to data preparation, in the case of DBSk-Means, the data owners will also participate in updating the EUDM through decrypting the Encrypted Shift Matrix (ESM) using Liu's decryption algorithm (Algorithm 2) and re-encrypting the resulting SM using the FDH-OPE scheme on each DBSk-Means iteration. Shift matrices are small, measuring $k \times a$; therefore the time required to decrypt/re-encrypt a shift matrix is negligible. Column 9 of Table 5.5 gives the runtime required to decrypt and re-encrypt shift matrices for the entire DBSk-Means process which, as expected, is slightly higher compared to that required when using shift matrices with respect to UDMs (see Chapter 4).

Some further analysis, in the case of DBSk-Means, was undertaken to evaluate the data owner participation when decrypting and re-encrypting shift matrices in comparison with the number of iterations required for each clustering exercise, and to determine the effect of the number of attributes (a) featured in a given dataset, on runtime. The results are presented in Figure 5.3 which shows two plots. The first plots runtime against attributes a, the second against the number of required iterations. From the plots, it can be seen that, as anticipated, runtime using the EUDM approach is higher compared to the UDM. The figure also shows that the runtime increases with the number of iterations and the number of attributes, although not in any clear linear manner.

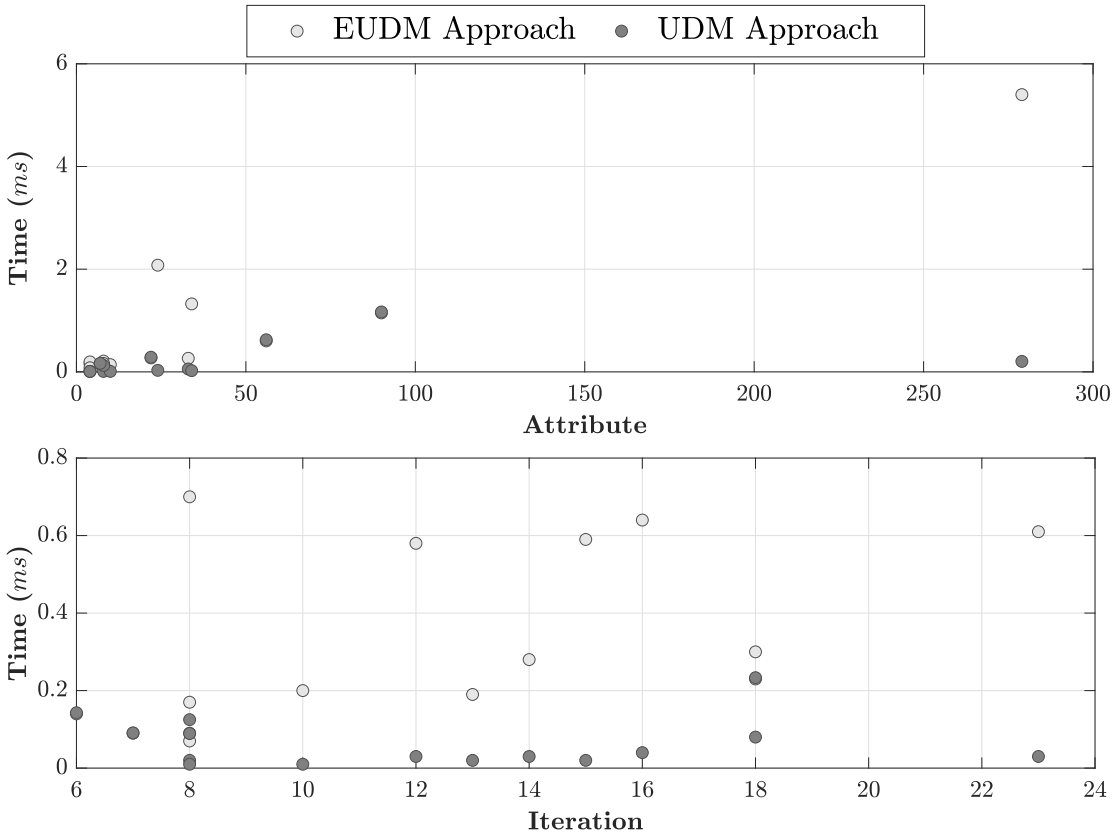


FIGURE 5.3: Time required by the data owner to decrypt shift matrices in the case of the UDM and EUDM approaches coupled with k-Means clustering

5.6.2 Secure Clustering Efficiency

The usage of the homomorphic mathematical properties of the adopted FHE scheme and the secure data comparison founded on the use of EUDMs or EDMs as appropriate, encrypted using the proposed FDH-OPE scheme, will clearly introduce some computational overhead. This section reports on the evaluation conducted regarding the usage of EUDMs in the context of DBSk-Means, and the usage of EDMs in the context of DBSNNC, by considering the reported runtimes and comparing the results with the runtimes required by the equivalent standard algorithms (using plaintext datasets). The runtime required to cluster the datasets using standard k-Means were given in Table 4.3; however, for the purpose of comparison, the results are again presented in columns 3 to 5 of Table 5.5. The table also gives the DBSk-Means runtime (columns 6 to 9). The results obtained for NNC and DBSNNC are given in Table 5.6 (note that the runtimes for DBSk-Means and DBSNNC do not include any data owner preparation time). Inspection of these results indicates that the runtime required by DBSk-Means and DBSNNC was greater than that required using the standard equivalent algorithms; this was to be expected. The largest dataset, Arrhythmia, can be clustered using the DBSk-Means in 5191ms compared to 38ms using standard k-Means, and clustered using DBSNNC in 4924ms compared to 4801ms. From Tables 5.5 and 5.6 it can be observed that the difference in runtime between k-Means and DBk-Means is higher than the difference between NNC and DBSNNC. However, it is argued here, these times were still within acceptable limits.

TABLE 5.5: Experimental results comparing the operation of DBSk-Means and standard k-Means

No.	k	Standard k-Means			DBSk-Means			
		Runtime (ms)	Num. Iter.	Sil. Coef.	Runtime (ms)	Num. Iter.	Sil. Coef.	Re-enc. ESM (ms)
1.	16	38	10	0.602	5191	10	0.602	54.00
2.	2	58	16	0.207	208	16	0.207	2.55
3.	2	19	12	0.370	67	12	0.370	2.32
4.	2	11	8	0.020	71	8	0.020	2.11
5.	6	19	18	0.787	60	18	0.787	2.63
6.	2	14	8	0.009	138	8	0.009	16.62
7.	6	14	15	0.801	672	15	0.801	19.85
8.	8	6	23	0.628	221	23	0.628	4.81
9.	2	12	13	0.169	274	13	0.169	1.83
10.	3	9	14	0.836	23	14	0.836	1.11
11.	15	26	18	0.590	2626	18	0.590	20.67
12.	3	3	8	0.645	25	8	0.645	4.82
13.	2	5	7	0.079	49	7	0.079	1.90
14.	2	19	8	0.000	128	8	0.000	1.31
15.	3	9	6	0.706	26	6	0.706	0.92

TABLE 5.6: Experimental results comparing the operation of DBSNNC and standard NNC

No.	σ	Standard NNC			DBSNNC		
		Runtime (ms)	Num. Clus.	Sil. Coef.	Runtime (ms)	Num. Clus.	Sil. Coef.
1.	1980.00	4801	16	0.889	4924	16	0.889
2.	11.00	3444	3	0.752	3462	3	0.752
3.	1046.00	515	4	0.895	540	4	0.895
4.	20.00	597	6	0.470	668	6	0.470
5.	990.00	3	38	0.999	4	38	0.999
6.	952.00	329	16	0.981	359	16	0.981
7.	26.00	344	8	0.745	368	8	0.745
8.	0.76	78	7	0.881	82	7	0.881
9.	100.00	449	98	0.997	481	98	0.997
10.	3.00	6	2	0.722	6	2	0.722
11.	10.00	765	19	0.753	822	19	0.753
12.	0.10	2	32	1.000	2	32	1.000
13.	91.00	34	8	0.930	38	8	0.930
14.	498.00	1068	2	0.741	1129	2	0.741
15.	4.00	18	2	0.579	19	2	0.579

5.6.3 Secure Clustering Accuracy

The "correctness" of the clustering configurations produced using the proposed secure data clustering, DBSk-Means and DBSNNC, were compared using the established Silhouette Coefficient (Sil. Coef.) metric calculated as explained in Sub-section 4.4.3. As already noted, the intuition was that the secure algorithms should produce equivalent (or comparable) configurations to those produced using equivalent standard algorithms, and if so the secure algorithms could be said to be operating correctly. The Sil. Coef. for k-Means and DBSk-Means are shown in columns 5 and 8 of Table 5.5 respectively, whilst the Sil. Coef. for NNC and DBSNNC are shown in columns 5 and 8 of Table 5.6. The results show that identical Sil. Coef. values were obtained in all cases. Thus indicating that the final cluster configurations produced using DBSk-Means and DBSNNC were similar to those produced using the corresponding standard algorithms. This, therefore, indicates that the proposed mechanisms can be used to provide a solution to the challenge of DMaaS.

The number of cluster iterations required using k-Means/DBSk-Means clustering was also compared and also the number of produced clusters using NNC/DBSNNC. The results are shown in columns 4 and 7 of Table 5.5, and columns 4 and 7 of Table 5.6. The results show that the number of iterations and number of clusters produced were the same in all cases, although (in the case of k-Means) the bigger the dataset the more iterations that were typically required regardless of whether secure or standard clustering was applied.

5.6.4 Security

In this section the security of the proposed EUDM/EDM approaches and Cryptographic Ensemble idea are discussed in terms of potential attacks. In the proposed solution, the TPDM has access to: (i) the encrypted dataset $[D']$, (ii) the encrypted UDM $[[EU]]$ or encrypted DM $[[ED]]$ and (iii) the encrypted and re-encrypted shift matrix $[S']/[[S']]$, thus *Cyphertext Only Attacks* (COAs), *Chosen Plaintext Attacks* (CPAs) and Knowing Plaintext Attack (KPA) are theoretically possible. These attacks can also be lunched when the external attackers somehow have access to the TPDM storage, and by an adversary, who is also a TPDM, who is able to send the ESM for re-encryption. In the following, the potential attacks are discussed.

A COA is where the attacker has access only to cyphertexts; no access to any plaintext data. In the proposed approach, this attack can be instigated whenever an attacker has access to the encrypted data $[D']$ and/or the corresponding EUDM/EDM ($[[EU]]/[[ED]]$). The $[D']$ and $[[EU]]/[[ED]]$ are encrypted using one of the schemes within the Cryptographic Ensemble; either Liu's FHE scheme or the proposed FDH-OPE scheme. Recall that Liu's FHE scheme is a probabilistic scheme that produces different cyphertexts for the same plaintext value each time it is applied, even when using the same secret key. This feature means that Liu's cyphertext is semantically secure and hence accessing cyphertexts does not provide any useful information with respect to the associated plaintexts from the perspective of an adversary.

With respect to EUDMs/EDMs a COA is the most likely form of attack. However, in practice, it is a very weak form of attack because of the adversary's lack of information. In the unlikely event that an adversary has background knowledge of the data distribution, or data frequency of the original data values, it might be possible to apply some form of statistical analysis or frequency analysis to an EUDM/EDM (secured using the FDH-OPE scheme) to gain information. For instance, by extracting statistical information from the cyphertexts, the adversary might be able to infer ranges containing

dense data. Using frequency analysis, an adversary could highlight cyphertexts bearing the same frequency as plaintexts (if such plaintexts were available) and then identify some cyphertexts values that have the same frequency. To guard against this possibility, the proposed FDH-OPE scheme firstly incorporates both *message space splitting* and *non-linear cyphertext space expansion* so as to obscure the statistical features of the generated cyphertexts (such as data distribution). Secondly, the proposed FDH-OPE scheme hides the data frequency using a "one-to-many" mapping function incorporating a random variable δ.

A CPA is where an attacker chooses random plaintexts to be encrypted so as to obtain the corresponding cyphertexts for further analysis (refer to Section 2.5 for more detail). In the proposed approaches, the only possible interaction with the data owner that could be used to launch the CPA attacks is the EUDM updating process when sending the ESM for re-encrypting. The attacker may send the chosen set of plaintexts, to be encrypted to FDH-OPE cyphertexts, in a form of shift matrix $[S']$. However, as specified in Algorithm 7, for the shift matrix $[S']$ to be processed correctly by the data owner it needs to be encrypted using Liu's FHE scheme. Encrypting $[S']$ requires access to the Liu's FHE secret key; however, this is not shared with other parties as the scheme is symmetric.

A KPA is when the attacker is able to build knowledge of cyphertexts and their corresponding plaintexts values. In this attack the attacker cannot choose the value of the desired plaintext, as in the case of CPA, instead the plaintexts is only known. A KPA is only possible when data is exchanged with a data owner in cyphertext form and received in plaintext form. When using an EDM a KPA cannot be launched as there is no data exchanged with the data owner. When using an EUDM, the data exchanged with the data owner is in cyphertext form, and thus the attacker will only have HE cyphertexts and their corresponding FDH-OPE cyphertexts.

Given the above, it is argued that the proposed EUDM and EDM approaches are secure against COAs, CPAs and KPAs.

5.6.5 Comparison Between UDM and EUDM

The second evaluation objective was to compare the operation of UDMs to EUDMs. Comparison with EDMs was not conducted as it does not support the updating process required for k-Means clustering. The idea underpinning the evaluation was to compare the performance of standard k-Means clustering with secure k-Means clustering algorithms founded on EUDM (DBSk-Means) and the UDM (Sk-Means) from the previous chapter. The evaluation criteria were: (i) data owner participation, (ii) clustering efficiency, (iii) clustering accuracy and (iv) security.

The data owner participates for both secure clustering mechanisms was limited to preparing the data for outsourcing and decrypting and re-encrypting the shift matrix on each k-Means iteration. The data preparation process comprised data encryption, UDM calculation and, in the case of EUDMs, encryption. Therefore, the data owner participation in the EUDM case added an encryption process compared to using UDMs. Tables 4.2 and 5.4 show the overall preparation runtime using UCI datasets in the context of UDMs and EUDMs, respectively. For comparison purposes, the results are also presented in Figure 5.4 in such a way that the dataset ID numbers used in Tables 4.2 and 5.4 are listed along the x-axis. The figure clearly shows the difference in complexity. In most cases EUDMs required more preparation time than UDMs, although the results show that the preparation time did not entail any significant overhead on behalf of the data owner. In addition to data preparation, data owners will also participate

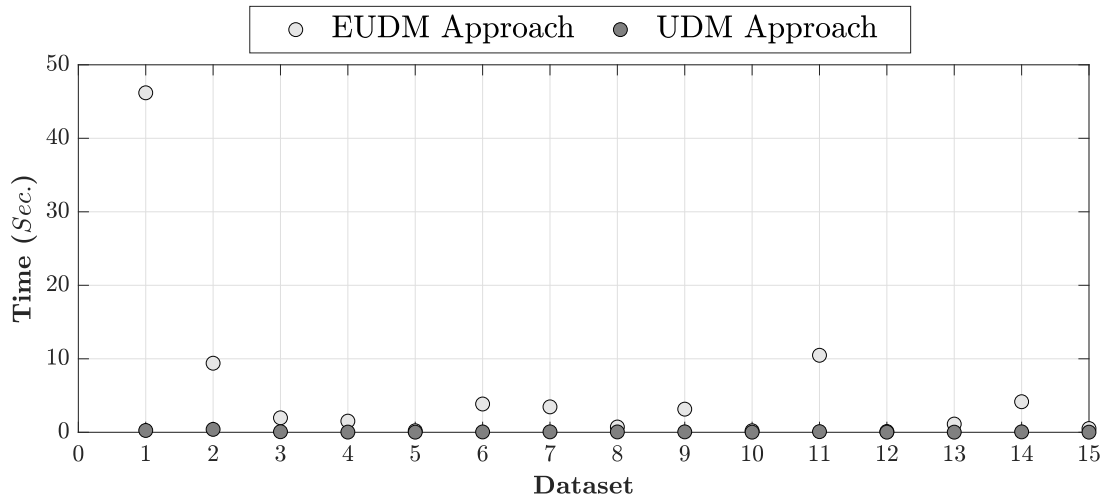


FIGURE 5.4: Compare the complexity of data owner participation in the context of UDMs and EUDMs

in updating UDMs and EUDMs through decrypting the shift matrix on each k-Means iteration and re-encrypting the results using the FDH-OPE scheme in the case of EUDMs. Therefore, participation can be expected to be slightly higher in the EUDM case than in the UDM case. The data owner participation to decrypt (and re-encrypt) shift matrices were analysed with respect to the number of iterations and attributes as shown in Figure 5.3. The figure shows an increase in the runtime as the number of attributes and iterations increase, although again not in any clear linear manner.

The runtime required for clustering data by utilising an EUDM, and the Cryptographic Ensemble, was compared with that required using a UDM. Both approaches were also compared with the standard k-Means runtime. The results were reported in Table 4.3 for standard k-Means compared to Sk-Means (using UDMs), and are given in Table 5.5 for standard k-Means compared to DBSk-Means (using EUDMs). From the tables it can be observed that the runtimes to cluster datasets using DBSk-Means and Sk-Means was higher than the time required to cluster the same datasets using the standard algorithm. This result was expected as the secure algorithms use the HE mathematical properties instead of standard operations, hence an overhead is introduced. The DBSk-Means runtime was slightly higher than that of Sk-Means. This was also expected as DBSk-Means operates using the Cryptographic Ensemble; the encryption of the UDM and shift matrix introduced another overhead. Interestingly, the number of iterations using Sk-Means, DBSk-Means and standard k-Means were all the same in the context of the considered UCI datasets.

The "accuracy" of the generated cluster configurations produced using UDMs and EUDMs, Sk-Means and DBSk-Means, was also compared with that produced using the standard algorithms. The Silhouette Coefficient was used as an evaluation measure. From Table 5.5, and with reference to Table 4.3, it can be seen that the clustering configurations were identical in all three cases algorithms (k-Means, Sk-Means and DBSk-Means). Indicating that the final clustering configurations produced using Sk-Means and DBSk-Means were identical to those produced using standard k-Means. Therefore, it was concluded that the Cryptographic Ensemble and their secure mathematical properties did not adversely affect the correctness of the cluster configurations generated.

In term of security, as noted in the foregoing chapter, the unencrypted UDM represents a very large set of linear equations that introduce a security risk when disclosed

in their plaintext form. An attacker who somehow has access to the UDM might be able to obtain some statistical information concerning the original data by reverse engineered the UDM's linear equations. The other potential threat that exists is with respect to the UDM updating process; in particular the process of sending an encrypted shift matrix $[S']$ and receiving the corresponding plaintext shift matrix S values. This might allow attackers to develop knowledge using the plaintexts and their corresponding HE cyphertexts. Therefore, it can be argued that the UDM approach is vulnerable to Known Plaintext Attacks (KPAs) and provide potential for reverse engineer of statistics concerning the data. A UDM is also vulnerable to dictionary attack whereby the adversary generates new cyphertexts and then derives their plaintext equivalent values using the HE properties and knowledge obtained when the shift matrix cyphertext is decrypted. This is prevented by encrypting the UDM elements to produce an EUDM.

5.7 Summary

This chapter has presented the idea of Cryptographic Ensembles and the twin concepts of Encrypted Updatable Distance Matrix (EUDM) and Encrypted Distance Matrices (EDM) that collectively provide a solution for supporting PPDM in the context of DMaaS. The proposed Cryptographic Ensemble seeks to address the security issue associated with the UDM concept presented in Chapter 4 whilst allowing data clustering to be conducted over encrypted data without decryption. The utility of the proposed EUDM/EDM approach was illustrated using two secure data clustering mechanism, DBSk-Means which used EUDMs, and DBSNNC which used EDMs. The evaluation demonstrated that the Cryptographic Ensemble could provide a solution to secure outsourced data clustering while maintaining the clustering effectiveness. The data owner participation with respect to DBSk-Means was limited to decrypting and re-encrypting the data using the FDH-OPE scheme, whilst data owner participation was entirely avoided using DBSNNC once the data had been prepared and the data clustering commenced. The evaluation also indicated that, although the encrypted approaches required more runtime (as expected), the runtime was not significant. The main advantage of the EUDM/EDM approach over the UDM approach, was that it was secure against a variety of attacks, including: (i) Cyphertext Only Attacks (COAs), (ii) Chosen Plaintext Attacks (CPAs) and (iii) Known Plaintext Attacks (KPAs). In the next chapter, an extension of the proposed EDM approach is presented to facilitate secure collaborative data clustering (the multiple data owners scenario).

Chapter 6

Global Encrypted Distance Matrices

6.1 Introduction

The work presented in this chapter is directed at providing an answer to the subsidiary research question originally posed in Chapter 1:

- Can the proposed single data owner scenario solutions be extended to support scalable collaborative data mining (the multiple data owners scenario) while keeping the data owner participation at a minimum?

The fundamental idea presented in this chapter is that of a Global Encrypted Distance Matrix (GEDM) generated from a collection of locally generated distance matrices produced by multiple data owners. To facilitate the generation of a GEDM, and its utilisation, the idea of a Cryptographic Ensemble was maintained but with the Frequency and Distribution Hiding OPE (FDH-OPE) scheme from the previous chapter replaced with a bespoken Multi-Users Order Preserving Encryption (MUOPE) scheme that preserves the order of data distributed across multiple sources. The application focus for the work presented in this chapter continuous to be data clustering. However, as in the case of EDMs, GEDMs are not updatable so not suited to secure k-Means style clustering, but entirely appropriate to DBSCAN and NNC style clustering. Secure classification will be considered later in this thesis.

Generally, the nature of collaborative data analysis introduces additional privacy preservation concerns compared to the single data owner scenario as discussed in [49]. The reader might find it useful to refer back to Section 2.3 where the data privacy preservation requirements for the single and multiple data owner scenarios were discussed. The main challenge in collaborative data clustering is that of maintaining the confidentiality of the data during processing without adversely affecting the effectiveness of the analysis. For example, in the case of many data clustering algorithms, when calculating distances between records and when comparing such distances against each other or against a threshold. From the literature, as already noted in Chapter 2, two different categories of solutions can be identified. The first is to resort to well-established Secure Multi-Party Computation (SMPC) protocols to securely calculate the comparator values required by individual data mining algorithms [20, 33, 36–40, 56, 103, 105]. The second is to adopt an appropriate Fully Homomorphic Encryption (FHE) scheme that splits the secret key among the multiple parties so that each party can locally encrypt their dataset. In this case the TPDM can manipulate the dataset using the FHE mathematical properties of the adopted scheme [19, 27, 28]. Although, to a certain extent, SMPC

and secret sharing address the problem of data confidentiality, the requirement for data owner participation, as the data mining progresses, results in a significant computation and communication overhead. This disadvantage consequently effects the scalability of the approaches rendering them infeasible for any form of large scale collaborative data mining. The work presented in this chapter seeks to address this disadvantage using the concept of GEDMs which allow secure collaborative data clustering facilitated by the proposed Multi-Users Order Preserving Encryption (OPE) scheme.

The remainder of this chapter is structured as follows. Section 6.2 presents the proposed MUOPE scheme designed to facilitate data order preservation in the context of the multiple data owners scenario. In Section 6.3 the idea of GEDMs, founded on the EDM concept first presented in Chapter 5, and designed to support multiple data owner scenario and collaborative data clustering, is presented. The application of GEDMs in the context of secure data clustering is detailed in Section 6.4; two specific approaches are considered, Secure DBSCAN (S-DBSCAN) and Secure NNC (S-NNC). Section 6.5 presents an extensive evaluation of the GEDM idea and the supporting MUOPE scheme. Finally, Section 6.6 provides a summary of the material presented in this chapter.

6.2 Multi-Users Order Preserving Encryption Scheme

From earlier work presented in the foregoing chapters a Distance Matrix (DM), generated by a single data owner, as described in Chapter 5, is essentially a set of linear equations that might support the undesirable re-engineering of the original data distribution. The proposed solution was to encrypt the DM using the bespoken FDH-OPE scheme which hides the data distribution and frequency in the generated cyphertexts, while still permitting comparison of distances. However, the FDH-OPE scheme, part of the Cryptographic Ensemble detailed in Chapter 5, could not provide a solution in the context of the multiple data owners scenario. The generated FDH-OPE cyphertexts would preserve the order for the DM belonging to a single data owner but not the order across all DMs were these DMs to be somehow combined. The MUOPE scheme was therefore proposed that will support order preservation across multiple DMs brought together into a single GEDM. The proposed MUOPE scheme is presented in detail in this section.

The MUOPE scheme can be seen as an extension of the FDH-OPE scheme presented previously in Section 5.2 in that the MUOPE scheme shares many of the FDH-OPE scheme's main features; data distribution and data frequency hiding in the generated cyphertexts. The MUOPE scheme is supported by two processes; Key Generation `KeyGen` and data encryption `Encrypt`. There is no decryption process as this is not required, at least in the context of the collaborative data clustering presented in this chapter. The MUOPE key is generated by a Semi-honest Third Party (STP) who acts as a mediator between the participating data owners (the set of participating parties $P = \{p_1, \ldots, p_u\}$). Once the key is generated, the scheme key will be sent to all participating parties who will locally encrypt their DM to arrive at an EDM.

Algorithm 11 gives the pseudo code for the MUOPE `KeyGen` process. The inputs are the message space interval $\mathcal{M}$ and the associated cyphertext space interval $\mathcal{C}$ as depicted previously in Figure 5.1 in Chapter 5. Each interval comprises maximum and minimum interval boundaries; l, l' and h, h' for the message space and corresponding cyphertext space, respectively. The interval boundaries are as agreed by the participating data owners selected in such a way that $|\mathcal{M}| \ll |\mathcal{C}|$.

The algorithm commences with the STP splitting the message space interval ($\mathcal{M}$) into t consecutive intervals, where t is a random number, to arrive at $\mathcal{M}interval =$

Algorithm 11 MUOPE key generation process

1: **procedure** KeyGen($\mathcal{M}, \mathcal{C}$)
2: $\mathcal{M}interval$ ←Message Space Splitting($\mathcal{M}$, t)
3: $Dens$ = SecureDensityAccumulation(t, $\mathcal{M}interval$)
4: $\mathcal{C}interval$ ← Non-linear Cypher Space expansion($Dens$, $\mathcal{C}$)
5: **Exit** with $\mathcal{C}interval$ and $\mathcal{M}interval$
6: **end procedure**
7: **procedure** SecureDensityAccumulation(t, $\mathcal{M}interval$)
8: **STP**: generates SK, PK Paillier's key
9: **STP**: shares the Paillier's public key (PK) and $\mathcal{M}interval$ with all participants
10: **STP**: declares $Vinitial$ as a set of t elements initialised randomly
11: **STP**: declares V as a set of t elements and assign $Vinitial$ to it
12: **STP**: V'= `Encrypt`(V, PK) ▷ Paillier encryption Equation 3.7
13: **STP**: sends V' to p_1
14: $\mathbf{p}_1$: $dens_1$= calculate density in DM_1 for each message space interval ($\mathcal{M}interval$)
15: $\mathbf{p}_1$: $dens'_1$= `Encrypt`($dens_1$, PK) ▷ Paillier encryption
16: $\mathbf{p}_1$: $V' = V' \oplus dens'_1$
17: **for** $i = 2$ to $i = u$ **do**
18: $\mathbf{p}_{i-1}$: will send V' to p_i
19: $\mathbf{p}_i$: $dens_i$ = calculate density in DM_i for each message space interval
20: $\mathbf{p}_i$: $dens'_i$= `Encrypt`($dens_i$, PK) ▷ Paillier encryption
21: $\mathbf{p}_i$: $V' = V' \oplus dens'_i$
22: **end for**
23: $\mathbf{p}_u$: will send V' to STP
24: **STP**: $Dens$ = `Decrypt`(V',SK) ▷ Paillier keys
25: **STP**: $Dens = Dens - Vinitial$
26: **Exit** with $Dens$
27: **end procedure**

$\{m_1, \ldots, m_t\}$ (line 2). As in the case of FDH-OPE, the cyphertext interval needs to split into the same number of intervals t. For the data distribution to be hidden, the message space intervals that have a large data density need to be related to large cyphertext space intervals. Therefore, again as in the case of FDH-OPE, the length of each cyphertexts space interval is determined according to the density of the data within the corresponding message space interval. The density for each interval in the message space, representing data distributed across multiple data owners, is securely accumulated by calling the *SecureDensityAccumulation* sub-procedure in line 3 which is given in lines 7 to 27 in the algorithm. In the pseudo code for the *SecureDensityAccumulation* sub-procedure STP indicates a command executed by the STP and p_i a command executed by data owner (party) p_i.

The inputs to the *SecureDensityAccumulation* sub-procedure are the number of intervals (t) and message space intervals $\mathcal{M}interval = \{m_1, \ldots, m_t\}$. The sub-procedure commences with the STP generating Paillier key pairs, SK and PK (line 8); recall that detail concerning Paillier encryption was presented in Sub-section 3.3.2 of Chapter 3. The generated public key PK and $\mathcal{M}interval$ are sent to all participating parties (line 9). The next step is for the STP to create a list $Vinitial$ of t elements that is initialised randomly (line 10). The STP then creates a list V of t elements corresponding to $\mathcal{M}interval$ and $\mathcal{C}interval$, $V = \{v_1, \ldots, v_t\}$ (line 11) which on start-up is initialised

with the values in $Vinitial$. The list V is encrypted using Paillier Encryption to give an encrypted list V' (line 12). The encrypted list V' will be then sent to the first data owner (party), p_1, who will calculate the density ($dens_1$) values with reference to his/her DM, DM_1 (lines 13 and 14). Data owner p_1 will then encrypt these values (again using Paillier encryption) and add them to the values held in V' using the Paillier HE additive property, $\oplus$ (lines 15 and 16). A loop is then entered (lines 17 to 22) in which the density of the data for each interval for the remaining participants p_2 to p_u is securely accumulated. The loop commences by sending the updated list V' to the next data owner who will calculate the data density with the reference to their DM, encrypt the density values and add the encrypted values to V' using the HE additive property. The process will be repeated for the remaining parties till there are no more data owners to be considered. The last party, p_u, will return V' to the STP (line 23). The STP then decrypts V' (line 24) using the Paillier scheme secret key to arrive at $Dens$ and subtracts the original random values $Vinitial$, first used to populate V, to give the density value for each interval (line 25). The $SecureDensityAccumulation$ sub-procedure will exit with data density $Dens$ (line 26). The $Dens$ values are then used, by the STP, to determine the length of each cyphertext space interval (line 4) by performing Non-linear Cyphertext Space expansion (as discussed in Section 5.2). The message space and cyphertext space interval boundaries that represent the MUOPE encryption keys are the returned in line 5. The MUOPE keys are used by individual data owners to locally encrypt their DM. The encryption function is the same as that given for FDH-OPE scheme; see Algorithm 6.

6.3 Global Encrypted Distance Matrices

This section presents the GEDM concept. As in the case of EDMs, a GEDM is a 2D matrix that holds distances between every record in the global dataset $GData$ with every other record in $GData$; $GData = \cup_{i=1}^{i=u} D'_i$, where D'_i is the dataset belonging to data owner (participant party) p_i which is encrypted using Liu's FHE scheme, and u is the number of parties. In other words, $GData$ is the union of the participating encrypted datasets and a GEDM is the combination of the associated EDMs where the EDMs are generated as described in Chapter 5. The GEDM is constructed by the STP. How the GEDM is constructed depends on how the data is partitioned across the participants. As noted in Chapter 2 (Sub-section 2.3.2), three different data partitionings can be identified; horizontal, vertical and arbitrary. This chapter considers the horizontal data partitioning, where each partition conforms to the same set of attributes A, but features different records. Alternative partitionings are considered in [39, 94, 102].

Recall that a DM, as presented in the previous chapter, is a 2D matrix that holds distances (differences) between each record in a dataset D with every other record in D (where D belongs to a single party). Therefore, a DM's dimensions are defined by the number of records in D. The matrix is symmetric about the leading diagonal so only values for the lower (or upper) triangular of the matrix need to be calculated. An EDM is then an encrypted DM that has been encrypted using the MUOPE scheme described in Section 6.2. Each participant p_i generates an EDM, EDM_i, for their dataset D_i. The resulting set of EDMs, $\mathbf{EDM} = \{EDM_1, EDM_2, \ldots, EDM_u\}$, are used to construct the desired GEDM. This is not simply a matter of appending EDMs to one another, because to be used as intended a GEDM needs to hold the encrypted distances between every record in $GData$ to every other record in $GData$, thus most of the content of a GEDM will hold encrypted distances between data records which belonging to different data owners. Figure 6.1 indicates (dark grey boxes) where the content of EDMs can be

used directly to populate a GEDM; along the leading diagonal. The remainder of the GEDM needs to be populated using a process, referred to as *pooling*, conducted between pairs of data owners as indicated by the examples also highlighted (light grey boxes) in Figure 6.1.

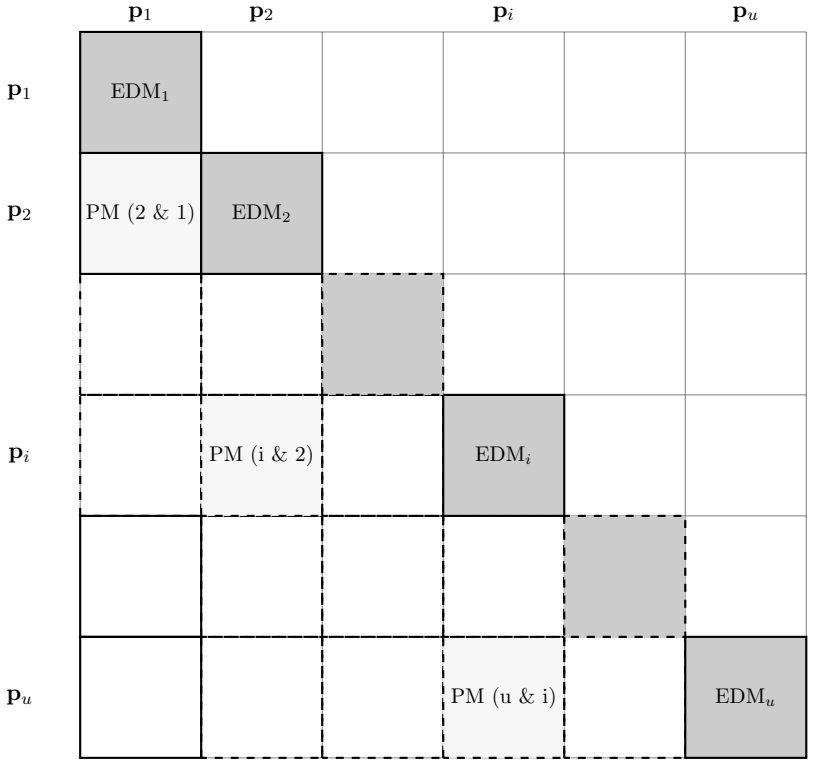


FIGURE 6.1: A GEDM highlighting (dark grey boxes) where the content of EDMs can be directly incorporated and examples (light grey boxes) of where pooling takes place between individual data owners.

The pseudo code for the pooling process is given in Algorithm 12. The process comprises two steps. The first uses the global set of EDMs, **EDM**, to populate the "leading diagonal band" of the GEDM, **GE** (lines 4 to 13). The second step requires data owner participation to generate Pool EDMs (PoolEDMs) that are then used to populate the remaining **GE** elements (lines 14 to 33). The pooling method is managed by the STP. The inputs to pooling sub-process are the global set of EDMs, **EDM**, and the number of attributes in the dataset a. The first step (lines 2 and 3) is for the STP to dimension the **GE** according to the number of records in $GData$ (given that only horizontal portioning is considered in this chapter this will be the sum of the number of records in each EDM belonging to each participant). Next, the distance values in each participant's EDM are inserted into **GE** (lines 4 to 13) thus populating the GEDM's "leading diagonal band" with the existing encrypted distances as shown in Figure 6.1. Note that in the pseudo code the notation $|EDM_p|$ indicates the number of records in EDM_p and not the overall size of EDM_p. The next stage (lines 14 to 33) is to populate the remainder of the GEDM with the distances between records held by pairs of data owners. The number of pairs will be $\frac{u(u-1)}{2}$. For each pair (i and j), a Pooled Matrix (PM) is constructed (line 16). A PM is a 3D matrix designed to hold the encrypted difference between each attribute in each record held by a data owner (p_i) and each record held by another owner (p_j). The dimensions for a PM are thus: the number of records held by p_j, the number of records held by p_i and the number

of attributes a in $GData$ (recall that horizontal partitioning features the same set of attributes across the participants). The PM will then be populated with a set of random values. Next (line 17) the PM is encrypted (using the MUOPE) to give PM', which is then sent to the data owners p_i and p_j who each create temporary EDMs holding the differences between their encrypted records and the contents of PM'. Data owner p_i will calculate the distance between each value in PM', using their MUOPE cypher, with the corresponding encrypted attribute value in their dataset D_i to produced PM'_1 (line 18); data owner p_j will do the same with respect to their dataset D_j to give PM'_2 (line 19). Both are returned to the STP where they are used to create a pooled EDM by adding the PM'_1 and PM'_2 elements to give $PoolEDM$ (line 20) which is then used to populate the appropriate section of **GE**. The relevant "start" indexes into **GE** are first calculated (lines 21 and 22) and then the pooled EDM is processed and used to populate the appropriate section of the **GE** (lines 23 to 31). The process will be repeated until **GE** has been fully populated.

A GEDM, once generated, can be used to determine the data similarity between any pair of records in the global dataset. Equation 6.1 is used to derive the "order" of similarity between encrypted records r'_x and r'_y regardless of who the records belong to.

$$sim(\mathbf{GE}, r'_x, r'_y) = ge_{x,y} \tag{6.1}$$

6.4 Secure Data Clustering Using GEDMs

Using a GEDM and the proposed Cryptographic Ensemble, collaborative data mining activities can be outsourced securely to a TPDM. Recall, in this case that the Cryptographic Ensemble comprises Liu's FHE scheme, described in Sub-section 3.3.1, and the MUOPE scheme described in Section 6.2 of this chapter. The first was used to encrypt the datasets belonging to individual data owners, and the second to encrypt EDMs and hence a GEDM.

To use a GEDM with respect to data mining activities the standard data mining algorithms that operate with unencrypted data need to be modified to work with encrypted data. This is illustrated, in this section, by considering two secure data clustering algorithms: (i) Secure DBSCAN (S-DBSCAN), a secure variation of DBSCAN [47] and (ii) Secure NNC, (S-NNC) a secure variation of NNC [48]. Preparing the dataset for secure collaborative data clustering, regardless of the algorithm used, required: (i) encrypting the datasets, (ii) EDM generation which in turn requires DM calculation and DM encryption to arrive at an EDM and (iii) generating the associated GEDM using the proposed pooling method. Once the datasets were encrypted and the GEDM generated no further data owner participation would be required. In the following two Sub-sections, Sub-sections 6.4.1 and 6.4.2, the S-DBSCAN and S-NNC data clustering algorithms are presented in further detail.

6.4.1 Secure DBSCAN

The pseudo code for the S-DBSCAN algorithm, founded on the use of a GEDM, is presented in Algorithm 13. The processing is entirely conducted by a TPDM, there is no data owner involvement. The inputs are: (i) an encrypted global dataset $GData'$ collated by the TPDM; (ii) a GEDM **GE**; and (iii) the DBSCAN minimum number of points and radius parameters, $MPts$ and ϵ', agreed by the participating parties. The dataset $GData'$ is built from the participating parties' datasets using the same ordering as used to build **GE**. To allow secure comparison using **GE**, the ϵ parameter value is

Algorithm 12 Pooling Methods for GEDM generation

1: **procedure** POOLINGMETHOD(**EDM**,a)
2: $GDataSize = \sum_{p=1}^{p=u} |EDM_p|$ ($EDM_p \in$ **EDM**)
3: **GE** = 2D matrix measuring $GDataSize \times GDataSize$
4: $currentRow = 0$, $currentCol = 0$
5: **for** $p = 1$ to $p = u$ **do**
6: **for** $r = 1$ to $r = |EDM_p|$ **do**
7: **for** $c = 1$ to $c = |EDM_p|$ **do**
8: $ge_{currentRow+r,currentCol+c} = EDM_{p[r,c]}$ ($ge_{x,y} \in$ **GE**)
9: **end for**
10: **end for**
11: $currentRow = currentRow + |EDM_p|$
12: $currentCol = currentCol + |EDM_p|$
13: **end for**
14: **for** $i = 1$ to $i = (u - 1)$ **do**
15: **for** $j = (i + 1)$ to $j = u$ **do**
16: PM = 3D matrix measuring $|EDM_j| \times |EDM_i| \times a$ and populated with random values
17: $PM' = PM$ encrypted using MUOPE
18: **Participant** p_i: PM'_1 = differences between PM' and D'_i
19: **Participant** p_j: PM'_2 = differences between D'_j and PM'
20: $PoolEDM = PM'_1$ and PM'_2 combined to form a pooled EDM
21: $startRow = \sum_{n=1}^{n=(j-1)} |EDM_n|$
22: $startCol = \sum_{n=1}^{n=(i-1)} |EDM_n|$
23: **for** $gRow = 1$ to $gRow = |EDM_j|$ **do**
24: **for** $gCol = 1$ to $gCol = |EDM_i|$ **do**
25: $v' = 0$
26: **for** $att = 1$ to $att = a$ **do**
27: $v' = v' + |PoolEDM_{gRow,gCol,att}|$
28: **end for**
29: $ge_{startRow+gRow,startCol+gCol} = v'$
30: **end for**
31: **end for**
32: **end for**
33: **end for**
34: **Exit** with **GE**
35: **end procedure**

encrypted using MUOPE to give ϵ'. Thus the TPDM does not have access to the raw (plaintext) radius value.

The algorithm commences by creating an empty set C, and initialising the number of clusters so far, k, to 0 (line 2). For each encrypted record r'_i in $GData'$, which has not been previously assigned to a cluster (is "*unclustered*"), the *RegionQuery* sub-procedure is called to determine the ϵ-neighbourhood set S of the record (lines 4 and 5). The set S comprises the set of records in $GData'$ whose distance from r'_i is less than or equal to ϵ'. To this end **GE** is used to provide secure data comparison (line 33 in the *RegionQuery* sub-procedure). In lines 6 to 9, if the number of records in S is greater than or equal to $MPts$, record r'_i is marked as "*clustered*", the number of clusters so far is incremented by 1 and a new cluster C_k is created. The cluster C_k is then expanded by considering

the set of records in S using the *ExpandCluster* sub-procedure (line 10). The inputs to the *ExpandCluster* are: (i) the cluster C_k so far, (ii) the list S, (iii) the **GE**, (iv) the density parameters and (v) $GData'$. The *ExpandCluster* procedure is a recursive procedure, that adds the records in set S to cluster C_k if a record is "*unclustered*" (lines 19 to 21). The ϵ-neighbourhood for each record added to C_k is retrieved by another call to the *RegionQuery* procedure and returned in list S_2 (line 22). If the size of S_2 is greater than or equal to $MPts$ the *ExpandCluster* procedure is called again; and so on. The *ExpandCluster* procedure exits with expanded cluster C (line 28). The expanded cluster is then added to the cluster set in line 11. The S-DBSCAN procedure continues in this way until all the records in $GData'$ have been processed. The algorithm exits with the cluster configuration C (line 15).

6.4.2 Secure NNC

The pseudo code for S-NNC, founded on the use of a GEDM and the Cryptographic Ensemble (comprised of Liu's FHE scheme and MUOPE) is given in Algorithm 14. As in the case of the S-DBSCAN algorithm, S-NNC is entirely conducted by a TPDM following a process similar to standard NNC and to the DBSNNC process presented in Chapter 5; the distinction is that a GEDM is used instead of an EDM to determine similarity (Equation 6.1). The inputs are: (i) a global dataset $GData'$ encrypted using Liu's FHE scheme and collated by the TPDM, (ii) a GEDM **GE** and (iii) a NNC threshold σ' agreed by participating parties and encrypted using the proposed MUOPE scheme. The reader might find it useful to refer back to Sub-section 5.4.3 where the DBSNNC algorithm founded on the EDM idea was presented.

6.5 Experimental Results and Evaluation

This section presents the results obtained from the experimental analysis conducted to evaluate the GEDM concept, the proposed MUOPE scheme and the proposed secure data clustering algorithms. The evaluation objectives, in more detail, were as follows:

1. **Data owners participation:** To determine the complexity (in terms of runtime) of the data preparation process.
2. **Clustering efficiency:** To analyse the runtime required by a TPDM to cluster a dataset using either S-DBSCAN and S-NNC founded on the use of GEDMs.
3. **Clustering accuracy:** To compare the "correctness" of the clustering configurations produced using S-DBSCAN and S-NNC with the configurations produced using the "standard" equivalent algorithms.
4. **Clustering security:** To investigate the security of the proposed mechanisms in terms of potential attacks and revealed information.
5. **Clustering scalability:** To determine the effect on efficiency when the number of participating parties was increased.

The conducted evaluation considered two different kinds of datasets, synthetic datasets and UCI machine learning repository datasets as also used with respect to the UDM and EUDM/EDM evaluation presented in Chapters 4 and 5. The S-DBSCAN and S-NNC algorithms were implemented using the Java programming language and all experiments were run using an iMac (3.8 GHz Intel Core i5) running under the macOS High Sierra operating system with 8GB of RAM. The results obtained are discussed, in the context of the above objectives, in the following five sub-sections; Sub-sections 6.5.1 to 6.5.5.

Algorithm 13 Secure DBSCAN (S-DBSCAN) clustering algorithm

procedure S-DBSCAN($GData'$,**GE**,$MPts$,ϵ')
  $C = \emptyset$, $k = 0$
  for $i = 1$ *to* $i = |GData'|$ **do**
    if r'_i *is unclustered* **then**
      S = RegionQuery(r'_i,ϵ',**GE**,$GData'$)
      if $|S| \geqslant MPts$ **then**
        mark r'_i as *clustered*
        $k = k + 1$
        $C_k = r'_i$ (new cluster)
        C_k = ExpandCluster(C_k,S,**GE**,ϵ',$MPts$,$GData'$)
        $C = C \cup C_k$
      end if
    end if
  end for
  Exit with **C**
end procedure
procedure EXPANDCLUSTER(C,S,**GE**,ϵ',$MPts$,$GData'$)
  for $\forall\ r'_i \in S$ **do**
    if r'_i *is unclustered* **then**
      mark r'_i as *clustered*
      $C = C \cup r'_i$
      S_2 = RegionQuery($r'_i, \epsilon', \mathbf{GE}, GData'$)
      if $|S_2| \geqslant MPts$ **then**
        C = ExpandCluster(C,S_2,**GE**,ϵ',$MPts$,$GData'$)
      end if
    end if
  end for
  Exit with **C**
end procedure
procedure REGIONQUERY(r'_{index},ϵ',**GE**,$GData'$)
  $N_\epsilon = \emptyset$
  for $\forall\ r'_j \in GData'$ **do**
    if $sim(\mathbf{GE}, r'_{index}, r'_j) \leqslant \epsilon'$ **then**
      $N_\epsilon.add(r'_j)$
    end if
  end for
  Exit with N_ϵ
end procedure

6.5.1 Data Owner Participation

Data owners participation was measured in terms of the runtime required for: (i) data encryption (*Data Enc.*), (ii) DM calculation (*DM Cal.*), (iii) DM encryption (*DM Enc.*) and (iv) calculation of the data density required to dimension the MUOPE cyphertext space (*Dens. Cal.*). The amount of time required to generate Liu's FHE scheme key (*FHE Key Gen.*) did not entail any significant overhead on behalf of the data owner. Experiments reported in Chapter 3 demonstrated that $0.85ms$,$0.87ms$ and $0.89ms$ were

Algorithm 14 Secure Nearest Neighbour Clustering (S-NNC)

```
1: procedure S-NNC(GData', GE, σ')
2:     C = Empty set of clusters
3:     C_1 = {r'_1}
4:     C = C ∪ C_1
5:     k = 1
6:     for i = 2 to i = |GData'| do
7:         Find the r'_m in some cluster C_x where the sim(GE, r'_i, r'_m) is the smallest
8:         if sim(GE, r'_i, r'_m) ⩽ σ' and r'_m is founded then
9:             C_x = C_x ∪ r'_i
10:            Update the cluster set C_x in the set of cluster C
11:        else
12:            k = k + 1
13:            C_k = {r'_i}
14:            C = C ∪ C_k
15:        end if
16:    end for
17:    Exit with C
18: end procedure
```

required for Liu's FHE scheme key generation when the number of sub-cyphertexts m was set to 3, 9 and 15 respectively.

The evaluation was conducted using a sequence of ten synthetic datasets, ranging in size from $1,000$ to $10,000$ records, increasing in steps of $1,000$. The number of attributes was kept constant at 125. UCI datasets were also used, but not reported on here because the runtimes for *Data Enc.*, *DM Cal.* and *DM Enc.* were reported on previously in Table 5.4 in Chapter 5 where it was also noted that the recorded runtimes were negligible.

The recorded runtimes using the synthetic datasets for *Data Enc.*, *DM Cal.*, *DM Enc.* and *Dens. Cal.* are given in Figure 6.2. As expected, the recorded runtimes increased in a linear manner as the number of records (n) increased. The reported time for *Data Enc.*, *DM Cal.*, *DM Enc.* and *Dens. Cal.* for $n = 1,000$ were $0.02Sec$, $0.27Sec$, $0.74Sec$ and $0.44Sec$, respectively. The time required when the dataset size increases to $n = 10,000$ were $0.36Sec$, $62.6Sec$, $385.7Sec$ and $564.9Sec$. Whatever the case, as found using the experiments conducted previously using the UCI datasets, regardless of the number of records considered, the amount of data owner participation was not found to be significant in that it did not introduce any perceivable overhead on behalf of the participating parties. Once the data owners have encrypted their data, and jointly created their GEDM, no further data owner participation was required; the entire clustering process being delegated to the TPDM.

6.5.2 Clustering Efficiency

The runtimes required for the TPDM to cluster the UCI datasets using standard (unencrypted) DBSCAN/NNC and (encrypted) S-DBSCAN/S-NNC were compared to evaluate any potential overhead that might be incurred from using the proposed MUOPE scheme coupled with the GEDM concept. In the reported experiments, the DBSCAN parameters, *MPts* and ϵ, were as given in columns 2 and 3 of Table 6.1 and the NNC σ parameter was as given in column 2 of Table 6.2. The algorithm's parameters were randomly selected from a sequence of experiments (not reported here) conducted using

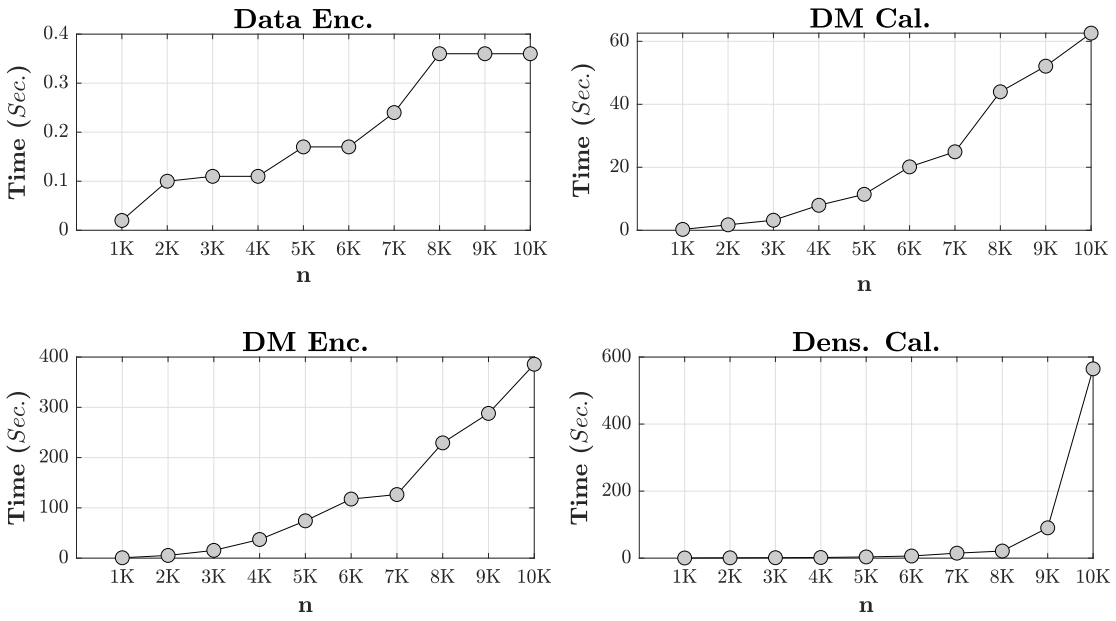


FIGURE 6.2: Time required (*Sec.*) for data owner participation in terms of number of records (n) in a data owner's local dataset

different values for the parameters, although in practice these values would be selected by the data owner. As expected, the runtime for S-DBSCAN and S-NNC were greater than when using the standard algorithms. However, inspection of the tables indicate that this did not present a significant overhead. For example, the Banknote Auth. dataset, the largest UCI dataset, in terms of number of records, required $44.2ms$ using DBSCAN compared to $264.9ms$ using S-DBSCAN; whilst standard NNC require $1630.22ms$ compared to $2228.84ms$ using S-NNC.

6.5.3 Clustering Accuracy

The reported *Sil. Coef.* and number of classes obtained using DBSCAN, and S-DBSCAN, and NNC and S-NNC, are given Tables 6.1 and 6.2, respectively. From the tables, it can be seen that the clustering configuration produced using S-DBSCAN and S-NNC were identical to those generated by the equivalent standard algorithms (using the same parameters). Recall that the main focus of the work presented in this thesis was not to produce an optimal clustering solution but to produce a secure clustering solution that was comparable with the clustering solutions produced using "standard" algorithms. Therefore, the values of algorithm parameters were selected without applying any methods that might help to achieve an optimal or better clustering configuration. Whatever the case, the obtained results provided clear evidence of the benefits that can be gained using the idea of a Cryptographic Ensemble and the GEDM concept to preserve data privacy without any loss of accuracy of the data clustering results.

6.5.4 Clustering Security

The TPDM is considered to be a "passive adversary" who follows the semi-honest model where the proposed S-DBSCAN and S-NNC algorithms given in Algorithm 13 and Algorithm 14 are honestly executed. As already noted in Chapter 2 (Sub-section 2.5.1) this was considered to be a reasonable assumption since the main objective of the TPDM (the CSP who provides the DMaaS) is to deliver a high quality service, that is both

TABLE 6.1: Algorithm parameters used in the evaluation, cluster configuration comparison and execution time in (ms) using DBSCAN and S-DBSCAN

No. UCI Dataset	MPts	ϵ	Standard DBSCAN			S-DBSCAN		
			Num of Clus.	Sil. Coef.	Exc. Time (ms)	Num of Clus.	Sil. Coef.	Exc. Time (ms)
1. Arrhythmia	2	600	6	0.472	14.4	6	0.472	51.3
2. Banknote Auth.	2	3	7	0.922	44.2	7	0.922	264.9
3. Blood Trans.	2	10	27	0.971	28.5	27	0.971	65.6
4. Brest Cancer	2	5	4	0.678	30.8	4	0.678	44.8
5. Breast Tissue	2	100	3	0.628	2.7	3	0.628	5.5
6. Chronic Kidney	2	70	19	0.970	12.1	19	0.970	48.4
7. Dermatology	2	10	16	0.853	12.4	16	0.853	24.7
8. Ecoli	2	60	1	-1.000	33.0	1	-1.000	43.0
9. Indian Liv. Pat.	3	40	7	0.789	12.0	7	0.789	56.6
10. Iris	5	2	2	0.722	5.5	2	0.722	13.7
11. Libras Mov.	5	5	11	0.715	12.0	11	0.715	39.5
12. Lung Cancer	2	20	1	0.053	1.8	1	0.053	2.4
13. Parkinsons	3	10	5	0.829	4.4	5	0.829	14.8
14. Pima Disease	5	20	4	0.691	10.9	4	0.691	74.6
15. Seeds	5	1	7	0.852	5.1	7	0.852	13.7

TABLE 6.2: Algorithm parameters used in the evaluation, cluster configuration comparison and execution time in (ms) using NNC and S-NNC

No. UCI Dataset	σ	Standard NNC			S-NNC		
		Num of Clus	Sil. Coef.	Exc. Time (ms)	Num of Clus	Sil. Coef.	Exc. Time (ms)
1. Arrhythmia	1	452	1.000	2736.91	452	1.000	2127.34
2. Banknote Auth.	5	21	0.895	1630.22	21	0.895	2228.84
3. Blood Trans.	68	34	0.999	342.04	34	0.999	388.84
4. Brest Cancer	10	108	0.903	253.08	108	0.903	282.19
5. Breast Tissue	1	105	1.000	2.4	105	1.000	3.83
6. Chronic Kidney	100	243	0.999	144.69	243	0.999	187.32
7. Dermatology	18	32	0.919	146.2	32	0.919	149.14
8. Ecoli	1	2	0.353	32.76	2	0.353	46.68
9. Indian Liv. Pat.	99	100	0.997	399.59	100	0.997	446.59
10. Iris	1	15	0.922	4.6	15	0.922	7.51
11. Libras Mov.	4	224	0.969	359.22	224	0.969	355.14
12. Lung Cancer	1	32	1.000	0.99	32	1.000	0.93
13. Parkinsons	73	11	0.953	18.16	11	0.953	22.9
14. Pima Disease	100	22	0.956	693.01	22	0.956	880.18
15. Seeds	1	103	0.979	9.12	103	0.979	14.88

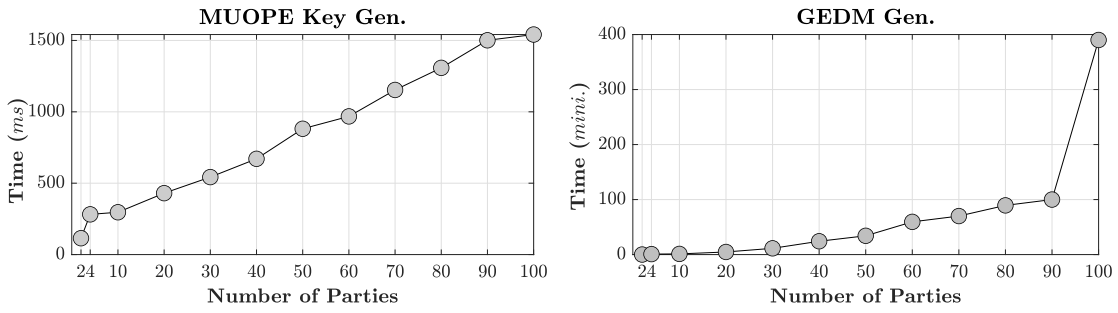
efficient and effective, to clients (data owners). In the proposed solution, the data and GEDM were encrypted and no decryption takes place at the TPDM side. Therefore, the security of the proposed solution relies on the security of the adopted Cryptographic Ensemble; thus, Liu's FHE scheme and the proposed MUOPE scheme. The security and threats associated with encrypting the dataset using Liu's FHE scheme were discussed in Chapter 4. The reader can refer back to Sub-section 4.4.4 where the security of Liu's FHE scheme and potential attacks that may be directed at it were discussed. The security of the data clustering, the MUOPE scheme and GEDM concept are discussed further in this sub-section.

Potential attacks that can be directed at the proposed solution are: (i) Cyphertext Only Attacks (COAs) when the adversary somehow has access to the encrypted GEDM and (ii) Overlapping Attacks (OAs) when the TPDM compares the distances with algorithm thresholds (as discussed in Sub-section 2.5.2). In terms of COAs, the GEDM could be used to extract statistical measures describing the frequency of distribution patterns which could be used to identify frequently occurring distributions which in turn could be used to identify the nature of plaintexts; but only if examples were available. As a countermeasure, the MUOPE (as in the case of FDH-OPE) uses "message space splitting" and "non-linear cyphertext space expansion" to hide the data distribution for data belonging to multiple data owners, and a "one-to-many" encryption function to hide data frequency; which makes inferences using COAs unlikely. OAs are precluded by encrypting the raw data used in the clustering, the DBSCAN radius parameter and the intermediate distances calculated by the TPDM using GEDM, which means that the TPDM compares the "order" of distance values and not the original distance values.

6.5.5 Clustering Scalability

The scalability of the proposed collaborative clustering was measured by considering: (i) the effect on runtime as the number of participants increased and (ii) the required memory resource as the size of the dataset considered increased. In the proposed approach collaboration between data owners occurs when: (i) generating the MUOPE encryption key (the *MUOPE Key Gen.* process) and (ii) deriving the Pooled EDMs to arrive at a GEDM (the *GEDM Gen.* process). A sequence of experiments was conducted whereby the number of participating data owners was increased from 10 to 100 in steps of 10; for completeness, the experiment also considered two and four data owners. Synthetic datasets were used, comprised of $10,000$ data records and 125 attributes equally distributed among the participants. The results are presented in Figure 6.3. From the figure, it can be seen that the *MUOPE Key Gen.* time was negligible, even for 100 participants; only $1541.39ms$ was required. As expected, the time required for *GEDM Gen.* increased with the number of participants. The GEDM can be generated in $0.05mins$ given only two participants, and in $390mins$ for 100 participants. The reason was that the increase in the number of participants (data owners) increased the number of EDM pairs to be considered, and consequently the number of Pooled EDMs that needed to be generated.

The GEDM, as already noted, is a 2D matrix where the two dimensions are correlated with the number of records in a given global dataset. Therefore, usage of a GEDM introduces a significant memory resources that might limit the adoption of the proposed solution. Figure 6.4 shows the number of elements for a range of GEDMs where the associated datasets feature different numbers of records and attributes. Recall, due to the similarity around the leading diagonal only the upper or lower triangle of a GEDM is required. As demonstrated by the Figure 6.4, regardless of the number of attributes, the



(a) (b)

FIGURE 6.3: Runtime to generate MUOPE keys and the GEDM as the number of participating parties (data owners) increases

number of elements in a GEDM is exponentially associated with the number of records in the given global dataset.

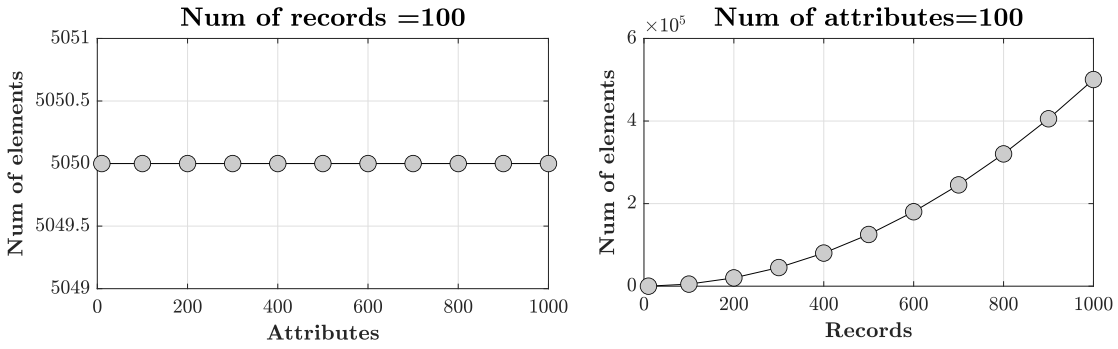


FIGURE 6.4: Number of elements in a GEDM for different sizes of data

6.6 Summary

This chapter has presented secure collaborative data clustering, in the context of the multiple data owners scenario, using the concept of a GEDM and a collaborative Cryptographic Ensemble. The proposed approach extended the EDM concept presented earlier and used a variation of the Cryptographic Ensemble presented in Chapter 5. The idea was that multiple data owners processed their dataset and generated local EDMs from which a GEDM was generated with reference to a STP. A particular advantage of the process was that it featured limited data owner participation. The process of combining EDMs to form a GEDM was conducted using what was referred to as the *pooling method*. The way that EDMs are "pooled" is dependent on the nature of data partitioning; in this chapter, only horizontal data partitioning was considered. The FDH-OPE scheme in the Cryptographic Ensemble, as used in connection with the single data owner scenario, was replaced with a novel multiple data owners scenario encryption scheme called the MUOPE scheme. The MUOPE scheme allowed a number of parties to encrypt their EDMs in such a way that the ordering of the data records was preserved across the parties. The proposed approach, unlike existing solutions, did not require data owner participation when the clustering was undertaken by a TPDM and did not entail any

loss of accuracy. The application of GEDM and the Cryptographic Ensemble was illustrated using secure variations of the established DBSCAN and NNC data clustering algorithms.

The proposed approach was extensively evaluated using UCI datasets and a sequence of synthetic datasets. The objectives of the evaluation were to consider the contribution of this chapter in terms of: (i) the amount of data owner participation, (ii) the efficiency of data clustering, (iii) the effectiveness (accuracy) of the data clustering, (iv) security in terms of potential attacks and (v) scalability. The results demonstrated that there was no overhead introduced on behalf of data owners when the GEDM idea was adopted. The clustering configurations obtained matched the results obtained using standard algorithms in terms of recorded *Sil. Coef.* values and number of clusters. The reported evaluation also demonstrated the scalability of the proposed approach. However, the memory resource required by GEDMs was substantial. In the next chapter, the idea of a Secure Chain Distance Matrices (SCDMs) is presented motivated by the memory resource requirement associated with GEDMs.

Chapter 7

Secure Chain Distance Matrices

7.1 Introduction

In the preceding chapters, Chapters 4, 5 and 6, the concept of the secure distance matrix that reduced the data owner participation was presented together with the idea of Cryptographic Ensembles. A number of variations of the distance matrix concept (UDMs, EUDMs, EDMs and GEDMs), and the Cryptographic Ensemble idea (FDH-OPE and MUOPE), were presented and illustrated in the context of secure single data owner and multiple data owner (collaborative) clustering. The key objective was to delegate most of the clustering process to a TPDM; previous approaches had involved substantial data owner involvement. In the proposed approaches, subsequent to data encryption, data owner participation was, in the worst case, limited to updating distance matrices, and in the best cases involved no data owner participation at all, depending on the nature of the clustering algorithm adopted (experiments were conducted using k-Means, NNC and DBSCAN).

However, the issue of concern with respect to the distance matrix approaches described so far was the extensive memory resources required. UDMs and EUDMs measured $n \times n \times a$, and EDMs and GEDMs $n \times n$. Where encryption was used, the storage requirement was compounded by the fact that cyphertext values require more space than the equivalent plaintext values. The issue of concern here was that the storage requirements would serve to limit the adoption of the proposed approaches with respect to large datasets. Recall that one of the research objectives central to the work presented in this thesis (see Chapter 1) was to provide a generic "versatile" solution that can be used to cater for a range of data mining algorithms especially data clustering and classification algorithms. To address this issue the concept of the Secure Chain Distance Matrix (SCDM) is proposed in this chapter, which can be used as an alternative to the previously considered distance matrices. As such SCDMs have the potential to provide for a wider range of secure third party data mining algorithms, in the context of both the single data owner scenario and the multiple data owner scenario, while at the same time significantly reducing the required memory resource.

This chapter introduces the SCDM concept and demonstrates its usage in the context of the single data owner scenario. The following chapter considers SCDM usage in the context of the multiple data owner scenario. Therefore, in this chapter the operation of the SCDM concept is compared with UDMs, EUDMs and EDMs; comparison of GEDMs is left to the following chapter.

The rest of this chapter is structured as follows. Section 7.2 provides a detailed description of the proposed SCDM concept. The section considers the process of generating SCDMs and updating them as required when clustering data using k-Means style

algorithms. Section 7.3 presents the required data preparation process, conducted by the data owner, ready for data outsourcing. This is followed by consideration of the application of SCDMs in the context of data clustering in Section 7.4 and data classification in Section 7.6. Each section presents one or more proposed secure data mining algorithms. The proposed algorithms were evaluated in Sections 7.5 and 7.7. Finally, Section 7.8 provides a summary and concludes the chapter.

7.2 Secure Chain Distance Matrices

This section presents the concept of Secure Chain Distance Matrices (SCDMs) designed to provide for third party data mining using Cryptographic Ensembles. The section comprises two sub-sections. The first, Sub-section 7.2.1, is concerned with the nature of SCDMs and how they can be used to derive (calculate) the similarity between data records and, where applicable, cluster centroids. The second, Sub-section 7.2.2, is concerned with the process of effectively and efficiently updating SCDMs whenever required as in the case of the Sk-Means algorithm presented later in Sub-section 7.4.1.

7.2.1 Secure Chain Distance Matrices

A SCDM is a 2D matrix that holds the *encrypted* distances (differences) between every attribute in every xth record in D, except the last record, and the following $x+1$th record in D, according to whatever ordering is featured in D. Given a dataset D that contains n records, $D = \{r_1, \ldots, r_n\}$, where each record r_x features a set of values corresponding to a attributes $r_x = \{r_{x,1}, \ldots, r_{x,a}\}$. A SCDM, **SC**, will comprises a set of elements $sc_{x,y}$ where each element holds the distance between the value for attribute y in record x and the value for attribute y in record $x+1$; $r_{x,y} - r_{x+1,y}$. A SCDM, **SC**, thus has the form shown in Equation 7.1.

$$\mathbf{SC} = \begin{bmatrix} sc_{1,1} & sc_{1,2} & \cdots & sc_{1,a} \\ \vdots & \vdots & \ddots & \vdots \\ sc_{n-1,1} & sc_{n-1,2} & \cdots & sc_{n-1,a} \end{bmatrix} \tag{7.1}$$

Given the above a SCDM associated with D will measured $(n-1) \times a$. In comparison with the storage requirement for a UDM or an EUDM, $n^2 \times a$, this is a reduction of $(n^2 - n + 1) \times a$. In comparison with an EDM where the storage requirement is n^2, this is a reduction of $(n^2 - na + a)$. In either case a substantial reduction.

SCDMs are generated by data owners in two steps: (i) Chain Distance Matrix (CDM) calculation and (ii) CDM encryption to arrive at a Secure CDM (SCDM). Algorithm 15 gives the pseudo code for the *CDMCalculation* process. The inputs are the raw dataset to be outsourced, *RawD*, and the number of attributes a that feature in *RawD*. The first step is for the data owner to process the raw data, where necessary, by casting categorical attributes values to an appropriate numeric form to arrive at a processed dataset D (line 2). The number of records (n) in the processed dataset is calculated in line 3 to be used for dimensioning the CDM. The CDM is dimensioned as $n-1 \times a$ (line 4). The CDM is then populated (lines 5 to 9); as already noted, the element $cdm_{x,y}$ is calculated as the distance between the yth attribute value in the xth record with the value for the same attribute in the following record (the $x+1$th record). The output of the *CDMCalculation* procedure is a CDM (line 10).

Although the CDM reduces the memory requirement with respect to UDMs, EUDMs and EDMs it still essentially comprises a set of linear equations that may support reverse

Algorithm 15 Chain Distance Matrix (CDM) calculation

```
1: procedure CDMCalculation(RawD, a)
2:     D = Dataset RawD converted to numeric dataset where necessary
3:     n = |D|
4:     CDM = 2D matrix measuring n − 1 × a
5:     for x = 1 to x = n − 1 do
6:         for y = 1 to y = a do
7:             cdm_{x,y} = r_{x,y} − r_{x+1,y} (cdm_{x,y} ∈ CDM)
8:         end for
9:     end for
10:    Exit with CDM
11: end procedure
```

engineering. Therefore, the second step is to encrypt each element in the CDM. With respect to the single data owner scenarios, on which this chapter is focused, the FDH-OPE scheme proposed earlier (Algorithm 6 presented in Section 5.2 of Chapter 5) was the most appropriate scheme for this; the MUOPE scheme entails overheads to facilitate multiple data owner collaboration that is not required in the single data owners scenario. The key feature of an encrypted CDM, a SCDM, is that a TPDM now has access to the distance value ordering, not the original distance values, between data records. The small set of elements in a SCDM, **SC**, allows a TPDM to derive/calculate the similarity between any two data records r_x and r_y (where $x < y$) in D as per equation 7.2.

$$sim(\mathbf{SC}, r_x, r_y) = \sum_{j=1}^{j=a} \left| \sum_{i=x}^{i=(y-1)} sc_{i,j} \right| \tag{7.2}$$

A SCDM can not only be used to determine the data similarity between data records, but can also be used to compare calculated distances against each other and with a particular threshold value. This means that the SCDM concept can be used to cluster encrypted data or to develop a classification model using the encrypted data records. In this section, the usage of the SCDM concept is illustrated using k-Means data clustering; however, as will become clear, SCDMs can be used with respect to other data mining algorithms. At the first k-Means iteration, the TPDM will use the first k records in the encrypted dataset (D'), encrypted using Liu's encryption (part of the proposed Cryptographic Ensemble), as the k-Means cluster centroids, $Cent'_1 = \{cent'_{1_1}, \ldots, cent'_{1_k}\}$. The remaining records in the encrypted dataset are then added to the nearest cluster by comparing the distances between data records and each cluster centroids. To this end the content of the generated SCDM, **SC**, will be used as per equation 7.2, but such that x represents the centroid identifier and y the data record identifier, hence the value of x is $1 \leqslant x \leqslant k$. The encrypted data records that have been added to clusters in the first iteration are then used to calculate the next iteration cluster centroids. The FHE mathematical properties of Liu's scheme were used to this end as presented in Chapter 3 to arrive at the new cluster centroids; $Cent'_2 = \{cent'_{2_1}, \ldots, cent'_{2_k}\}$. On the next iteration the TPDM re-determines the cluster contents using the distances between data records in D' and the set of new cluster centroids $Cent'_2$. However, the content of **SC** cannot be used as the new clustering centroid $Cent'_2$ are not added to the **SC**. The straightforward solution would be to send **SC** and $Cent'_2$ to the data owner for an update. However, this will create a computational and communication overhead on behalf of the data owner. The proposed solution, as in the case of UDMs, is to use the idea of a shift

matrix to update **SC** so that it can be used to determine data similarities with a new set of centroids. The SCDM updating process is presented in the following sub-section. The reader should be reminded that the need to update SCDMs is more to do with the operation of k-Means clustering than any deficiency with respect to the SCDM idea; if an alternative form of clustering were adopted, such as NNC or DBSCAN, there would be no need to update SCDMs.

7.2.2 SCDM Updating Process

The updating of a SCDM is not required in all cases. In the case of the secure k-Means clustering used to illustrate the utility of SCDMs in this section this is a requirement. As already noted, if an alternative form of secure data clustering were adopted, such as NNC or DBSCAN, there would be no need to update SCDMs. However, secure k-Means clustering is considered here because it allows illustration of the full potential of SCDMs. The same process can be used to allow for similarity determination between the outsourced encrypted data records and any other record, for example in the context classification querying processing as discussed in Chapter 2 and discussed in further detail in Section 7.6 below.

Whatever the case, the updating process is managed by the TPDM with very limited data owner participation. The pseudo code for the updating process is given in Algorithm 16. The inputs are: (i) the SCDM to be updated **SC**, (ii) the previous iteration clusters centroids $Cent'_i$, (iii) the new set of clusters centroids $Cent'_{i+1}$, (iv) the iteration number i, (v) the encrypted dataset D' and (vi) the number of attributes featured in dataset a. The first step in the process is for the TPDM to declare and dimension a 2D Chain Matrix **CH** (line 2). The first dimension of **CH** is associated with the number of centroids in $Cent'_i$ and $Cent'_{i+1}$, and the second dimension is associated with the number of attributes a. The ideas is that **CH** can be added (or "bound") to **SC** so that the chain feature can be extended to allow similarity determination over the new cluster centroids. In line 3 a Shift Matrix, S', is generated by the TPDM using the differences between the centroids from the previous iteration and the current cluster centroids. Recall that the cluster centroids are encrypted using Liu's FHE scheme. Therefore, the differences are calculated using the FHE mathematical property of Liu's FHE scheme ($\ominus$). The **CH** matrix is then populated by the TPDM (lines 4 to 11). The **CH** will hold distances (or difference) between two consecutive centroids in $Cent'_{i+1}$. The last element of **CH** will hold the distance between the last centroid $cent'_{i+1_k}$ and the first record in D'; the record r'_1. The form of **CH** is thus as shown in the following equation:

$$\mathbf{CH} = \begin{bmatrix} ch_{1,1} & ch_{1,2} & \cdots & ch_{1,a} \\ \vdots & \vdots & \ddots & \vdots \\ ch_{k,1} & ch_{k,2} & \cdots & ch_{k,a} \end{bmatrix}$$

$$= \begin{bmatrix} (cent'_{i+1_{1,1}} \ominus cent'_{i+1_{2,1}}) & (cent'_{i+1_{1,2}} \ominus cent'_{i+1_{2,2}}) & \cdots & (cent'_{i+1_{1,a}} \ominus cent'_{i+1_{2,a}}) \\ \vdots & \vdots & \ddots & \vdots \\ (cent'_{i+1_{k,1}} \ominus r'_{1,1}) & (cent'_{i+1_{k,2}} \ominus r'_{1,2}) & \cdots & (cent'_{i+1_{k,a}} \ominus r'_{1,a}) \end{bmatrix}$$

Returning to the pseudo code given in Algorithm 16 we now have an encrypted chain matrix, **CH**, encrypted using Liu's FHE scheme. For this to be bound to a SCDM it must be encrypted using the proposed FDH-OPE scheme because the SCDM, **SC**, is encrypted using FDH-OPE. The TPDM does not have the encryption keys to either, these are with the data owner. Thus data owner involvement is required. In line 12 of

the pseudo code **CH** is sent to the data owner and then re-encrypted using the FDH-OPE scheme to give $\mathbf{CH}'$ which is returned to the TPDM. This is achieved by calling the *ReEncryptCH* sub-procedure given in lines 20 to 30. The TPDM can then use $\mathbf{CH}'$ to update **SC**. On the first iteration (the first time the **SC** is updated), $\mathbf{CH}'$ will be concatenated to the matrix **SC** (lines 13 and 14 and also shown in Equation 7.3). For the remaining iterations, the first k elements in **SC** are modified by assigning the values of $\mathbf{CH}'$ to the first k elements of **SC** (line 16).

The algorithm exits with an updated **SC** and terminate in line 18. The k-Means algorithm continues in this manner until a stable cluster configuration is arrived at. This happens when the shift matrix S', calculated on line 3, is comprised of all zeros. Note that the shift matrix is encrypted using Liu's FHE scheme for which the data owner has the key, so the checking of the shift matrix for all zeroes also needs to be done by the data owner. If so the data owner sets the value for the *termination* variable to *true*, otherwise to *false*. This is done at the same time as $\mathbf{CH}'$ is generated so as to reduce the amount of data owner participation in the secure data mining.

$$\mathbf{SC} = \begin{bmatrix} ch'_{1,1} & ch'_{1,2} & \cdots & ch'_{1,a} \\ \vdots & \vdots & \ddots & \vdots \\ ch'_{k,1} & ch'_{k,2} & \cdots & ch'_{k,a} \\ sc_{1,1} & sc_{1,2} & \cdots & sc_{1,a} \\ \vdots & \vdots & \ddots & \vdots \\ sc_{n-1,1} & sc_{n-1,2} & \cdots & sc_{n-1,a} \end{bmatrix} \tag{7.3}$$

7.3 Data Outsourcing Process

This section discusses the data preparation process conducted by the data owner, on start-up, to prepare data and the SCDM for the TPDM. The data privacy is preserved by utilising the same Cryptographic Ensemble, comprised of Liu's FHE scheme and the proposed FDH-OPE scheme, as presented in Chapter 5. The initial step is for the data owner to process the raw data, $RawD$, by translating categorical attributes into a suitable format that allows distance calculation and data encryption. Similar to the approaches presented in the earlier chapters, casting is used to transfer categorical values to a discrete integer form. The next step is to encrypt the processed data (attribute-wise) to produce an encrypted dataset D'. The data is encrypted using Liu's FHE scheme following process given in Algorithm 1. Therefore, the data owner is also responsible for generating Liu's FHE encryption key $SK(m)$ as presented in Sub-section 3.3.1. Using the processed data, the CDM is then calculated by the data owner using algorithm 15 which is then encrypted using the FDH-OPE scheme (Section 5.2) to arrive at a Secure CDM (a SCDM). The outputs from the preparation process are thus the encrypted dataset D' and the $SCDM$ ready to be sent to the TPDM. Once the data is outsourced different data clustering and data classification mechanisms, that required distances comparisons, can be conducted on behalf of the data owner who instructs the TPDM by specifying the desired algorithm and the relevant parameters. The process for conducting secure k-Means clustering using the SCDM concept and the Cryptographic Ensemble idea have already been discussed earlier in this chapter. For evaluation purposes three different, single data owner scenario, data clustering mechanisms where considered: k-Means as introduced earlier, DBSCAN and NNC, and the k-NN data classification mechanism. The first three are considered in further detail in Section 7.4 and the last in Section 7.6.

Algorithm 16 SCDM updating process

procedure UPDATESCDM(**SC**, $Cent'_i$, $Cent'_{i+1}$, i, D', a)
  CH = 2D matrix measuring $|Cent'_i| \times a$
  $S' = Cent'_i \ominus Cent'_{i+1}$
  for $x = 1$ to $x = |Cent'_{i+1}| - 1$ **do**
    for $y = 1$ to $y = a$ **do**
      $ch_{x,y} = cent'_{i+1_{x,y}} \ominus cent'_{i+1_{x+1,y}}$ ($ch_{x,y} \in$ **CH**)
    end for
  end for
  for $y = 1$ to $y = a$ **do**
    $ch_{k,y} = cent'_{i+1_{k,y}} \ominus r'_{1,y}$ ($ch_{k,y} \in$ **CH** and $r'_{1,y} \in D'$)
  end for
  CH$'$, *terminate* = ReEncryptCH(**CH**, S')
  if $i == 1$ **then**
    SC = **CH**$' +$ **SC** ▷ (Equation 7.3)
  else
    SC = replace the first k elements of **SC** with **CH**$'$
  end if
  Exit with **SC** and *terminate*
end procedure
procedure REENCRYPTCH(**CH**, S')
  CHPlaintexts= Decrypt(**CH**) ▷ using Liu's decryption Algorithm 2
  CH$'$ = Encrypt(**CHPlaintexts**) ▷ using FDH-OPE Encryption Algorithm 6
  S = Decrypt(S') ▷ using Liu's decryption Algorithm 2
  if S==0 **then**
    terminate = true
  else
    terminate = false
  end if
  Exit with **CH**$'$ and *terminate*
end procedure

7.4 Secure Data Clustering Using SCDM

This section presents the three exemplar secure data clustering algorithms considered with respect to the evaluation reported on later in this chapter. Note that in each case the proposed secure data clustering is delegated to a TPDM. Two of the algorithms considered, Secure NNC and Secure DBSCAN, achieve the ideal solution of no further data owner participation once the clustering was commenced; whilst the Secure k-Means algorithm, as noted above, requires minimal data owner participation. Each is discussed in further detail in the following three sub-sections; Sub-sections 7.4.1 to 7.4.3.

7.4.1 Secure k-Means

The Secure k-Means (Sk-Means) algorithm, operated in a very similar manner as the standard k-Means algorithm [46]. The difference was that the mathematical operations in standard k-Means were replaced with the equivalent mathematical properties of Liu's FHE scheme as presented in Sub-section 3.3.1.2. The pseudo code for Sk-Means is given in Algorithm 17. The inputs are the encrypted dataset D', the SCDM **SC** and the number of desired clusters k. The algorithm commences by dimensioning the set

of cluster $C = \{c_1, \ldots, c_k\}$ and assigning the first k encrypted records in D' to the k clusters, one per cluster, to act as cluster centroids (lines 2 and 3). In line 4, the iteration counter, i, is initialised. A centroid set $Cent'_i = \{cent'_{i_1}, \ldots, cent'_{i_k}\}$ is then defined to hold the current centroids (lines 5 and 6). The remaining encrypted data records are then assigned to their appropriate clusters according to their similarity with respect to the cluster centroids in $Cent'_i$ using the *populateClusters* sub-procedure (called from line 7) and given at the end of the algorithm (lines 19 to 29).

The *populateClusters* sub-procedure inputs are: (i) the starting index (or identifier) for the records that remain in the encrypted dataset that still need to be allocated to appropriate clusters (rid), (ii) the set of cluster C, (iii) the set of current iteration cluster centroids $Cent'$, (iv) the SCDM (**SC**) and (v) the encrypted dataset (D'). The *populateClusters* sub-procedure uses **SC** to determine the similarity between encrypted data record r'_y and each cluster centroid in $Cent'$ (lines 21 to 27). Equation 7.2 is used to this end where the cluster centroids are always the first k elements in **SC** (line 23). Using the determined similarity values the record r'_y will be added to the nearest cluster (line 26). The sub-process continues until all remaining records have been processed. The result is then returned (line 28).

Returning to the main process, once a cluster configuration has been generated, a new set of encrypted cluster centroids, $Cent'_{i+1}$, is calculated as the mean of each attribute value for each record belonging to the same cluster (line 8). This calculation is achieved using the HE features of Liu's FHE scheme following the *CalculateCentroids* sub-procedure given in Algorithm 5 in Chapter 4. The content of **SC** is then updated in line 9 by calling the *UpdateSCDM* sub-procedure given in Algorithm 16. Recall that *UpdateSCDM* will return **SC** and the variable *terminate* set to true if the termination condition has been met. The content of **SC** then provides the information required for determining the similarity between the new set of cluster centroids $Cent'_{i+1}$ and the encrypted content of D'. Next, a loop is entered (lines 10 to 16) that is repeated until the termination condition, steady centroids, is arrived at, *terminate* true. The loop starts by creating a new empty set of cluster C (line 11) and all records will be assigned to appropriate cluster with respect to the new set of centroids $Cent'_{i+1}$ (line 12) by calling the *populateClusters* sub-procedure. The iteration counter i is then incremented (line 13). The process of calculating the centroids, updating SC and populating the cluster is continue until the termination condition is arrived at. The algorithm will exit, line 17, with the final cluster configuration C.

7.4.2 Secure NNC

The pseudo code for the proposed Secure NNC (SNNC) is presented in Algorithm 18. The SNNC algorithm operated in a similar manner to the standard NNC algorithm [48]. The main differences were that the dataset, D', and threshold value σ' were encrypted, and that the similarity between data records was determined using the SCDM concept. The algorithm's inputs were: (i) an encrypted dataset D', (ii) a SCDM **SC** and (iii) the value for σ encrypted using the FDH-OPE scheme to give σ'. The algorithm commences by creating an empty set of cluster C (line 2). The first cluster, C_1, is created and the first record in D' assigned to it (lines 3 and 4). The number of generated clusters k is initialised to 1 in line 5. A loop is then entered (lines 6 to 16) which repeats until all encrypted records in D' have been processed. The loop commences by finding a record r'_m in some cluster C_x where the distance to the current record r'_i is the smallest (line 7). To this end **SC** was used as per Equation 7.2. The calculated distance was then securely compared with σ' (line 8). In the case where the calculated distance is less

Algorithm 17 Secure k-Means (Sk-Means) clustering algorithm

1: **procedure** SK-MEANS(D', **SC**, k)
2: C = Set of k empty clusters
3: Assign the first k records in D' to C (one per cluster)
4: $i = 1$
5: $Cent'_i$ = Set of k cluster centroids
6: Assign first k records in D' to $Cent'_i$
7: C = PopulateClusters($k+1, C, Cent'_i$,**SC**,D')
8: $Cent'_{i+1}$ = CalculateCentroids(C,k) ▷ Algorithm 5
9: **SC**, *terminate* = UpdateSCDM(**SC**, $Cent'_i$, $Cent'_{i+1}$, 1, D', a) ▷ Algorithm 16
10: **while** ! *terminate* **do**
11: C = Set of k empty clusters
12: C = PopulateClusters($1, C, Cent'_{i+1}$,**SC**,D')
13: $i = i + 1$
14: $Cent'_{i+1}$ = CalculateCentroids(C,k) ▷ Algorithm 5
15: **SC**, *terminate* = UpdateSCDM(**SC**, $Cent'_i$, $Cent'_{i+1}$, i, D', a)
16: **end while**
17: **Exit** with C
18: **end procedure**
19: **procedure** POPULATECLUSTERS(rid,C,$Cent'$,**SC**,D')
20: $id = null$
21: **for** $y = rid$ to $y = |D'|$ **do**
22: **for** $x = 1$ to $x = |C|$ **do**
23: $sim = sim(\mathbf{SC}, cent'_x, r'_y)$ where $cent'_x \in Cent'$ and $r'_y \in D'$ ▷ (Equation 7.2)
24: id = cluster identifier with lowest sim value so far
25: **end for**
26: $c_{id} = c_{id} \cup r'_y$ ($c_{id} \in C$)
27: **end for**
28: **Exit** with C
29: **end procedure**

than or equal to σ' the currently record (r'_i) will be added to cluster C_x (lines 9 and 10). Otherwise r'_i is assigned to a new cluster (lines 12 to 14). The algorithm will continue in this manner until all data records have been assigned to a cluster. The algorithm will exit, line 17, with a cluster configuration C.

7.4.3 Secure DBSCAN

The pseudo code for Secure DBSCAN (SDBSCAN) is presented in Algorithm 19. The inputs are: (i) the encrypted dataset D'; (ii) the SCDM, **SC**, previously generated by the data owner; and (iii) the desired density parameters ($MPts, \epsilon'$). Similar to standard DBSCAN [47], density is defined as the minimum number of points, $MPts$, within a certain distance ϵ. The threshold ϵ is encrypted to allow secure comparison and also to hide the correlation (distance below or above the threshold) between data records when the TPDM executes SDBSCAN. The algorithm commences by creating an empty set of clusters C and initialising the number of clusters so far variable k to 0 (lines 2 and 3). For each record r'_i in D' that has not been previously assigned to a cluster, is "unclustered", the set S is determined (lines 5 and 6). The set S is the ϵ-neighbourhood

Algorithm 18 Secure NN Clustering (SNNC) algorithm

```
1: procedure SNNC(D', SC, σ')
2:     C = Empty set of clusters
3:     C_1 = {r'_1}
4:     C = C ∪ C_1
5:     k = 1
6:     for i = 2 to i = |D'| do
7:         Find r'_m in some cluster C_x where sim(SC, r'_i, r'_m) the smallest
8:         if sim(SC, r'_i, r'_m) ⩽ σ' and r'_m is founded then
9:             C_x = C_x ∪ r'_i
10:            Update Cluster set C_x in cluster set C
11:        else
12:            k = k + 1
13:            C_k = {r'_i}
14:            C = C ∪ C_k
15:        end if
16:    end for
17:    Exit with C
18: end procedure
```

of r'_i and comprises the set of records in D' whose distance from r'_i is less than or equal to ϵ'. The set is determined by calling the *RegionQuery* sub-procedure (line 6) that is given at the end of the algorithm (lines 31 to 40). In the *RegionQuery* sub-procedure **SC** is used to determine the overall distances between records (line 34). If the number of records in S is greater than or equal to $MPts$ the density requirement is satisfied and hence r'_i is marked as "clustered" and considered to represent a new cluster C_k (lines 7 to 10). The C_k cluster is then expanded by considering the records in S using the *ExpandCluster* sub-procedure called in line 11 and given in lines 18 to 30. The inputs to the *ExpandCluster* sub-procedure are: (i) the cluster C_k so far, (i) the set S, (ii) the SCDM **SC**, (iv) the DBSCAN density parameters ($MPts$ and ϵ') and (v) the encrypted dataset D'. The *ExpandCluster* sub-procedure is a recursive procedure. For each record in S which has not been previously clustered we add the record to C_k and then determine the ϵ-neighbourhood S_2 for this record (lines 19 to 23). If the size of S_2 is greater than or equal to $MPts$ we call the *ExpandCluster* sub-procedure again (lines 24 and 25) and so on until all the records in D' are processed, at which point the algorithm will exist in line 16 with the cluster configuration C.

7.5 Results and Evaluations

This section reports on the experimental analysis conducted to evaluate the utilisation of the SCDM concept, coupled with the Cryptographic Ensemble idea, with respect to secure data clustering. The evaluation was directed at two objectives:

1. To evaluate the application of SCDMs in a context of the secure data clustering processes present in Sub-sections 7.4.1, 7.4.2 and 7.4.3.

2. To compare the operation of SCDMs with that of EUDMs as presented in Chapter 5.

Algorithm 19 Secure DBSCAN (SDBSCAN) clustering algorithm

```
1: procedure SDBSCAN(D', SC, MPts, ε')
2:     C = ∅
3:     k = 0
4:     for i = 1 to i = |D'| do
5:         if r'_i is unclustered then
6:             S = RegionQuery(r'_i, ε', SC, D')
7:             if |S| ⩾ MPts then
8:                 mark r'_i as clustered
9:                 k = k + 1
10:                C_k = r'_i
11:                C_k = ExpandCluster(C_k, S, SC, MPts, ε', D')
12:                C = C ∪ C_k
13:            end if
14:        end if
15:    end for
16:    Exit with C
17: end procedure
18: procedure ExpandCluster(C,S,SC,MPts,ε', D')
19:    for ∀ r'_i ∈ S do
20:        if r'_i is unclustered then
21:            mark r'_i as clustered
22:            C = C ∪ r'_i
23:            S_2 = RegionQuery(r'_i, ε', SC, D')
24:            if |S_2| ⩾ MPts then
25:                C = ExpandCluster(C, S_2, SC, MPts, ε', D')
26:            end if
27:        end if
28:    end for
29:    Exit with C
30: end procedure
31: procedure RegionQuery(r'_index, ε', SC, D')
32:    N_ε = ∅
33:    for ∀ r'_j ∈ D' do
34:        distance = sim(SC, r'_index, r'_j)          ▷ (Equation 7.2)
35:        if distance ⩽ ε' then
36:            N_ε.add(r'_j)
37:        end if
38:    end for
39:    Exit with N_ε
40: end procedure
```

The first of the above objectives is considered in Sub-section 7.5.1 below, while the second in Sub-section 7.5.2.

7.5.1 SCDM Application Evaluation

To evaluate the application of SCDMs the operation of the three proposed secure clustering algorithms presented above were considered. The following criteria were used:

1. **Amount and complexity of data owner participation:** The computation overhead introduced on behalf of data owners when the proposed secure data clustering algorithms were adopted.
2. **Clustering efficiency:** The runtime complexity of the proposed approaches.
3. **Clustering accuracy:** The "correctness" (or effectiveness) of the resulting clustering configurations produced by proposed approaches compared to equivalent standard approaches.
4. **Security:** The potential for attacks.

For the evaluation the selected fifteen UCI datasets used previously were again used. Each of the above criteria is considered in further detail below.

Amount and complexity of data owner participation: Data owner participation was measured in terms of the runtime required for data preparation and the amount of data owner involvement whilst secure data clustering, undertaken by the TPDM, was in progress. Recall that data preparation comprises: (i) Cryptographic Ensemble key generation (*Key Gen*), (ii) data encryption (*Data Enc*), (iii) CDM calculation (*CDM Cal*) and (iv) CDM encryption (*CDM Enc*) to arrive at the SCDM. The *Key Gen* process is a one time process that does not add any overhead on behalf of the data owner. The evaluation results demonstrated that the average time required to generate the FDH-OPE encryption keys was 80.32ms, whilst the Liu's FHE scheme keys were generated in $0.85ms$. The results obtained with respect to *Data Enc*, *CDM Cal* and *CDM Enc*, using the UCI datasets, are given in columns 6, 7 and 8 of Table 7.1, respectively. From the table, it can be seen that negligible time was required for preparing the data. For example, the total time required for data preparation in the context of the largest dataset, Arrhythmia, was $149.22ms$ (reported as $9.80ms$ for data encryption, $3.33ms$ for CDM calculation and $136.09ms$ for CDM encryption). The degree of data owner involvement once data clustering was in progress (undertaken by the TPDM) depends on the nature of adopted algorithm. The SNNC and SDBSCAN approaches do not entail any data owner participation, whereas in the case of Sk-Means the participation was limited to the decryption and re-encryption of the Chain matrix, CH, on each iteration. The data owner involvement, compared to the entire data clustering runtime is shown in Figure 7.1. Formally, in the case Sk-Means, data owner participation is $O(k \times i \times a)$, where k is the number of desired clusters to be produced, i is the number of iterations and a is the number of attributes.

Clustering efficiency: Clustering efficiency was evaluated by comparing the runtime to cluster the datasets using the proposed secure clustering mechanisms (Sk-Means, SNNC and SDBSCAN) with the standard equivalent algorithms that operates on unencrypted data. The aim was to demonstrate that the overhead introduced when the clustering was conducted over encrypted data was not significant. The clustering runtime results are given in Figure 7.2. The results are the average of ten runs. For each dataset the reported time is the entire time for clustering the datasets and thus the time for data preparation by the data owners are excluded, whilst the involvement of the data owner when clustering is progress (in the case of Sk-Means) is included. From the figure, it can be seen that the overall runtimes required for the secure clustering algorithms, as expected, were longer than in the case of standard algorithms. The largest dataset (Arrhythmia), as anticipated, resulted in the longest runtime. From the figure it can also be seen that Sk-Means is the more efficient algorithm as only the distances between data records and cluster centroids were required, and not distances between each data record

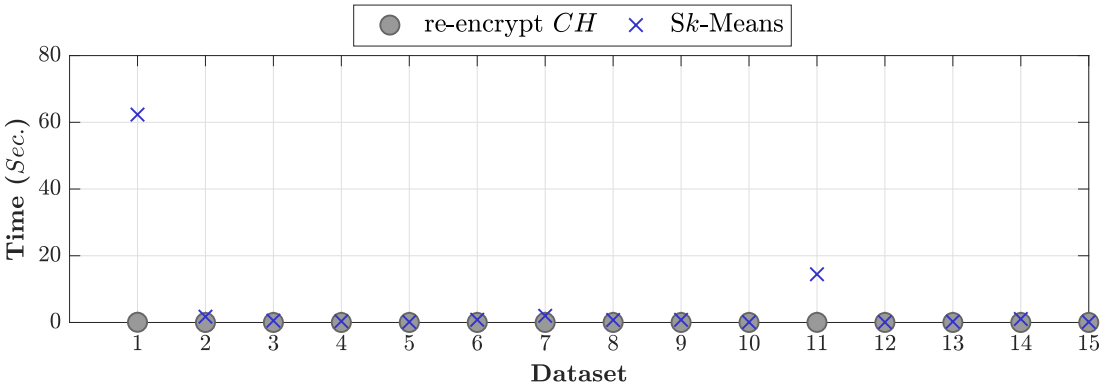


FIGURE 7.1: The complexity of the data owner participation to decrypt and re-encrypt the CH matrix against the entire runtime for Sk-Means

TABLE 7.1: Runtimes for data owner data preparation and algorithm operating statistics

No. UCI Dataset	σ	k	$MPts$	ϵ	Data Enc (ms)	CDM Cal (ms)	CDM Enc (ms)
1. Arrhythmia	1	16	2	600	9.80	3.33	136.09
2. Banknote Auth.	5	2	2	3	1.85	0.2	32.08
3. Blood Trans.	68	2	2	10	4.04	0.13	20.17
4. Brest Cancer	10	2	2	5	1.70	0.19	27.36
5. Breast Tissue	1	6	2	100	0.40	0.03	23.73
6. Chronic Kidney	100	2	2	70	2.65	0.28	37.38
7. Dermatology	18	6	2	10	1.80	0.43	31.77
8. Ecoli	1	8	2	60	0.98	0.09	31.54
9. Indian Liv. Pat.	99	2	3	40	1.30	0.15	39.39
10. Iris	1	3	5	2	0.20	0.04	17.87
11. Libras Mov.	4	15	5	5	4.80	1.26	92.07
12. Lung Cancer	1	3	2	20	0.59	0.05	13.34
13. Parkinsons	73	2	3	10	1.90	0.13	36.00
14. Pima Disease	100	2	5	20	1.44	0.18	37.18
15. Seeds	1	3	5	1	0.48	0.06	26.77

and each other record as in the case of SNNC and SDBSCAN. However, inspection of the recorded results indicates that this did not present a significant overhead. The results also show that the bigger the dataset the larger the SCDM, and consequently the greater the time required to interact with the SCDM to cluster the data; this was to be expected.

Clustering accuracy: The accuracy (the "correctness") of generated cluster configurations was measured by comparing the results obtained using the proposed secure algorithms with the standard (unencrypted) equivalents. The parameters (σ, k, $MPts$ and ϵ) used in each case are given in columns 2 to 5 of Table 7.1. The accuracy metric used was the Silhouette Coefficient (*Sil. Coef.*) [188] as described in Sub-section 4.4.3. Table 7.2 shows the results obtained. For reasons of reader convenience, the results obtained from the standard algorithms that were given previously in Tables 4.3, 5.5 and 6.2, are again given in columns 2, 3, 6, 7, 10 and 11 of Table 7.2. From the Table

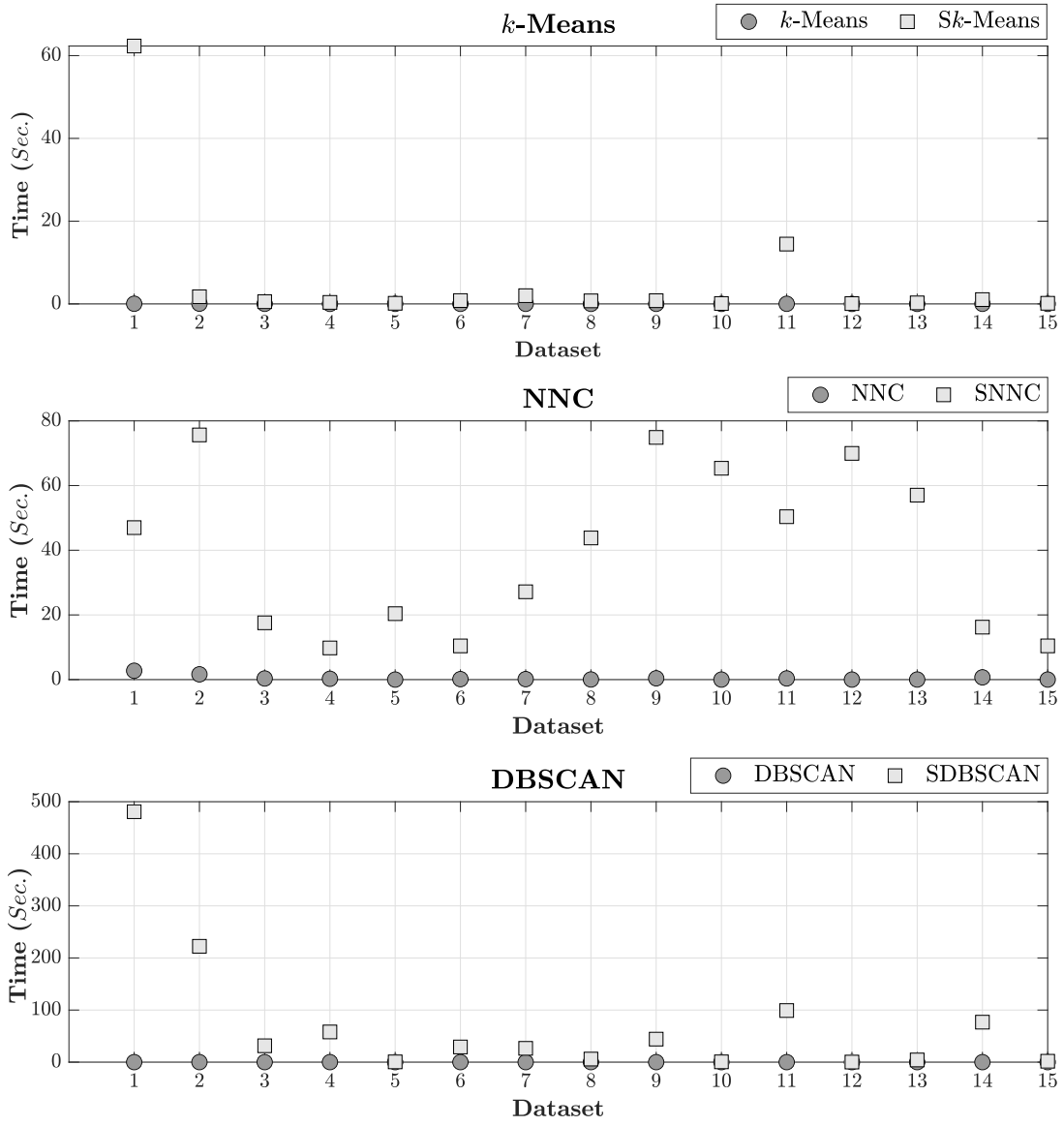


FIGURE 7.2: Comparison of the runtimes (*Sec.*) using standard and secure clustering algorithms (k-Means, NNC and DBSCAN)

7.2, it can be seen that the cluster configurations produced using the proposed secure algorithms were the same in 35 of the 45 cases (same number of clusters and same *Sil. Coef.*). Where the configurations were different, in one case the *Sil. Coef.* value was the same, in the remaining cases the *Sil. Coef.* value using the secure clustering was better in seven of the nine cases. The reason for the different configurations obtained was the nature of the FDH-OPE scheme which featured a degree of deliberately introduced randomness so that identical plaintexts values resulted in different cyphertext values, even when the same encryption key was used. Consequently, because differences between records was obtained using a "summing-up" process, this randomness was compounded when differences were calculated along the chain matrix, this was especially the case given a large degree of separation between records within the chain. It is interesting to note that this "randomness" sometimes produced better cluster configurations.

Security: The TPDM, as already noted, is considered to be a passive adversary who will

TABLE 7.2: Cluster configuration comparison using standard and secure algorithms (differing results highlighted in bold font)

No.	Standard DBSCAN		SDBSCAN		Standard NNC		SNNC		Standard k-Means		Sk-Means	
	Num. Clus.	Sil. Coef	Num. Clus.	Sil. Coef	Num. Clus.	Sil. Coef	Num. Clus.	Sil. Coef	Iter.	Sil. Coef	Iter.	Sil. Coef
1.	6	0.472	6	0.472	452	1.000	452	1.000	10	0.602	10	0.602
2.	7	0.922	7	0.922	21	0.895	21	0.895	16	0.207	16	0.207
3.	**27**	**0.971**	**33**	**0.976**	**34**	0.999	**35**	0.999	12	0.370	12	0.370
4.	**4**	**0.678**	**1**	**0.485**	**108**	**0.903**	**135**	**0.926**	8	0.020	8	0.020
5.	3	0.628	3	0.628	105	1.000	105	1.000	18	0.787	18	0.787
6.	19	0.970	19	0.970	243	0.999	243	0.999	8	0.009	8	0.009
7.	**16**	**0.853**	**15**	**0.881**	**32**	**0.919**	**37**	**0.915**	**15**	**0.801**	**7**	**0.822**
8.	1	-1.000	1	-1.000	2	0.353	2	0.353	**23**	**0.628**	**14**	**0.631**
9.	7	0.789	7	0.789	100	0.997	100	0.997	13	0.169	13	0.169
10.	2	0.722	2	0.722	**15**	**0.922**	**16**	**0.927**	14	0.836	14	0.836
11.	11	0.715	11	0.715	224	0.969	224	0.969	18	0.590	18	0.590
12.	1	0.053	1	0.053	32	1.000	32	1.000	**8**	**0.645**	**10**	**0.738**
13.	5	0.829	5	0.829	11	0.953	11	0.953	7	0.079	7	0.079
14.	4	0.691	4	0.691	22	0.956	22	0.956	8	0.000	8	0.000
15.	7	0.852	7	0.852	103	0.979	103	0.979	6	0.706	6	0.706

process a given data mining task as specified. However, it is assumed that the passive adversary will also try to learn additional information about the data by analysing intermediate results and messages exchanged during algorithm execution. Using the proposed clustering algorithms the dataset, the SCDM and any intermediate results were all encrypted using the Cryptographic Ensemble and no decryption took place at the TPDM side. Therefore, the security of the proposed clustering relies on the security of proposed Cryptographic Ensemble composed of Liu's FHE scheme, to encrypt the raw data, and the proposed FDH-OPE scheme, to encrypt the CDM to arrive at the SCDM. The security analysis is thus as reported in Chapter 5.

7.5.2 Comparison Between SCDMs and EUDMs

The second evaluation objective was to compare the operation of SCDMs to EUDMs in the context of their application to secure data clustering. The comparison criteria were: (i) complexity of data owner participation, (ii) clustering efficiency, (iii) clustering accuracy and (iv) scalability in terms of required memory resource. The outcomes from the comparative analysis related to the above criteria are presented below.

Complexity of data owner participation: The EUDM and SCDM were compared in terms of their data owner participation. This encompassed two main processes, data preparation and involvement once data clustering was in progress. The data preparation process with respect to the EUDM, as discussed in Sub-section 5.4.1, comprised: data encryption, UDM calculation and UDM encryption. The SCDM preparation comprised: data encryption, CDM calculation and CDM encryption as described previously in Section 7.3. The reported runtimes to prepare the UCI datasets using both the EUDM and SCDM approaches are given in Tables 5.4 and 7.1, respectively. The results demonstrate that negligible time was required, even for large datasets. It should be noted that Liu's FHE scheme, adopted with respect to the proposed approaches, encrypted plaintexts values to *three* sub-cyphertexts ($m = 3$); therefore the reported runtimes can be expected to increase when using larger values of m. The reader may wish to refer back to Sub-section 3.3.1.4 where the computational aspects of Liu's FHE scheme were discussed. The time complexity for calculating the UDM and encrypting it, as expected, was higher than that required for calculating the CDM and encrypt it, as the number of elements to be calculated is higher in the case of EUDMs. From the results, it was concluded that approaches founded on SCDMs are more efficient than approaches founded on EUDMs in terms of data preparation.

Data owner involvement whilst the TPDM was undertaking the data clustering, as noted earlier, is dependant on the nature of the clustering algorithm used. The NNC and DBSCAN approaches, using either EUDMs or SCDMs, did not entail any data owner participation, whist the k-Means approach requires data owner participation during the decryption and re-encrypting of the Shift Matrix or Chain Matrix of size $k \times a$. Figure 7.3 shows the recorded times required to decrypt and re-encrypt the matrices using the Secure k-Means approach coupled with either an EUDM or a SCDM, DBSk-Means and Sk-Means respectively. As expected, the time for data owner involvement was comparable as the matrices used were of the same size.

Clustering efficiency: With respect to the efficiency of secure data clustering founded on the use of SCDMs compared to EUDMs, this was determined in terms of *overall runtime* for clustering the encrypted data, both UCI datasets and synthetic datasets. The later to conduct a more in-depth analysis. The usage of the EUDM concept was illustrated using DBSk-Means and DBSNNC, as described in Chapter 5; whilst the usage of the SCDM concept was illustrated using Sk-Means, SNNC and SDBSCAN

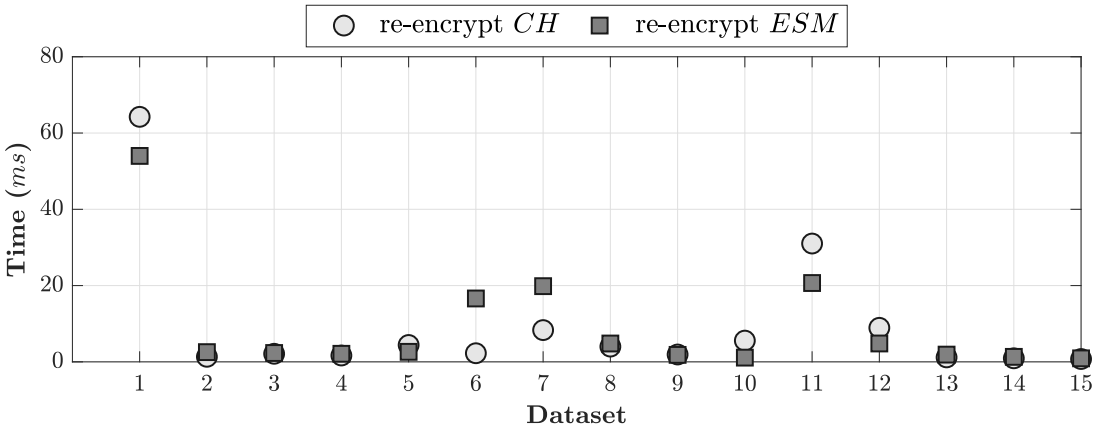


FIGURE 7.3: Runtime required by the data owner to decrypt and re-encrypt Encrypted Shift Matrices (ESMs) and Chain matrices (CH) during data clustering using Secure k-Means

as described in this chapter. Therefore the efficiency comparison was conducted with respect to DBSk-Means and Sk-Means, and DBSNNC and SNNC.

The individual runtime results obtained using the UCI datasets were presented previously in Tables 5.5, 5.6 and Figure 7.2. These results are brought together in Figure 7.4. From the figure, as expected, the recorded runtimes for data clustering using SCDMs is longer than that required using EUDMs. The reason for this is the "chain" feature used to derive the similarity using the SCDM. Using SCDMs the similarity between two records required the "summing-up" process that add of a number of SCDM records (as shown in Equation 7.2), whilst in the case of EUDMs the similarity can be simply determined by adding up a one record in EUDM as shown in Equation 5.6.

For the experiments using simulated datasets comprised of $10,000$ records, but with varying numbers of attributes from 10 to 100 increasing in steps of 10, was used. For the comparison of the operation EUDMs and SCDMs the time complexity of determining the similarity between the first and the last record in each dataset was considered. Figure 7.5 shows the obtained average runtimes for ten runs. The reported runtime for using the EUDM and SCDM for similarity determination increased in time with number of attributes featured in data records although in the case of SCDM was higher as the similarity is determined by "summing-up" a number of SCDM elements.

Clustering accuracy: The accuracy (effectiveness/correctness) of secure data clustering algorithms founded on EUDM and SCDM were compared by comparing the produced clustering configurations. The "correctness" of clustering configurations were then evaluated using the equivalent standard algorithm that operates over unencrypted dataset. As already demonstrated with respect to the evaluation of EUDMs reported on in Sub-section 5.6.3, the clustering configurations produced using EUDM based algorithms were identical to the clustering configuration produced using standard algorithms as evidenced by the Silhouette Coefficient *Sil. Coef.* (Tables 5.5 and 5.6). The clustering algorithms that utilise SCDMs produced comparable results, as opposed to identical results, to those produced in the case of standard algorithms (Table 7.2). The reason for these differences, as noted earlier, was caused by the chain feature of SCDMs coupled with the FDH-OPE scheme that produced different cyphertexts by adding noise to the generated cyphertexts. This noise was accumulated when determining similarity that in turn resulted in the production of different configurations to those produced using standard algorithms.

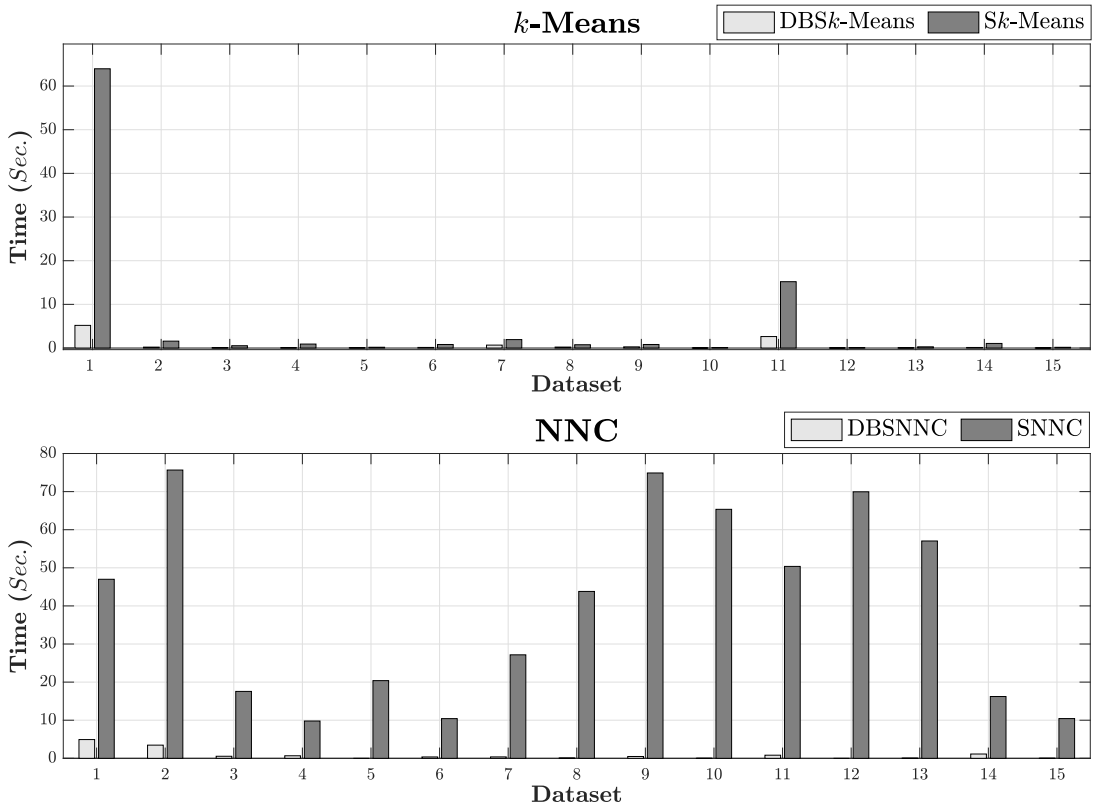


FIGURE 7.4: Efficiency of the Secure k-Means and Secure NNC algorithms founded on the EUDM and SCDM

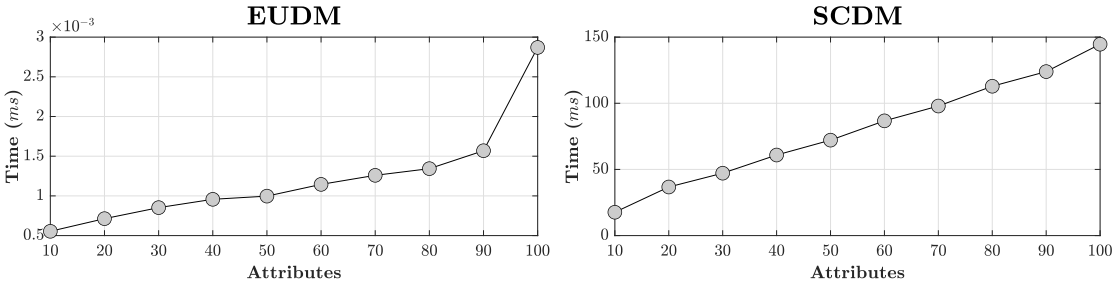


FIGURE 7.5: Complexity of comparing a single data record, using EUDMs and SCDMs, with datasets comprised of $10K$ records and features different numbers of attributes

Scalability: Recall that an EUDM is a 3D matrix where the first and second dimension are equivalent to the number of records n and the third dimension is the number of attributes a featured in the given dataset. Recall also that a SCDM is a 2D matrix where the first dimension is equivalent to $n-1$ and the second to a. The required memory resources are thus different according to the number of elements in both matrices. The number of elements in the EUDM and SCDM with respect to UCI datasets were shown in Table 7.3. The number of elements and thus the memory resources are more compact in the case of SCDMs compared to EUDMs. Experiments were also conducted to study the effect of increasing values of n and a, using synthetic datasets, on the size of EUDMs and SCDMs. The results are shown in Figure 7.6. From the figure it can clearly be seen that SCDMs are more memory efficient than EUDMs.

TABLE 7.3: Number of elements in EUDMs and SCDMs for different UCI datasets

No. UCI Dataset	Num. of UDM Elements	Num. of CDM Elements
1. Arrhythmia	28563462	125829
2. Banknote Auth.	3767512	5484
3. Blood Trans.	1120504	2988
4. Brest Cancer	650133	6501
5. Breast Tissue	51039	945
6. Chronic Kidney	1924800	9576
7. Dermatology	2283474	12410
8. Ecoli	452928	2680
9. Indian Liv. Pat.	1702360	5820
10. Iris	45300	596
11. Libras Mov.	5848200	32310
12. Lung Cancer	29568	1736
13. Parkinsons	420420	4268
14. Pima Disease	2362368	6136
15. Seeds	155085	1463

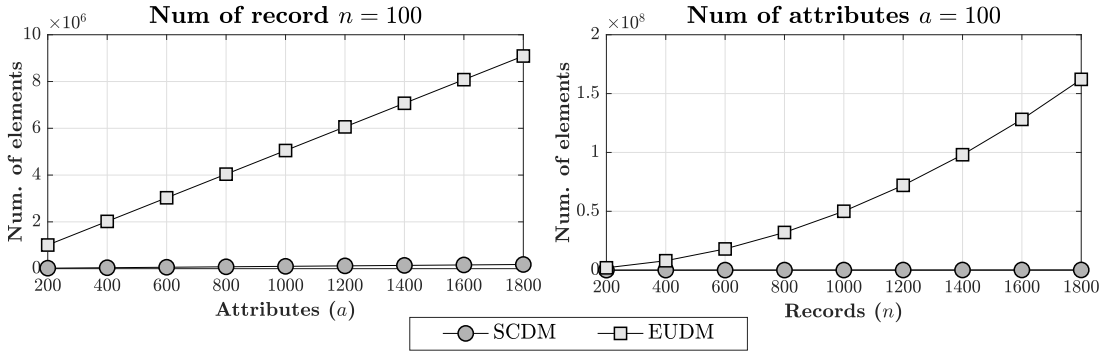


FIGURE 7.6: Number of elements in SCDMs and EUDMs for different sizes of synthetic datasets

7.6 Secure Data Classification Using SCDM

This section describes the application of the SCDM concept, coupled with the Cryptographic Ensemble idea, in the context of secure data classification. The basic concept is that a data owner outsources their prelabelled encrypted data to a TPDM who then uses this to create a classification model of some kind that can be used to label new encrypted records, without decryption and without involving the original data owner in the process. The new records to be labelled may be submitted by the original data owner whose data was used to create the classification model, however, the proposed system can also be used to provide a collaborative query service whereby authorised Query Owners (QOs) can label their private records (as described in Chapter 2). In this thesis, the term "querying" refers to the process of labelling/classifying a new record using a pre-constructed classification model held by a TPDM. The query record thus refers to the unlabelled data record sent by the QO to the TPDM.

The section is divided into two sub-sections. The proposed secure classification system model is presented in Sub-section 7.6.1 where the role of each participant and

the design goals of the system are considered. The system model can be used with respect to a number of established classification model generation algorithms. However, the k Nearest Neighbour (kNN) data classification mechanism [189] is considered here for illustrative and evaluation purposes, both because of its simplicity and because of its wide domain of application [190–193]. The implementation of Secure kNN (SkNN) classification, using the proposed model, is thus discussed in Sub-section 7.6.2.

7.6.1 Secure Classification System Model

The proposed secure classification system model is given in Figure 7.7. From the figure it can be seen that it features three types of participant: a data owner, a TPDM and one or more QOs. The TPDM is the CSP that provides storage and computation services such as DMaaS. The data owner is the party that has a prelabelled dataset D available for secure sharing, that consists of n records, $D = \{r_1, \ldots, r_n\}$. Each record r_i has $a + 1$ attribute values; $r_i = \{r_{i,1}, \ldots, r_{i,a}, r_{i,a+1}\}$ where $r_{i,a+1}$ is the class label for data record r_i. The QOs are authorised parties who wish to classify their data records $Q = \{q_1, q_2, \ldots\}$. Each query record q_i has a attribute values; $q_i = \{q_{i,1}, \ldots, q_{i,a}\}$. The dataset D is prepared, by the data owner, for outsourcing to the TPDM, as described previously in Section 7.3. The dataset is thus encrypted using Liu's FHE scheme and the associated SCDM using the proposed FDH-OPE scheme. However, for the classification to operate as intended, the class label (the $a + 1$th attribute) are not encrypted.

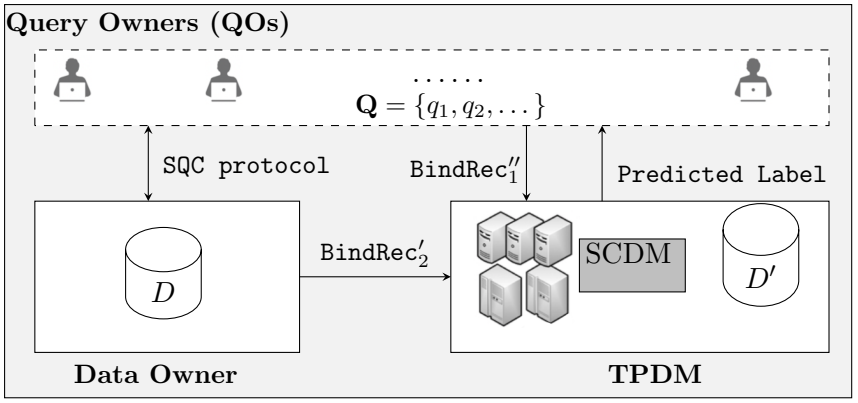


FIGURE 7.7: Secure classification system model

The primary design goal for the proposed system model is to preserve the privacy of both the data and query records. To maintain the privacy of any query $q_i \in Q$ (where Q is a set of queries) the query needs to be encrypted by the QO who owns the query, before it is submitted to the TPDM for processing. Clearly to allow q_i to be processed using the proposed approach q_i needs to be encrypted using the same encryption key as that used to encrypt the SkNN classification model held by the TPDM. The data owner holds the encryption key and clearly this should not be directly shared with the QOs. Using the proposed system query encryption is therefore achieved using a proposed Secure Query Cyphering (SQC) protocol that preserves the privacy of the query record and the confidentiality of data owner's private key. A second requirement, to enhance security, is that querying is controlled by the data owner. How both the query encryption and data owner control is achieved depends on the nature of the encrypted classification model. In the following section this is considered in further detail in the context of Secure kNN (SkNN) classification.

7.6.2 Secure k Nearest Neighbour

This section describes how the proposed secure classification system model, described in the foregoing sub-section can be used to realise Secure k Nearest Neighbour (SkNN) data classification. The proposed SkNN approach consists of three main steps:

1. **Query encryption:** The secure encryption of the QO's query record to preserve privacy, while maintaining data owner encryption key confidentiality. To this end a Secure Query Cyphering (SQC) protocol is used, described in further detail in Sub-section 7.6.2.1.

2. **QO authorisation and binding process:** The encrypted query q' is "bound" with the SCDM to allow the data similarity between the contents of D' and q' to be determined. Only the query belonging to an authorised QOs can be bound to the SCDM. The QO authorisation and binding process are detailed in Sub-section 7.6.2.2.

3. **SkNN query classification:** Query resolution (classification) conducted in two sub-steps: (i) nearest neighbour record retrieval and (ii) major class label determination. Both are discussed further in Sub-section 7.6.2.3.

7.6.2.1 Secure Query Cyphering Protocol

The Secure Query Cyphering (SQC) protocol operates between a data owner and one or more QOs and is designed to allow the QOs to encrypt a query record, $q_i = \{q_{i,1}, q_{i,2}, \ldots, q_{i,a}\}$, using FDH-OPE, so that a "binding" record can be generated which in turn is utilised by the TPDM to update its SCDM. The binding process and the updating of the SCDM is discussed in the following sub-section, this sub-section presents the SQC protocol. To encrypt the query record q_i, using the FDH-OPE scheme, QO requires the FDH-OPE key. As the FDH-OPE scheme, presented in Section 5.2, is a symmetric scheme sharing the key with the QO presents a security risk. The idea, instead of providing the FDH-OPE key, is therefore to provide the QO with the parameters to allow FDH-OPE encryption. However, provision of these parameters still presents a security threat. Therefore, the parameters are encrypted using Liu's FHE scheme whose mathematical features permit the FDH-OPE encryption of q_i without decryption of the parameters. As a result, q_i will be double encrypted, firstly using FDH-OPE to give q'_i, and secondly using Liu's FHE scheme to give q''_i. Equation 7.4 shows the sequence of double encryption. It should be noted here that the Liu FHE scheme keys used with respect to the SQC protocol are different to the Liu FHE scheme keys used to encrypt the dataset D. To distinguish between the two, the former will be referred to as the *Shared Liu's* scheme (shared because later in the SkNN process the keys will be shared with the TPDM).

$$q''_i = \texttt{Encrypt}_{Liu'sFHE}\big(\texttt{Encrypt}_{FDH-OPE}(q_i)\big) \tag{7.4}$$

The process for encrypting the value v using FDH-OPE was given in line 7 of Algorithm 6, but is given again in equation 7.5. As illustrated earlier in Chapter 5, l'_j is the minimum boundary for the cyphertext space interval in question, $scale_j$ is the required scaling between the cyphertext space interval and the corresponding message space interval, and δ_j is a random noise value included to prevent identical values being encrypted in the same way on repeated encryptions. The reader may find it useful to refer back to Section 5.2 where the FDH-OPE scheme was presented. Equation 7.5 can be

rewritten as per Equation 7.6 (with noise δ_j removed). Removing the δ_j from encryption function provides the potential to reduce the effect of accumulating error when determining the similarity between the query and encrypted data using the SCDM. This will contribute to maintaining the accuracy of query prediction services, however revealing the frequency of attribute values in the query set as no random values are incorporated to hide the frequency.

$$v' = l'_j + scale_j \times (|v| - l_j) + \delta_j \tag{7.5}$$

$$v' = scale_j \times |v| + l'_j - scale_j \times l_j \tag{7.6}$$

Equation in 7.6 can be further simplified to:

$$v' = scale_j \times |v| + e_j$$

where $e_j = l'_j - scale_j \times l_j$. The parameters $scale_j$ and e_j are calculated by the data owner, encrypted using the ***Shared Liu's*** FHE scheme to give $scale'_j$ and e'_j, and sent to the relevant QO. Of course, the values of $scale_j$ and e_j are dependent on the interval in which $|v|$ falls; thus this also needs to be established within the context of the SQC protocol. The SQC protocol to achieve the above can be summarised as follows:

Protocol: Secure Query Cyphering (SQC)

1: **Data owner** generates the ***Shared Liu's*** key and sends the key to the TPDM.
2: Using binary questioning with the **QO**, the **data owner** identifies the FDH-OPE interval ID(j) within which each query attribute value in q_i is contained.
3: **Data owner** calculates the FDH-OPE values for $scale_j$ and e_j for each attribute value in q_i.
4: **Data owner** encrypts the $scale_j$ and e_j values using the ***Shared Liu's*** FHE scheme to arrive at $scale'_j$ and e'_j.
5: **Data owner** sends $scale'_j$ and e'_j to **QO**.
6: Using $scale'_j$ and e'_j, **QO** double encrypts the query attribute values in q_i using the HE mathematical properties of ***Shared Liu's*** scheme ($\circledast$,$\oplus$) as equation: $q''_{i,att} = (q_{i,att} \circledast scale'_j) \oplus e'_j$

7.6.2.2 QO Authorisation and Binding Process

The binding process is the process whereby a query record is incorporated into the SCDM held by the TPDM. Recall that the SCDM contains distances (differences) between corresponding attribute values in each pair of consecutive records in the dataset encrypted using the FDH-OPE scheme. The binding process requires that the difference between the query record q held by QO and the first record in the dataset held by the data owner be added to the SCDM without sending either to the TPDM. The binding process is therefore a collaborative process conducted between the data owner and the QO, and is required not only to allow a response to QO's query, but also for the query to be authorised by the data owner.

The pseudo code for the binding process is given in Algorithm 21. The inputs are: (i) the SCDM **SC** and (ii) the query record encrypted using the SQC protocol q''_i. The process starts with the data owner generating a random record p of length a, $p = \{p_1, \ldots, p_a\}$ (line 2). This is then encrypted twice, firstly using the FDH-OPE scheme to give p', and secondly using the ***Shared Liu's*** scheme to give p'', which is then

Algorithm 21 Authorisation and binding process

1: **procedure** AUTHORISATIONBINDING($\mathbf{SC}$,q_i'')
2: **Data owner:** generate a random record p
3: **Data owner:** p'= Encrypt (p) ▷ using FDH-OPE
4: **Data owner:** p''= Encrypt (p') ▷ using *Shared Liu's*
5: **Data owner:** send p'' to relevant QO
6: **QO:** BindRec$_1''$= $q_i'' \ominus p''$
7: **QO:** send BindRec$_1''$ to TPDM
8: **Data owner:** BindRec$_2$= $p - r_1$
9: **Data owner:** BindRec$_2'$= Encrypt(BindRec$_2$) ▷ using FDH-OPE
10: **Data owner:** send BindRec$_2'$ to TPDM
11: **TPDM:** BindRec$_1'$= Decrypt(BindRec$_1''$) ▷ using *Shared Liu's*
12: **TPDM:** *Pivot*= BindRec$_1'$ + BindRec$_2'$
13: **TPDM:** $\mathbf{SC}_{updated} = Pivot + \mathbf{SC}$ ▷ Equation 7.7
14: **Exit** with $\mathbf{SC}_{updated}$
15: **end procedure**

sent to the relevant QO (lines 3 to 5). The double encryption is required because, to retain the confidentiality of the FDH-OPE key held by the data owner, q_i is also double encrypted as discussed in Sub-section 7.6.2.1 above. The QO will then generate a binding record $BindRec_1''$ representing the difference between their double encrypted query record q_i'' and the p'' (line 6). This is achieved using the *Shared Liu's* FHE mathematical property $\ominus$ as described in Sub-section 3.3.1.2. The binding record $BindRec_1''$ is then sent (line 7) to the TPDM (see Figure 7.7). At the same time the data owner will calculate the binding record $BindRec_2$ (line 8), representing the distances between p and the first record in the dataset D. The binding record, $BindRec_2$ is encrypted (line 9) using FDH-OPE to give $BindRec_2'$, which is then sent to the TPDM (line 10). Receipt of $BindRec_2'$ signals to the TPDM that the data owner has given "approval" for the query, without this the TPDM will not process the query. The role of the data owner and QO is now finished.

Once the TPDM has received $BindRec_1''$ and $BindRec_2'$, the TPDM single decrypts the double encrypted $BindRec_1''$, to give $BindRec_1'$ (line 11). Both binding records are now encrypted using FDH-OPE. The TPDM then creates a *Pivot* record by adding $BindRec_1'$ to $BindRec_2'$ (line 12). The *Pivot* record will now hold the encrypted distance between the query record q_i and the first record in $r_1 \in D$ without either being confided to the TPDM, or each other. The pivot record is then added to the SCDM $\mathbf{SC}$ in line 13 as indicated by Equation 7.7. The algorithm will exit in line 14 with the updated $\mathbf{SC}$, $\mathbf{SC}_{updated}$. The similarity between the query record q (at index 1 in $\mathbf{SC}_{updated}$) and the xth encrypted record in the dataset is calculated using Equation 7.8.

$$\mathbf{SC}_{updated} = \begin{bmatrix} pivot_{1,1} & pivot_{1,2} & \cdots & pivot_{1,a} \\ sc_{1,1} & sc_{1,2} & \cdots & sc_{1,a} \\ \vdots & \vdots & \ddots & \vdots \\ sc_{n-1,1} & sc_{n-1,2} & \cdots & sc_{n-1,a} \end{bmatrix} \tag{7.7}$$

$$sim(\mathbf{SC}_{updated}, q, r_x') = \sum_{j=1}^{j=a} \left| \sum_{i=1}^{i=x} \mathbf{sc}_{updated\ i,j} \right| \tag{7.8}$$

7.6.2.3 SkNN Query Classification

The processing (classification) of queries from an authorised QO (note that the data owner may also be a QO) is entirely delegated to the TPDM following a process similar to standard kNN [189]. The main reasons for using a TPDM for query classification are:

1. The limited computing resource and technical expertise that data owners are anticipated to have. The assumption is that the data owner's core business is not data analytics, but some other form of commerce whereby data is generated which the data owner is prepared to share for commercial gain.
2. The expectation that data owners and QOs are likely to want to avail themselves of the analytical capabilities offered using a mobile device of some kind.

Using a TPDM for query resolution also provides the additional benefit that query outcomes (query predicted label) for queries belonging to a QO are not shared with the data owner (unless of course the data owner and the QO are the same individual).

Algorithm 22 shows the pseudo code for SkNN data classification. The inputs are: (i) the SCDM ($\mathbf{SC}_{updated}$) on completion of the authorisation and binding process (Algorithm 21) whereby the distance between the query record and the first record in D has been inserted at index 1 ($sc_{updated\ 1}$) of the SCDM, (ii) the encrypted dataset D' and (iii) the desired value for k as in the original kNN. The SkNN process comprises two stages: (i) secure Nearest Neighbour (NN) retrieval (lines 2 to 6) and (ii) determination of the major class label (line 7 which calls a procedure given in lines 10 to 18). The first stage starts with the calculation of the similarity between query record q' and each other record $r'_j \in D'$ as per Equation 7.8 (line 4). The calculated distance ($dist'$), together with the associated class label held at $r'_{j,a+1}$, is added to the neighbour list NL (line 5). The second stage, determining the major class label, is commenced by ordering the neighbour list according to the $dist'$ values (line 11). Recall that the FDH-OPE scheme used to encrypt the SCDM is an order preserving encryption scheme, thus facilitating secure data ordering. The first k elements in the neighbour list are then used to create a list C of length l that represents the number of class labels in D'. The list C holds counts of the number of records in the first k elements in NL that correspond to each class label featured in the first k elements in NL. Recall that, the class label attributes are not encrypted as discussed in Sub-section 7.6.1. The maximum class label (the class label with the highest count) is returned as the predicted label for the query (line 8).

7.7 Results and Evaluations

The evaluation of the proposed SkNN data classification approach, including the binding process and the SQC protocol, is presented in this section. The evaluation was conducted using the same UCI and synthetic datasets as used for earlier evaluations. The objectives were to consider the proposed solution in terms of: (i) computation and communication costs on behalf of the data owner, (ii) computation and communication costs on behalf of QOs, (iii) performance of SkNN in terms of runtime, (iv) performance of SkNN in terms of classification accuracy and (v) the security of the proposed approach. Each will be discussed in detail in the following sub-sections.

7.7.1 Data Owner Computation Cost Analysis

Using the proposed SkNN approach the data owner will participate in: (i) preparing data for the TPDM, (ii) running the SQC protocol and (iii) authorising QO queries.

Algorithm 22 Secure kNN (SkNN) classification algorithm

procedure SkNN($\mathbf{SC}_{updated}$,D',k)
  $NL = \emptyset$
  for $j = 1$ to $j = |D'|$ **do**
    $dist' = \text{sim}(\mathbf{SC}_{updated}, q', r'_j)$ where q' is the current query ▷ Equation 7.8
    $NL = NL \cup \{dist', r'_{j,a+1}\}$
  end for
  $predictedClass = \text{majorClassLabel}(NL, k)$
  Exit with $predictedClass$
end procedure
procedure MAJORCLASSLABEL(NL, k)
  Order NL using $dist'$
  $C = \{c_1, \ldots, c_l\}$
  for $i = 1$ to $i = k$ **do**
    $LabelCurrentRec =$ label of nl_i
    $c_{LabelCurrentRec} = c_{LabelCurrentRec} + 1$
  end for
  Exit with Max(C)
end procedure

The SkNN approach requires no data owner involvement during the processing of QO queries once authorisation has taken place. The data preparation was the same as in the case of the SCDM data clustering approach, and encompasses: (i) the generation of secret keys, (ii) data encryption, (iii) CDM calculation and (iv) CDM encryption to produce a SCDM. The evaluation of the complexity of data owner participation was presented in Sub-section 7.5.1. The SQC protocol requires data owner participation in determining and encrypting the scale ($scale$) and e values required by the `Shared Liu's` scheme so as to allow QOs to encrypt their queries. The runtimes for calculating $scale$ and encrypting e for some interval was recorded as $0.16ms$ and $0.11ms$ respectively, which means that no significant computational overhead is encountered by the data owner. The data owner also participates in authorising QO queries by generating and then encrypting the binding record $BindRec'_2$. Table 7.4 (row 4) shows the recorded runtimes (ms) for different sizes of the $BindRec'_2$ record. The runtime results presented in the table are the average of ten runs. As anticipated, the average runtime required to generate the $BindRec'_2$ increases with the dimension of the query record; for example a query record with $1K$ attributes will be generated in $2.4ms$, while $16.4ms$ is required to generate a $BindRec'_2$ with $10K$ attributes. These results have shown that the process of authorising QO queries does not introduce any significant overhead.

7.7.2 Query Owner Computation Cost Analysis

The QO will participate in: (i) the SQC protocol to encrypt their query records and (ii) compute the binding record, $BindRec''_1$. Rows 2 and 3 of Table 7.4 give the average runtime, over ten runs, required to encrypt a range of query records of increasing length (number of attributes) and the time required by a QO to calculate a binding record $BindRec''_1$. The required runtime to encrypt a query record increases with the number of attributes featured in the query. A query with $1,000$ attributes can be encrypted in $4.4ms$ whilst a query with $10,000$ attributes can be encrypted in $18.9ms$. The required runtime to generate $BindRec''_1$ increases with the number of attributes. The QO binding

TABLE 7.4: Average runtimes (ms) for data owner and QO participation when generating binding records and running the SQC protocol with respect to different numbers of attributes (a)

Process	Number of attribute values (a)									
	1K	2K	3K	4K	5K	6K	7K	8K	9K	10K
Encrypt query record q''	4.4	6.1	10.8	11.3	11.6	13.4	14.2	15.5	17.8	18.9
Calculate and encrypt the $BindRec''_1$	2.3	5.0	6.3	7.0	8.8	9.3	9.4	11.4	11.6	13.8
Calculate and encrypt the $BindRec'_2$	2.4	4.2	9.5	7.0	8.9	9.9	12.0	13.9	15.6	16.4

record for a query with $1,000$ attributes can be generated in $2.3ms$ while $13.8ms$ was recorded for a query with $10,000$ attributes. These results shown that regardless of query dimension, the runtime associated with QO participation was not significant and therefore did not introduce any overhead of note.

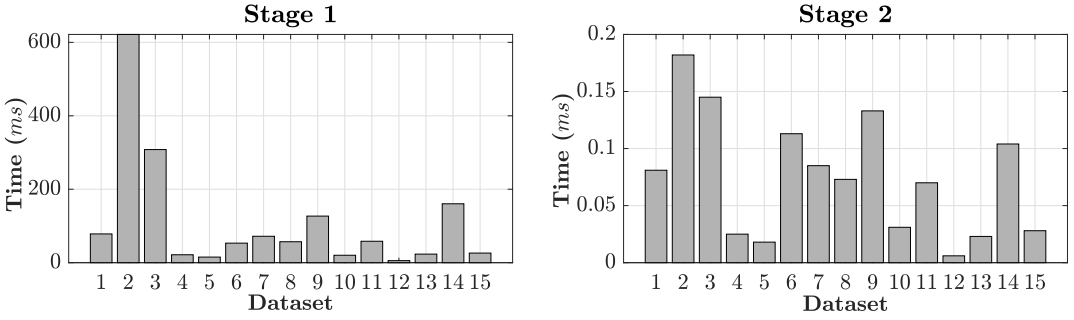
7.7.3 SkNN Efficiency

The runtime required to classify data using the proposed SkNN approach was compared with the runtime required for the standard kNN algorithm operating over unencrypted data. Figure 7.8 shows the average recorded runtimes required to classify the UCI datasets for the two stages of the kNN algorithm: secure NN retrieval (Stage 1) and determination of the major class label (Stage 2). The x-axis gives the evaluation dataset number from Table 7.5. The reported runtimes were measured in terms of average runtimes obtained using Ten-fold Cross Validation (TCV). As expected, the overall time required for SkNN Stage 1 was longer than in the case of the standard approach. Note that runtimes for (standard) kNN Stage 1 are reported in milli-second (ms), while runtimes for SkNN Stage 1 are reported in second (Sec). The results shows that the bigger the dataset the larger the SCDM, and consequently the greater the time required to interact with the SCDM to classify a record. However, inspection of the recorded results indicates that this did not present a significant overhead. The largest dataset, in terms of number of records and number of attributes, is the *Arrhythmia* dataset for which the recorded runtime was $78.29ms$ for standard kNN and $55.13Sec$ for SkNN. The Stage 2 runtimes were almost the same since the major class was determined over unencrypted class labels in both cases.

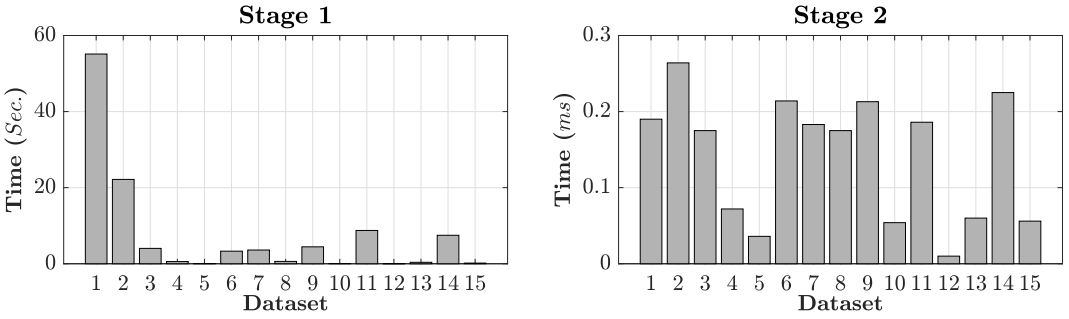
The effect of the size of a query record, measured in terms of a (number of attribute values) and the selected value for k was also evaluated. A range of values for a was considered from $1,000$ to $10,000$ increasing in steps of $1,000$, coupled with $k = 1$, $k = 5$ and $k = 9$. The required classification runtime in each case is plotted in Figure 7.9. As expected, the runtime increases as the size of the query record increases, whilst the value of k does not introduce any significant overhead.

7.7.4 SkNN Accuracy

The classification accuracy obtained using the proposed SkNN approach was compared with the accuracy obtained using standard kNN. The aim was to evidence that SkNN



(a) Standard kNN



(b) SkNN

FIGURE 7.8: Comparison of runtimes using standard kNN and SkNN classification

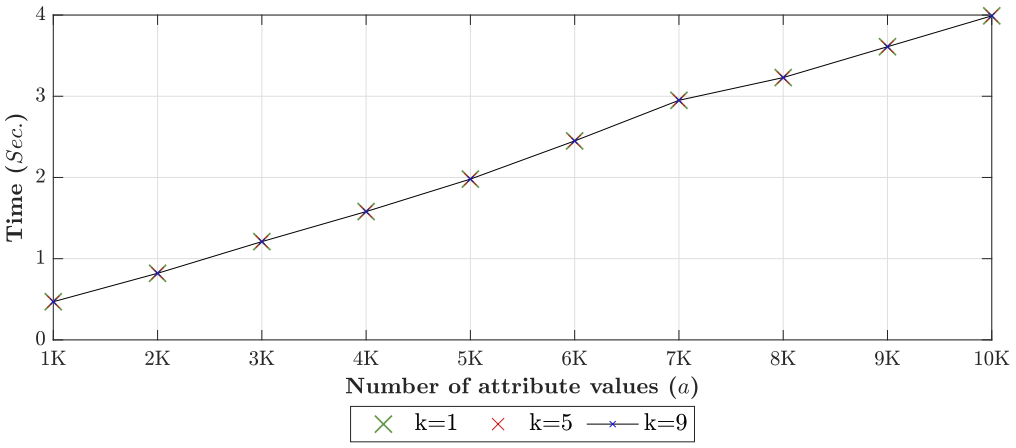


FIGURE 7.9: Average computation costs of SkNN for varying values of k and number of attributes in a query record

operated correctly; for this to be the case the accuracy values obtained should be comparable. The UCI evaluation datasets were split into training (the outsourced dataset D) and testing (the query set Q). Average Precision, Recall and the F1 measure [194] were used as the evaluation metrics, obtained using TCV. So as to conduct a fair comparison the same value for k was used in all cases. The values of Precision (**P**), Recall (**R**) and the **F1** measure were calculated as per Equation 7.9 where TP is the *True Positive* count, the number of times the model correctly predicts the positive class; and TN is the *True Negative* count, the number of times the model correctly predicts the negative class. FP and FN are thus the *False Positive* and *False Negative* counts when the model incorrectly predicts the positive and negative classes, respectively. Note that high precision relates to a low false positive rate, and high recall relates to a low false

negative rate. High scores for precision and recall show that the predictor is performing well. The F1 measure combines both precision and recall and is thus the most significant measure.

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$
$$F1 = \frac{2 \times Precision \times Recal}{Precision + Recal} \tag{7.9}$$

The obtained results are presented in Table 7.5. From the table, it can be seen that from the fifteen datasets considered, in six cases the results obtained were different (highlighted in bold font); interestingly in five of the cases SkNN produced a better performance. In the remaining case the performance was not as good (lower F1 value recorded in the context of Arrhythmia). The difference, it was again conjectured, was because the FDH-OPE scheme does not support equality matching in that two identical plaintext values will have different encrypted equivalents because of the δ random noise added. The usage of SCDMs in determining the data similarity results in an accumulation of the added noise value (δ). Sometimes this operated in favour of SkNN by preventing overfitting. The overall average Precision, Recall and F1 values were 0.71, 0.72 and 0.71 for standard kNN and 0.72, 0.73 and 0.72 for SkNN, indicating that both approaches produced similar results and therefore it was concluded that the proposed SkNN approach operated correctly.

TABLE 7.5: Comparison of prediction accuracies using standard kNN and SkNN (differing results highlighted in bold font)

No. UCI Dataset	Standard kNN			SkNN		
	P	R	F1	P	R	F1
1. Arrhythmia	0.25	0.22	**0.24**	0.25	0.22	**0.23**
2. Banknote Auth.	1.00	1.00	1.00	1.00	1.00	1.00
3. Blood Trans.	**0.60**	**0.59**	**0.60**	**0.61**	**0.61**	**0.61**
4. Breast Cancer	0.64	0.63	0.63	0.64	0.63	0.63
5. Breast Tissue	0.57	0.57	0.57	0.57	0.57	0.57
6. Chronic Kidney	0.82	**0.84**	0.82	0.82	**0.85**	0.82
7. Dermatology	**0.90**	**0.90**	**0.90**	**0.92**	**0.92**	**0.92**
8. Ecoli	**0.58**	**0.61**	**0.59**	**0.65**	**0.69**	**0.67**
9. Indian Liv. Pat.	0.58	0.58	0.58	0.58	0.58	0.58
10. Iris	0.96	0.96	0.96	0.96	0.96	0.96
11. Libras Mov.	0.88	0.87	0.87	0.88	0.87	0.87
12. Lung Cancer	**0.45**	**0.51**	**0.47**	**0.50**	**0.58**	**0.52**
13. Parkinsons	0.81	0.82	0.81	0.81	0.82	0.81
14. Pima Disease	0.67	0.66	0.66	0.67	0.66	0.66
15. Seeds	0.90	0.90	0.90	0.90	0.90	0.90
Average	**0.71**	**0.72**	**0.71**	**0.72**	**0.73**	**0.72**

7.7.5 SkNN Security

In the same manner as when the security of UDMs and EUDMs were considered, using the SkNN approach, the TPDM and QOs are assumed to be non-colluding parties, and the TPDM is considered to be a "passive adversary" who follows the semi-honest model

where the proposed solution (algorithms and protocols) are honestly executed. As noted previously (Section 2.5) this is a reasonable assumption since the primary objective of CSPs, acting as TPDMs offering DMaaS, is to deliver a high quality services to clients (data owners). The private data belonging to a data owner and the private queries belonging to a QOs are not shared with any other parties in the proposed solutions. The TPDM is the only party who gains access to the encrypted dataset D', the SCDM and the query binding records. No decryption takes place at the TPDM side which implies even more security. Therefore, the associated threats when the attacker has access to D' and $SCDM$ are as discussed in Sub-section 7.5.1. Therefore only the SQC protocol security, and the binding and authorising process security, are discussed in this section.

The SQC protocol operates between the data owner and QOs as presented in Sub-section 7.6.2.1. Using this protocol, the parameters required to encrypt a query record q are only exchanged in cyphertext form. The confidentiality of the FDH-OPE keys was guaranteed by encrypting the required parameters, *scale* and e, using Liu's FHE scheme. Therefore, the security of SQC depends on the security of Liu's FHE scheme which has been demonstrated to be semantically secure.

The QO authorisation and binding process are dependent on the security of the FDH-OPE scheme. The binding records, BindRec_1 and BindRec_2 are sent to the TPDM in encrypted from using the FDH-OPE scheme. The TPDM will have access to the order of the distance between random record p and the first record r_1 and the order of the distance between query record q and the random record p. The latter is encrypted using Liu's FHE scheme. The only potential attack is a COA as discussed in Sub-section 7.5.1. In the case where the QOs are active adversaries; the QOs will deviate from the designated, pre-designed, algorithm. The only affect is that the QO may receive a wrong prediction label.

7.8 Summary

This chapter has presented the concept of Secure Chain Distance Matrices (SCDMs) that utilise the Cryptographic Ensemble idea, presented earlier, to provide for secure outsourced data mining. The proposed approach seeks to reduce the required memory resource, in comparison with the UDM and EUDM approaches, by minimising the number of elements in the SCDM. The SCDM has a chain feature that allows data similarity determination between records. The level of data owner participation when using the SCDM concept is the same as in the case of the EUDM concept. The SCDM can support a number of different kinds of data mining algorithms. The application of SCDMs was considered in the context of three different data clustering mechanisms (k-Means, NNC and DBSCAN) and data classification using kNN. The main advantage of SCDMs, over the forgoing approaches presented in Chapters 4 and 5, was that it dramatically reduced the required memory resource by factor equivalent to the number of records in the input data. Moreover, the SCDM provided a scalable solution. The evaluation demonstrated that the proposed secure algorithms founded on the SCDM and the Cryptographic Ensemble, produced comparable results to the equivalent standard algorithms, this indicating that the approach was correct. Recorded differences were due to the semantic security property of the FDH-OPE scheme and the associated chain feature, which in turn resulted in an accumulation of the amount of noise. In the following chapter the concept of SCDMs is extended to facilitate the multiple data owner scenario; collaborative mining using the Super SCDM (SSCDM) concept.

Chapter 8

Super Secure Chain Distance Matrices

8.1 Introduction

In the previous chapter the SCDM concept, founded on the Cryptographic Ensemble idea, and its application in the context of secure data clustering and classification was presented. The discussion of the application of SCDMs was focused on the single data owner scenario. In this chapter the SCDM concept is extended to the multiple data owner scenario, again in the context of the subsidiary research question considered in Chapter 6, namely:

- Can the proposed single data owner scenario solutions be extended to support scalable collaborative data mining (the multiple data owners scenario) while keeping the data owner participation at a minimum?

The involvement of multiple data owners imposes new security challenges; as discussed in [49] and briefly highlighted in Sub-sections 2.3.2 and 2.5.2. To facilitate the multiple data owner scenario the Super Secure Chain Distance Matrix (SSCDM) is presented in this chapter. The SSCDM concept is an extension of the SCDM concept presented in the previous chapter. The fundamental idea was for individual data owners to locally generate a CDM following the process given in Algorithm 15. Each CDM is then encrypted using the MUOPE scheme, presented in Chapter 6, to arrive at a Secure CDM (SCDM). The Super SCDM (SSCDM) is then generated using a Semi-honest Third Party (STP) who "binds" multiple SCDMs to arrive at a single SSCDM. The process for binding multiple SCDMs to form a single SSCDM depends on the nature of the partitioning; how the data is split across the participating parties. This chapter presents also the "binding" processes required for horizontal, vertical and arbitrary data partitioning, in the multiple data owner context, designed to maintain the joint requirements for low memory resource and limited data owner participation. The SSCDM idea allows the calculation of distances between records distributed over multiple data sources, without having access to "actual" data records and without resorting to the SMPC protocols who's use is frequently reported in the literature [36, 37, 40, 102, 103].

An issue with the distance matrix solutions presented in earlier chapters is that although the clustering (classification) has been conducted using a proxy for the real data (UDMs, EUDMs and SCDMs), the proposed matrices have been accompanied with an encrypted version of the actual data. The reason for this was to make the proxies proposed as versatile as possible. For example to support k-Means clustering where

access to the original data was required, as well as other forms of clustering such as NNC and DBSCAN. The vision of the work presented in this chapter is that the original data should not be sent to the TPDM in any form. If the versatility of the proxies is reduced, for example by precluding k-Means clustering, this ambition can be fully realised. This is explored in this chapter using the concept of *virtual record* lists.

As in the case of all the secure clustering approaches considered in earlier chapters the secure collaborative clustering approaches considered in this chapter, at a high level, operate in a similar manner. In this chapter this high level process is referred to as Secure CLustering (SecureCL). The SecureCL process is compatible with a number of clustering approaches. Two are considered in this chapter, Secure DBSCAN and Secure NNC. Secure k-Means is not considered for reasons outlined above.

The rest of this chapter is organised as follows. Section 8.2 presents the SecureCL algorithm. This is followed in Section 8.3 with a presentation of the SSCDM concept and the generation of SSCDMs with respect to three types of data partitioning; horizontal, vertical and arbitrary. SSCDM management, the adding and deleting of records, is then considered in Section 8.4. Section 8.5 presents the SecureCL high level algorithm, and the SSCDM concept, with respect to a range of clustering algorithms including Secure DBSCAN and Secure NNC. An extensive evaluation of the SSCDM concept, and the SecureCL algorithms, was conducted and reported on in Section 8.6. The chapter is concluded with a summary in Section 8.7.

8.2 Secure CLustering

As noted in the introduction to this chapter the collaborative clustering algorithms presented in this chapter conform, at least at a high level, to a generic Secure CLustering (SecureCL) process. A schematic of SecureCL is presented in Figure 8.1. The numbering used in the figure indicates the individual steps in the process as follows.

1. Each individual data owner (p_i) creates their own CDM ($\mathbf{CDM}_i$) from their local dataset D_i (as presented in Algorithm 15). Recall that the CDM, is a $(|D_i| - 1 \times a)$ matrix designed to hold "distance" values, where $|D_i|$ is the number of records in the dataset D_i and a is the number of attributes in the associated attribute set A.

2. The Semi-honest Third Party (STP) and data owners generate the key for the MUOPE scheme, as presented in Algorithm 11. Recall that the MUOPE scheme is an OPE scheme designed for use with respect to data distributed across multiple data owners.

3. Each data owner encrypts their CDM, using the generated MUOPE scheme. The scheme preserves the ordering of the encrypted data; this means that the similarity between data records can be determined, an essential requirement for data clustering.

4. An SSCDM is then generated by the STP with some data owner participation. The SSCDM is a combination of the SCDMs generated by the individual data owners. The SSCDM is constructed according to the data partitioning featured in the global dataset, using a binding process, as discussed in Section 8.3.

5. The STP passes the SSCDM to the TPDM, after which STP is then ready to repeat the process with new datasets and/or new data owners.

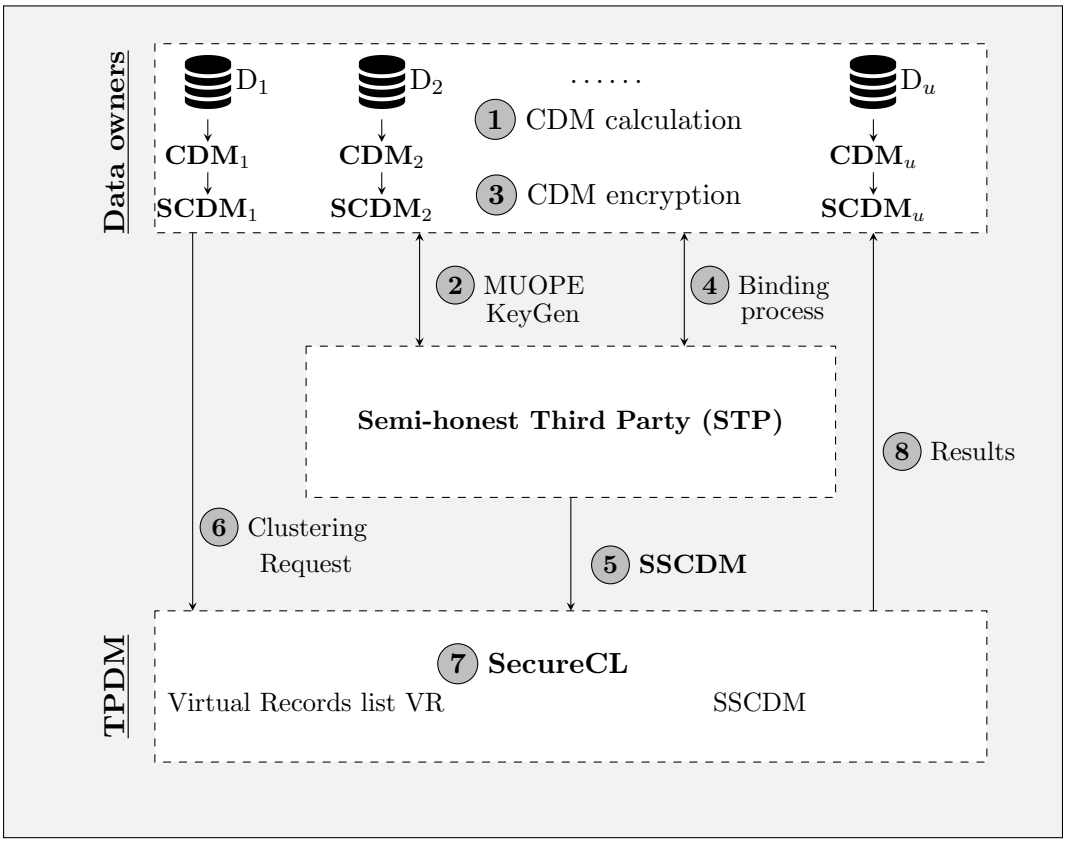


FIGURE 8.1: Schematic of the SecureCL process

6. The TPDM is now in a position to provide secure collaborative data clustering as a service (SecureCL). Data owners can request a clustering by specifying a supported clustering algorithm together with the associated parameters. Data owners can launch secure data clustering over the encrypted data proxy, without further data owner involvement.

7. The entire secure data clustering is conducted by the TPDM. The data similarities between records, regardless of their data owner, is determined using the SSCDM as per equation 8.1 and compared using the ordering preserved in the generated MUOPE cyphertexts.

8. The results of the data clustering are returned to participating parties, in such a way that each party will receive the results for their own records.

$$sim(\mathbf{SSCDM}, vr_x, vr_y) = \sum_{j=1}^{j=a} \left| \sum_{i=x}^{i=(y-1)} sscdm_{i,j} \right| \tag{8.1}$$

So as to realise the vision of secure data mining espoused within the context of this thesis, as noted in the introduction to this chapter, the SecureCL process uses a Virtual Record (VR) list instead of encrypted versions of the data to be processed. A VR is simply a list of cluster identifiers which on start up will simply contain default values. Figure 8.2, shows the structure of a VR list. From the figure it can be seen that the list indices refer to the records held by data owners, index numbers 1 to $|SCDM_1| + 1$ for records belonging to data owner p_1, $|SCDM_1| + 2$ to $(|SCDM_1| + |SCDM_2| + 2)$ for

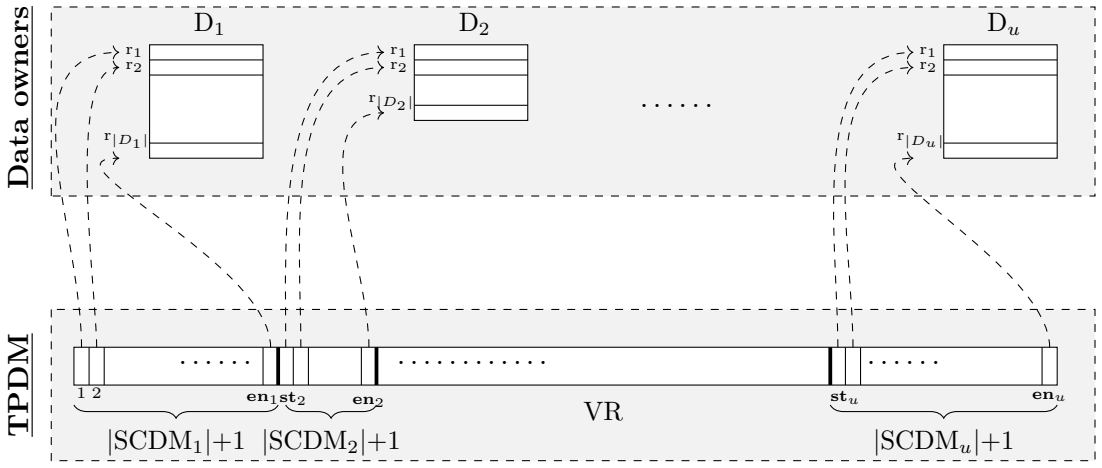


FIGURE 8.2: Example of using Virtual Records (VR) list indices to refer to dataset records held by different data owner

records belonging to data owner p_2, and so on. The overall length of a VR matches the number of records in the global datasets; $D{=}\cup_{i=1}^{i=u} D_i$. A VR list can thus formally be defined as $VR = \{vr_1, \ldots, vr_{|SSCDM|+1}\}$. The start point, st_i, for the indices associated with a particular data owner p_i is calculated as per Equation 8.2; the end point, en_i, is calculated as $en_i = st_i + |SCDM_i|$.

$$st_i = \begin{cases} 1, & \text{if } i = 1 \\ (\sum_{z=1}^{z=i-1} |SCDM_z| + 1) + 1, & \text{otherwise} \end{cases} \tag{8.2}$$

8.3 Super Secure Chain Distance Matrices

As already noted a SSCDM is a combination of one or more SCDMs generated by individual data owners. The SCDMs are generated in two steps: (i) CDM calculation and (ii) CDM encryption using the MUOPE scheme to arrive at a SCDM. The SCDMs are then "bound" together to form a SSCDM. As noted earlier, the nature of the binding process depends on the nature of the partitioning. Three different forms of data partitioning can be identified: horizontal [64], vertical [65] and arbitrary as discussed previously in Sub-section 2.3.2. The binding process in the context of each of these is discussed in the following three sub-sections respectively.

8.3.1 Binding Process for Horizontally Partitioned Data

Horizontal data partitioned, as defined in Sub-section 2.3.2, is where the participating parties conform to the same set of attributes, $A{=}\ \{v_1, \ldots, v_a\}$, but each holds different records [64]. In other words, the global dataset D is decomposed into "horizontal" segments each belonging to a single party. The global data schema is thus the same for all participant's data. Multiple SCDMs, representing horizontally partitioned data, can be "bound" by the STP with a very limited data owner participation, to form a SSCDM using the *HorizontalBinding* process presented in Algorithm 23. The inputs are: (i) an "ID" number i that represent the ID of the last party whose SCDM was bound to the SSCDM, (ii) the $SCDM_{i+1}$ belonging to party p_{i+1} (the current party) to be bound with SSCDM, and (iii) the global SSCDM (**SSC**) accumulated so far. At the beginning of the process the **SSC** so far is $SCDM_1$ belonging to the first party and

$i = 1$. The algorithm starts with the STP generating a random record *Rand* of length a, $Rand = \{rand_1, rand_2, \ldots, rand_a\}$ which is then encrypted using the MUOPE scheme (line 2). The record *Rand* is sent to p_i and p_{i+1}. Party p_i will calculate the difference between the MUOPE cypher of the last record in their local dataset D_i and record *Rand* to give $C_1 = \{c_{1,1}, c_{1,2}, \ldots c_{1,a}\}$ (line 3). At the same time p_{i+1} will calculate the difference between *Rand* and the MUOPE cypher of the first record in their dataset D_{i+1} to give $C_2 = \{c_{2,1}, c_{2,2}, \ldots c_{2,a}\}$ (line 4). C_1 and C_2 are then sent back to the STP. On receiving C_1 and C_2 the STP will next generate a *Pivot* record by adding each attribute in C_1 with the corresponding attribute in C_2 (line 5). The *Pivot* record will then be used to bind $SCDM_i$ (already added in **SSC**) and $SCDM_{i+1}$ (line 6). The process exits with the **SSC** accumulated so far. The process will be repeated using $SCDM_{i+1}$ and $SCDM_{i+2}$ until the entire SSCDM (**SSC**) is constructed.

Algorithm 23 Horizontal binding process

1: **procedure** HORIZONTALBINDING($i, SCDM_{i+1}$, **SSC**)
2: **STP**: $Rand = \{rand_1, rand_2, \ldots, rand_a\}$ that is encrypted using MUOPE
3: $\mathbf{P}_i$: C_1 = Distances between MUOPE cypher of last record in D_i and *Rand*
4: $\mathbf{P}_{i+1}$: C_2 = Distances between *Rand* and MUOPE cypher of first record in D_{i+1}
5: **STP**: $Pivot = C_1 + C_2$
6: **STP**: **SSC** = $concatenate$(**SSC**, $Pivot, SCDM_{i+1}$)
7: **Exit** with **SSC**
8: **end procedure**

8.3.2 Binding Process for Vertically Partitioned Data

Vertical data partitioning is where the participating parties conform to the same set of records, but each holds different attributes derived from a global set of attributes A [65]. In other words, the global dataset D is decomposed into "vertical" segments each belonging to a single party. In this context, multiple SCDMs can be bound following the process given in Algorithm 24. The inputs are $SCDM_i$ belonging to party p_i and the SSCDM (**SSC**) accumulated so far. At the beginning of process, the SSCDM so far is simply $SCDM_1$, the SCDM belonging to the first data owner. The vertical binding process does not require generation of a *Pivot* record as in the case of horizontal binding; it operates by simply appending the SCDMs to one another. The algorithm will exit with an updated SSCDM comprised of the bound SCDMs for parties p_1 to p_i. This process will be repeated until the entire SSCDM (**SSC**) is constructed.

Algorithm 24 Vertical binding process

1: **procedure** VERTICALBINDING($SCDM_i$, **SSC**)
2: **SSC** = $concatenate$(**SSC**, $SCDM_i$)
3: **Exit** with **SSC**
4: **end procedure**

8.3.3 Binding Process for Arbitrary Partitioned Data

The arbitrary data partition generalises the vertical and horizontal partitioning cases. The data is assumed to comprise an arbitrary collection, hence there is no agreed global schema. As such, arbitrary partition data dictates an alternative SCDM binding. The

process commences with a *schema agreement* process to derive an ordered global set of attributes A and an ordered global set of records R, this is orchestrated by the STP. The binding process is then as shown in Algorithm 25. The inputs are: (i) the number of attributes a and number of records n that result from the *schema agreement* process, (ii) the current SCDM ($SCDM_i$) belonging to data owner p_i and (iii) the SSCDM so far (**SSC**). On start up the **SSC** so far will simply be a zero valued $(n-1) \times a$ matrix. The process commences with the construction of a temporary $(n-1) \times a$ SSCDM, $SCDM'$, initially populated with only zero values (line 2), to which $SCDM_i$ is added (line 3) in such a way that matches the order of the agreed attribute schema. The updated SSCDM is then constructed by adding the content of $SCDM'$ to **SSC** (line 4). The algorithm exits with an updated SSCDM so far (that binds the SCDMs belong to parties p_1 to p_i) in line 5. This process will be repeated until the entire SSCDM (**SSC**) is constructed.

Algorithm 25 Arbitrary binding process

1: **procedure** ARBITRARYBINDING($a, n, SCDM_i$, **SSC**)
2: $SCDM' = n - 1 \times a$ matrix populated with 0 values
3: $SCDM' = SCDM' + SCDM_i$
4: **SSC** = **SSC** + $SCDM'$
5: **Exit** with **SSCDM**
6: **end procedure**

8.4 SSCDM Management

Given any data mining scenario directed at an evolving dataset, there is the potential that new records may be added or existing records may be deleted. However, where this occurs, the SSCDM does not need to be regenerated, the relevant content can simply be updated. This process is referred to as *SSCDM management* and is considered in this section. New records when added to participant datasets can be simply inserted in to the SSCDM so far, in the part that holds the participant in question's SCDM, in a manner similar to the binding processes described above depending on the nature of the data partitioning. This will consequently require revision the participant's SCDM length held by the TPDM and the VR list.

Deletion of records is more complex, the appropriate SSCDM elements need to be removed and new pivot records need to be derived. The pseudo code for achieving this is given in Algorithm 26. The inputs are: (i) the index (*index*) of the record to be deleted with reference to the participant's dataset, (ii) the size of the attribute set a, (iii) the SSCDM (**SSC**) and (iv) the virtual record list VR. The algorithm commences by determining the caller party id, p_{id} (the party that wishes to delete the record), using the *Caller* procedure (line 2). The index associated with the record to be deleted, of party p_{id}, in the SSCDM (**SSC**) and the VR list are determined using the *CalculateIndex* procedure which returns the SSCDM index i and the virtual id record $vr_{virtualId}$ in the VR list (line 3). In the case when the index of the record to be deleted is the first or the last element in the SSCDM, the updating process is accomplished by removing ssc_i and updating the VR list by deleting $vr_{virtualId}$ (line 9). Otherwise, the updating process is accomplished by generating a new record that replaces two SSCDM rows, ssc_{i-1} and ssc_i. The newly generated record will then be used to update ssc_{i-1} (lines 5 and 7), while ssc_i will be deleted (line 9). The procedure will exit with an updated SSCDM and an updated virtual record list VR.

Algorithm 26 Delete SSCDM element

```
1: procedure DeleteSSCDMElement(index, a, SSC, VR)
2:     p_id ← Caller()
3:     i, virtualId ← CalculateIndex(SSC, index, p_id)
4:     if (i ≠ 1 and i ≠ |SSC|) then
5:         for j = 1 to j = a do
6:             ssc_{i−1,j} = ssc_{i−1,j} + ssc_{i,j} (ssc_{i,j} ∈ SSC)
7:         end for
8:     end if
9:     Delete ssc_i, vr_{virtualId}
10:    Exit with SSC and VR
11: end procedure
```

8.5 SecureCL: Secure Collaborative Data Clustering

This section presents two secure clustering algorithms, a secure DBSCAN implementation and a secure NNC implementation, that make use of the proposed SSCDM concept coupled with the Cryptographic Ensemble idea. Both of the algorithms represent implementations of the SecureCL "parent" algorithm presented above. Note that the algorithms do not entail any data owner participation and do not require resort to SMPC protocols for data comparison as in the case of earlier work [20, 33–37, 40, 102, 103, 105]. The secure DBSCAN implementation (SDBSCAN) is described further in Sub-section 8.5.1 and the secure NNC implementation (SNNC) in Sub-section 8.5.2.

8.5.1 Secure DBSCAN

Using the Secure DBSCAN (SDBSCAN) algorithm the clustering is entirely conducted by the TPDM, but following a process very similar to the standard DBSCAN [47]. The pseudo code is given in Algorithm 27. The inputs are: (i) the SSCDM (**SSC**) received from the STP, and (ii) the desired density parameters, minimum number of points ($MPts$) and radius (ϵ') as agreed by the participating parties. Recall that the actual data will not be sent to the TPDM; only the SSCDM will be sent to the TPDM. The ϵ value is encrypted using the proposed MUOPE scheme to give ϵ' so that the TPDM does not have the real radius value. Hiding the real radius value is seen as essential to hide the correlation between data records when the TPDM compares the distances between data records and the radius value, so as to prevent the launch of Overlapping Attacks (OAs) as discussed in Sub-section 2.5.2. The SDBSCAN algorithm commences by: creating a "virtual" record list VR comprised of the individual record IDs, VR= $\{vr_1, vr_2, \ldots, vr_{|SSCDM|+1}\}$ and an empty set of clusters C; and by initialising the number of cluster so far to 0 (line 2). The virtual record list VR is then processed (lines 3 to 14). For each record $vr_i \in VR$ that has not been previously assigned to a cluster, is "unclustered", the set S is determined (lines 4 and 5). The set S is the ϵ-neighbourhood of vr_i and comprises the set of virtual record IDs in VR whose distance from vr_i is less than or equals to ϵ'. The set is determined by calling the *RegionQuery* sub-procedure, given in lines 30 to 39, where **SSC** is used to determine the overall distances between records (see Equation 8.1). If the number of records in S is greater than $MPts$ the density requirement is satisfied and thus vr_i is marked as "clustered" and considered to represent a new cluster C_k (lines 6 to 9). This cluster is then expanded by considering the virtual records in S using the *ExpandCluster* sub-procedure called in line 10. The inputs to the *ExpandCluster* are: (i) the cluster C_k so far, (ii) the set S, (iii) **SSC**

Algorithm 27 Secure DBSCAN (SDBSCAN) clustering algorithm

```
1: procedure SDBSCAN(SSC, MPts, ϵ′)
2:     VR =list of virtual record IDs that set all to be "unclustered" , C = ∅, k = 0
3:     for i = 1 to i = |VR| do
4:         if vr_i is unclustered then
5:             S = RegionQuery(vr_i, ϵ′, SSC)
6:             if |S| ⩾ MPts then
7:                 mark vr_i as clustered
8:                 k = k + 1
9:                 C_k = vr_i (new cluster)
10:                C_k = ExpandCluster(C_k,S, SSC,ϵ′,MPts)
11:                C = C ∪ C_k
12:            end if
13:        end if
14:    end for
15:    Exit with C
16: end procedure
17: procedure EXPANDCLUSTER(C, S, SSC, ϵ′, MPts)
18:    for ∀ vr_i ∈ S do
19:        if vr_i is unclustered then
20:            mark vr_i as clustered
21:            C = C ∪ vr_i
22:            S_2 = RegionQuery(vr_i, ϵ′, SSC)
23:            if |S_2| ⩾ MPts then
24:                C = ExpandCluster(C, S_2, SSC, ϵ′, MPts)
25:            end if
26:        end if
27:    end for
28:    Exit with C
29: end procedure
30: procedure REGIONQUERY(vr_index, ϵ′, SSC)
31:    N_ϵ = ∅
32:    for ∀ vr_j ∈ VR do
33:        distance = sim(SSC, vr_index, vr_j)          ▷ Equation 8.1
34:        if distance ⩽ ϵ′ then
35:            N_ϵ.add(j)
36:        end if
37:    end for
38:    Exit with N_ϵ
39: end procedure
```

and (iv) the density parameters ϵ' and $MPts$. The *ExpandCluster* sub-procedure is a recursive procedure (lines 17 to 29). For each record in S which has not been previously assigned to a cluster we add the record to C_k and then determine the ϵ-neighbourhood S_2 for this record (lines 18 to 22). If the size of S_2 is greater than $MPts$ we call the *ExpandCluster* sub-procedure again (lines 23 to 25), and so on until all the records in VR are processed. The algorithm will exist with the cluster configuration C. At the end of the data clustering process each party will only receive from the TPDM the cluster labels for their own data (virtual record list).

8.5.2 Secure NNC

The Secure NNC (SNNC) algorithm, as in the case of all SecureCL derivative algorithms, is conducted, by the TPDM, in a similar manner to that of standard NNC [48] as summarised in Algorithm 28. The inputs are the SSCDM (**SSC**) and the desired SNNC threshold σ'; the later agreed by the participating parties. To allow secure data clustering the threshold σ is encrypted using MUOPE to give σ'. As in the case of the SDBSCAN algorithm, the SNNC algorithm commences by creating the "virtual" record list VR and an empty cluster set C; and initialising the cluster so far variable to 1 (line 2). The first virtual record (vr_1) is added to the first cluster (lines 3 and 4), then iteratively the remaining records are assigned to clusters (lines 5 to 15). As for standard NNC, virtual record vr_i will be assigned to cluster C_m if there exist some virtual record vr_j in cluster m whose distance from vr_i is less than or equals to σ'. Otherwise, vr_i is assigned to a new cluster. Therefore, in line 6, the *findNearestRecordCluster* sub-procedure is called to return the cluster ID (m) of the nearest record and their corresponding distance value ($smallDist'$). The *findNearestRecordCluster* sub-procedure, given in lines 18 to 30. To this end the **SSC** is used to determine the similarity between records (line 23). The record vr_i will be assigned to cluster m when the distance ($smallDist'$) is less than the threshold σ' (lines 7 to 9), otherwise a new cluster is generated (lines 11 to 13). The algorithm exits with clustering configuration C in line 16.

8.6 Experimental Results and Evaluation

This section reports on the evaluation of the SSCDMs concept in the context of SecureCL. The evaluation was conducted using datasets selected from the UCI machine learning repository as listed in Table 4.1 of Chapter 4, and synthetic datasets increasing in size from $1,000$ to $10,000$ records in steps of $1,000$. The objectives of the evaluation were as follows:

1. **Time Complexity of data owner participation**: To analysis the runtimes required for data owner participation when preparing the data using the SSCDM approach.
2. **Collaborative clustering efficiency**: To compare the efficiency of the proposed SecureCL algorithms with the "standard" equivalent algorithms that operate over unencrypted data.
3. **Collaborative clustering accuracy**: To compare the accuracy of final clustering configurations obtained using the proposed SecureCL algorithms with respect to the standard algorithms (using the same algorithm parameters).
4. **Collaborative clustering scalability**: To analysis the scalability of the proposed approach by considering the effect on runtimes as the number of participants is increased.
5. **Security**: To analysis the security when adopting the SSCDM concept, and the proposed binding process, in terms of potential attacks.
6. **Comparison of SSCDM and GEDM for collaborative data clustering**: To compare the SSCDM concept presented in this chapter with the GEDM concept presented in Chapter 6 in terms of: (i) data owner participation prior to data outsourcing, (ii) efficiency, (iii) accuracy and (iv) required memory resources.

Algorithm 28 Secure Nearest Neighbour Clustering (SNNC)

```
1: procedure SNNC(SSC, σ′)
2:     VR =list of record IDs, C = Empty set of clusters, k = 1
3:     C_k = {vr_1}
4:     C = C ∪ C_k
5:     for i = 2 to i = |VR| do
6:         m, smallDist′ ← findNearestRecordCluster(i, C, SSC)
7:         if smallDist′ ⩽ σ′ and m ≠ null then
8:             C_m = C_m ∪ vr_i
9:             Update the cluster set C_m in the set of clusters C
10:        else
11:            k = k + 1
12:            C_k = {vr_i}
13:            C = C ∪ C_k
14:        end if
15:    end for
16:    Exit with C
17: end procedure
18: procedure findNearestRecordCluster(index, C, SSC)
19:    smallDist′ =MaxNumber and m = null
20:    for clusterID = 1 to clusterID = |C| do
21:        for j = 1 to j = |C_clusterID| do
22:            rid = Get(j, C_clusterID) where rid is the jth record in C_clusterID
23:            if sim(SSC, vr_index, vr_rid) < smallDist′ then        ▷ Equation 8.1
24:                smallDist′ = sim(SSC, vr_index, vr_rid)
25:                m = clusterID
26:            end if
27:        end for
28:    end for
29:    Exit with m and smallDist′
30: end procedure
```

Each of these objectives is considered in the following six sub-sections, (Sub-section 8.6.1 to Sub-section 8.6.6). The SDBSCAN and SNNC algorithms, and the associated processes, were all implemented using the Java programming language. All the experiments reported in this chapter were conducted using 3.8 GHz Intel Core i5 with 8GB 2400 MHz DDR4 memory, running under the macOS High Sierra operating system.

8.6.1 Complexity of Data Owner participation

The complexity of data owner participation with respect to the calculation and encryption of their CDMs (*CDM Cal.* and *CDM Enc.*), and data density calculation (*Dens. Cal.*), was presented and discussed previously in Sub-section 7.5.1 of Chapter 7 and Sub-section 6.5.1 of Chapter 6, respectively. The observation was, not unexpectedly, that runtime increased with dataset size. The experiments were therefore not repeated with respect to SecureCL.

However, experiments were conducted, using synthetic datasets, directed at evaluating the effect on data owner participation of increasing the number of records (n) and increasing the number of attributes (a). The data used in the experiments belonged to

one data owner. In each case the size of the targeted dimension (n or a) was increasing from $1,000$ to $10,000$ in steps of $1,000$, while the other dimension was kept constant at 100. The results are presented in Figure 8.3. As expected, the average runtime required to generate the CDM, encrypt the CDM and calculate data density increased *linearly* as the size of data records n and attributes a increased. For example, when $a = 1K$, where K is $1,000$, the CDM was generated in $63.73ms$, encrypted in $168.04ms$ and density calculated in $42.27ms$, when $a = 10K$ the corresponding runtimes are $468.31ms$, $1101.37ms$ and $228.3ms$. The recorded runtimes when $n = 1K$ were $60.7ms$, $158.57ms$ and $42.89ms$, compared to $569.99ms$, $1225.79ms$ and $293.09ms$ when $n = 10K$. These results shown that, regardless of dataset size, at least in the context of the conducted experiments, the runtime associated with data owner participation was not significant and therefore did not introduce any limiting overhead with respect to the data owner.

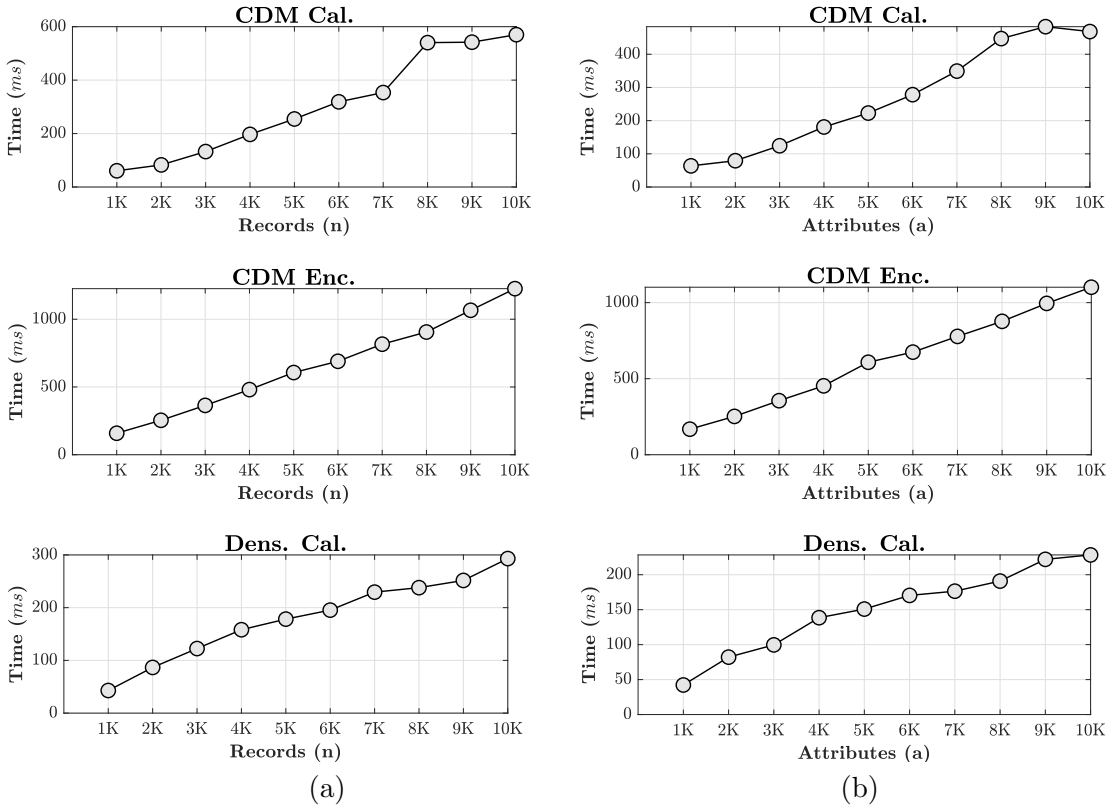


FIGURE 8.3: Average runtimes (ms) for CDM calculation, CDM encryption and density calculation using a range of values for n (number of records) and a (number of attributes)

Note that when using SecureCL, because of the limitations imposed (see discussion at the start of this chapter) no data owner participation was required once clustering was in progress. This was achieved without recourse to key delegation, as in the case of "secret sharing" (as found in [19]), and without resorting to SMPC protocols (for example as found in [20, 33–37]).

8.6.2 Collaborative Clustering Efficiency

The utilisation of the SSCDM data proxy for secure data comparison will clearly introduce a computational overhead compared to standard clustering algorithms (using plaintext data). To evaluate the computational overhead associated with the secure

approach the runtimes required to cluster the experimental datasets using standard algorithms were compared with the runtimes required using the proposed SecureCL algorithms. The fifteen UCI machine learning repository datasets given in Table 8.1 were used for this purpose.

The results are presented in Figure 8.4; the numbering used on the x-axes matches the dataset numbering used in Tables 8.1 and 8.2. The algorithm thresholds and parameters ($MPts$, ϵ and σ) are as given in columns 2 and 3 of Table 8.1 and column 2 of Table 8.2 and were randomly selected from a sequence of experiments (not reported here); in practice these will be agreed and specified by the data owners (how this might be done is beyond the scope of this thesis). The results indicated that: (i) the runtimes required by SDBSCAN and SNNC, as expected, are greater than those require using standard DBSCAN and NNC, and (ii) the efficiency of the SecureCL algorithms is not correlated to the number of parties involved as the data clustering is entirely delegated to the TPDM.

The difference between the standard and secure data clustering is caused by the utilisation of SSCDMs to determine similarity. The usage of SSCDM, as in the case of the SCDM concept, depends on the size of the datasets; the bigger the dataset the larger the SSCDM the greater the time required to use the SSCDM for determining similarity. However, it is argued here that these runtimes were still in acceptable bounds. As the SSCDM concept is an extension of the SCDM concept, the reader might find it useful to refer back to Sub-Section 7.5.2 where the usage of SCDM for data similarity, in the context of different sizes of records, was evaluated.

8.6.3 Collaborative Clustering Accuracy

The "correctness" of cluster configurations produced using SecureCL, either SDBSCAN or SNNC, were measured by comparing the clustering configuration results obtained with those obtained using the equivalent standard (unencrypted) approach. The Silhouette Coefficient (*Sil. Coef.*) was used as the evaluation metric [188]. To demonstrate that the proposed solutions to support collaborative clustering operated correctly, SDBSCAN and SNNC should produces comparable *Sil. Coef.* values to those produced using the standard approaches when using the same algorithm parameters. The number of produced clusters was also compared. The results are presented in Tables 8.1 and 8.2. The standard DBSCAN and NNC results were presented previously in Tables 6.1 and 6.2; however, for comparison purposes, these results have been included in Tables 8.1 and 8.2 (columns 4 and 5 of Table 8.1, and columns 3 and 4 of Table 8.2). From the tables it can be seen that, in most cases, the SecureCL algorithms produced identical configurations to those produced using standard approaches; only in 7 out of 30 cases were different results obtained (highlighted in bold font). The reason for the difference is that the distances calculated using the chaining feature of SSCDMs sometimes caused an accumulated error because of the random noise (δ) included in the MUOPE scheme. Interestingly, with respect to the differing results, the SecureCL algorithms produced slightly better configurations in four out of the seven cases (Iris and Breast cancer in the context of NNC, and Dermatology and Blood Trans in the context of DBSCAN). The Blood Trans dataset produced a different number of classes using NNC but with the same *Sil. Coef.* In the remaining two cases (Breast Cancer in the context of DBSCAN and Dermatology in the context of NNC) the standard approach was slightly better. In the worst case, Breast Cancer, the unencrypted dataset clustered to four classes. The number of records in each cluster were 434, 4, 3, 3 respectively and the remaining records marked as outliers. In the second, third and fourth clusters the distances between

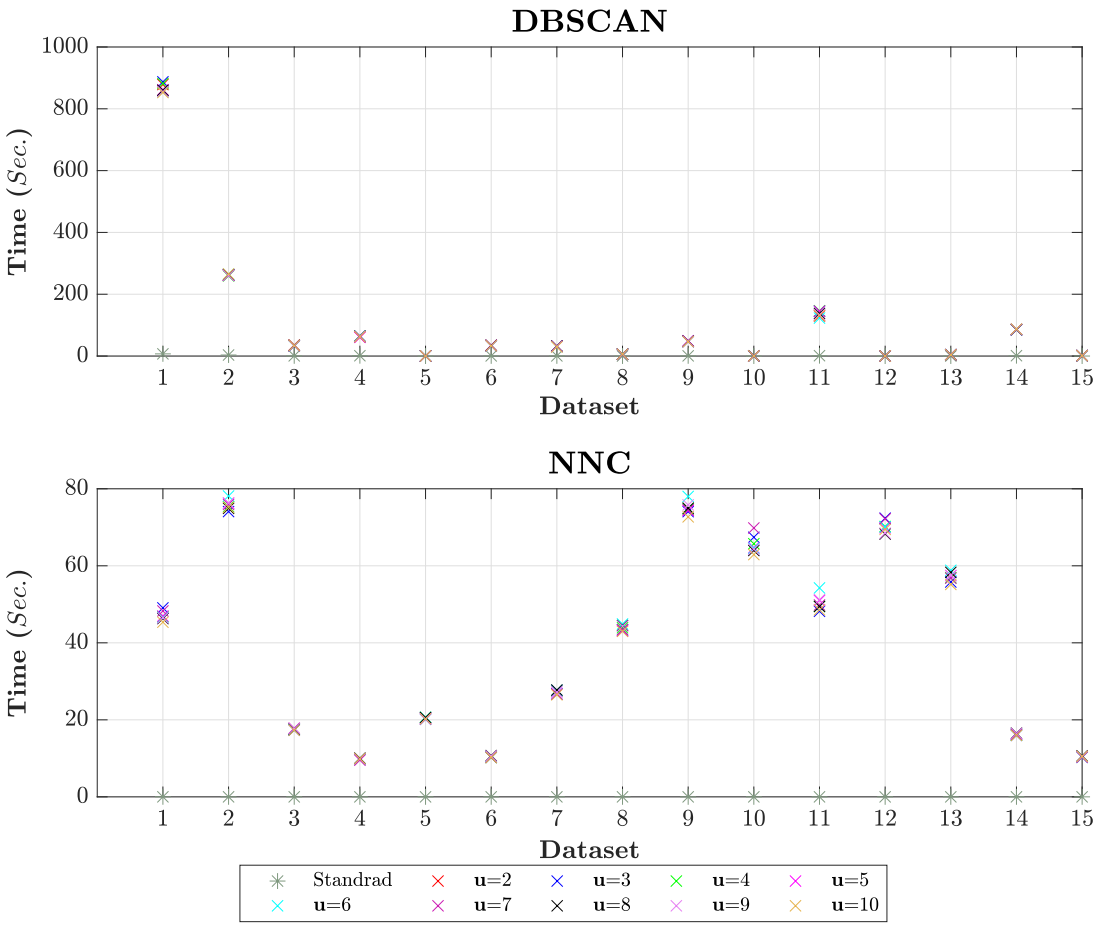


FIGURE 8.4: Comparison of runtimes using standard and secure clustering algorithms founded on the SSCDM concept, for different number of parties (u)

records were exactly equal to 5, the threshold value ϵ for DBSCAN. The probabilistic feature of MUOPE made the comparison between the threshold and, the distance derived using SSCDM unequal and thus clustered these records to the outliers class. The results demonstrate that SecureCL approach provides a suitable solution to secure collaborative data clustering in that accuracy is not adversely affected. The results obtained using the SSCDM concept, in the context of collaborative clustering, corroborated the results obtained using the SCDM approaches presented previously in Chapter 7, when using the same threshold parameters for equivalent algorithms.

8.6.4 Collaborative Clustering Scalability

The scalability of the proposed solution to the collaborative clustering problem was evaluated by analysing the runtime as the number of participating parties (u) was increased. Using SecureCL, increasing the number of participants would have an effect on the efficiency of: (i) the MUOPE key generation process (*MUOPE Key Gen*) and (ii) the binding process for generating SSCDMs (*SSCDM Gen*). Experiments were conducted using a range of values for u from 10 to 100 increasing in step of 5; for completeness $u = 2$ and $u = 4$ were also considered. For the experiments a $n = 7000$ and $a = 125$ synthetic dataset, distributed equally across the participants, was used, and the runtimes recorded. The results concerning the *MUOPE Key Gen* scalability evaluation were as discussed in Sub-section 6.5.5. The results for *SSCDM Gen* are shown in Figure 8.5. As

Table 8.1: Cluster configuration for standard and secure DBSCAN (differing results highlighted in bold font)

UCI Dataset	MPts	ϵ	DBSCAN		SDBSCAN	
			Num. Clus.	Sil. Coef.	Num. Clus.	Sil. Coef.
1. Arrhythmia	2	600	6	0.472	6	0.472
2. Banknote Auth.	2	3	7	0.922	7	0.922
3. Blood Trans.	2	10	**27**	**0.971**	**33**	**0.976**
4. Breast Cancer	2	5	**4**	**0.678**	**1**	**0.485**
5. Breast Tissue	2	100	3	0.628	3	0.628
6. Chronic Kidney	2	70	19	0.970	19	0.970
7. Dermatology	2	10	**16**	**0.853**	**15**	**0.881**
8. Ecoli	2	60	1	-1.000	1	-1.000
9. Indian Liv. Pat.	3	40	7	0.789	7	0.789
10. Iris	5	2	2	0.722	2	0.722
11. Libras Mov.	5	5	11	0.715	11	0.715
12. Lung Cancer	2	20	1	0.053	1	0.053
13. Parkinsons	3	10	5	0.829	5	0.829
14. Pima Disease	5	20	4	0.691	4	0.691
15. Seeds	5	1	7	0.852	7	0.852

Table 8.2: Cluster configuration for standard and secure NNC (differing results highlighted in bold font)

UCI Dataset	σ	NNC		SNNC	
		Num. Clus.	Sil. Coef.	Num. Clus.	Sil. Coef.
1. Arrhythmia	1	452	1.000	452	1.000
2. Banknote Auth.	5	21	0.895	21	0.895
3. Blood Trans.	68	**34**	0.999	**35**	0.999
4. Breast Cancer	10	**108**	**0.903**	**135**	**0.926**
5. Breast Tissue	1	105	1.000	105	1.000
6. Chronic Kidney	100	243	0.999	243	0.999
7. Dermatology	18	**32**	**0.919**	**37**	**0.915**
8. Ecoli	1	2	0.353	2	0.353
9. Indian Liv. Pat.	99	100	0.997	100	0.997
10. Iris	1	**15**	**0.922**	**16**	**0.927**
11. Libras Mov.	4	224	0.969	224	0.969
12. Lung Cancer	1	32	1.000	32	1.000
13. Parkinsons	73	11	0.953	11	0.953
14. Pima Disease	100	22	0.956	22	0.956
15. Seeds	1	103	0.979	103	0.979

expected, the time complexity for both *MUOPE key Gen* and *SSCDM Gen* increased linearly with the number of parties. However, the increased runtime was not significant. The experimental results show that the MUOPE key can be effectively generated even for large numbers of parties; in the case of 100 participants in $1541.39ms$. The binding process runtime for horizontal and vertical partitioning was negligible, whilst for

arbitrary partitioning this was higher due to the requirement for schema agreement.

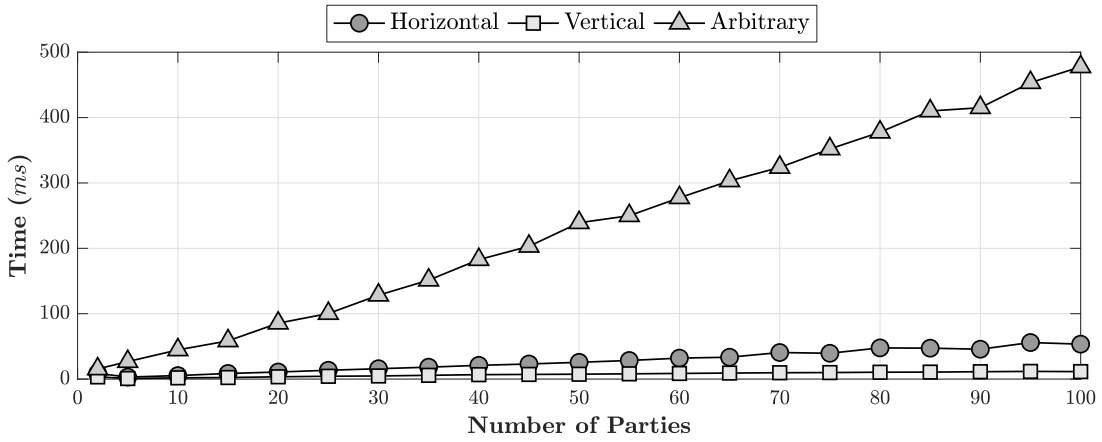


FIGURE 8.5: Runtime to construct SSCDMs for different types of partitioning as the number of participants (data owners) increases

8.6.5 Security

The security of SecureCL was evaluated by identifying potential attacks that could be directed to breach the privacy of the outsourced data. For the purpose of this evaluation, as in the case of similar evaluations reported on in previous chapters, the TPDM was treated as a "passive adversary" in the context of what is usually referred to as the "semi-honest model". Recall that in this model the TPDM is expected to "honestly" execute the SecureCL facilitated clustering; but during the process uses the results, and any intermediate knowledge, to extract additional information about the parties' data, as defined in [17]. This was considered a reasonable assumption since the main objective of the CSP (the DMaaS provider) is to deliver high quality and accurate services to clients (data owners).

Using SecureCL attacks directed over the actual data are entirely precluded because only a proxy for the data was exchanged, the SSCDM encrypted using MUOPE; no actual data (encrypted or otherwise) is therefore confided to the TPDM or shared with any other participants (other data owners). Hence the only possible attacks are Cyphertext Only Attacks (COAs) that can be launched when the attacker somehow has access to the SSCDM. COAs are more likely to succeed when the attacker has background knowledge about the original data (data frequency and/or distribution) that can be used to identify cyphertexts associated with highly frequent data items. However, as a countermeasure to COAs, the proposed MUOPE scheme reduces the information leakage in the generated cyphertexts by obscuring the data frequency and distribution as discussed earlier in Chapter 6. Recall that the data distribution is obscured using the concept of *message space splitting* and *non-linear cypher space expansion*, which is derived from the shared data density, in such a way that message space intervals with high data density will have larger (expanded) cypher space intervals. The data frequency is hidden using the encryption function that generates different cyphertexts for the same plaintext value, even when the same encryption keys are used. In the proposed solution no decryption takes places on the TPDM side; thus providing for additional security. The TPDM, who compares data records using the encrypted SSCDM and assigns records to appropriate clusters, cannot initiate OAs that rely on the results of comparisons and the real data, because the real data will not be available. The entire clustering process is delegated to

the TPDM and each party will receive the cluster labels for their own dataset, hence a non-honest data owner party cannot launch any form of attack.

8.6.6 Comparison of SSCDMs and GEDMs for Secure Collaborative Data Clustering

This section considers the results from the comparative evaluation of the GEDM and SSCDM concepts and their application in the context of secure collaborative data clustering. The evaluation criteria was to consider the two approaches in terms of: (i) the complexity of the data preparation process, (ii) the efficiency of the collaborative data clustering, (iii) the accuracy of the obtained clustering configurations and (iv) the required memory resource. Security was not considered as an evaluation criteria as both approaches utilised the same encryption scheme (the MUOPE scheme), although the SSCDM approach does not required confiding of the data in any form to the TPDM or with other participants. Each of the above criteria is considered in further detail in the following sub-sections, Sub-section 8.6.6.1 to 8.6.6.4.

8.6.6.1 Comparison of SSCDMs and GEDMs In Terms of The Complexity of The Data Preparation Process

This sub-section considers the complexity of the data preparation process when adopting GEDMs, as presented previously in Chapter 6, and that of SSCDMs as presented in this chapter. The complexity, in both cases, was measured in terms of the runtime required to securely prepare datasets for outsourcing.

Using GEDMs for secure collaborative data clustering the preparation process comprises: (i) data encryption, (ii) DM calculation, (iii) DM encryption, (iv) density calculation (required to MUOPE key generation) and (v) participating in the pooling method to construct the GEDM with reference to STP. The complexity of GEDM data owner participation was evaluated using synthetic dataset ranging in size from $1,000$ to $10,000$, increasing in steps of $1,000$. The results are presented in Figures 6.2 and 6.3.

The data owner participation using SSCDMs for secure collaborative data clustering, encompasses: (i) CDM calculation, (ii) CDM encryption, (iii) density calculation and (iv) participating in the binding process to construct SSCDMs. The same synthetic datasets were used for the evaluation. The results obtained were as shown in Figures 8.3 and 8.5.

Table 8.3 shows the total runtime required to prepare the synthetic datasets in both cases. The SSCDM generation and GEDM generation were not considered as the runtimes were dependent on the number of parties involved (the presented runtimes are the average of ten runs). From the table, it can be observed that the GEDM data owner participation is higher than that required by the SSCDM, but closer inspection of the table indicates that this difference is not significant. It can be observed from Figures 6.3(b) and 8.5 that the runtimes for the pooling process using GEDMs is higher than the runtime required to generate the SSCDMs. However, it should also be noted that the data preparation process is a one time process that is conducted before the data is outsourced to the TPDM and thus it will not introduce any significant overhead in both cases.

8.6.6.2 Comparison of SSCDMs and GEDMs In Terms of Efficiency

This sub-section considers the results from the comparative evaluation of the two secure data clustering approaches in terms of runtimes. Tables 6.1 and 6.2, and Figure 8.4,

TABLE 8.3: Average runtimes (*Sec.*) for GEDM and SSCDM data preparation process

	1K	2K	3K	4K	5K	6K	7K	8K	9K	10K
GEDM	1.5	8.2	20.0	47.1	89.4	144.2	166.5	294.9	430.8	1013.6
SSCDM	0.3	0.4	0.6	0.8	1	1.1	1.3	1.5	1.7	1.8

summarise the runtime results obtained using NNC and DBSCAN in the case of both GEDMs and SSCDMs. The reported results show that the runtimes required to cluster datasets using algorithms that use the GEDM concept were much faster than the algorithms that adopted the SSCDM concept; note that the runtime for the SSCDM results is given in Seconds (*Sec.*) whilst for the GEDM results it is given in milli-seconds (ms). The reason for this difference is due to the time required to utilise the SSCDM, and the "chain feature", to determine similarity. Therefore, for completeness, the time required to use the GEDM and SSCDM for determining the similarity between the two data records were also compared by considering datasets with $10K$ records but with varying numbers of attributes from 10 to 100 increasing in steps of 10. Figure 8.6 shows the required runtime for determining the similarity between the first and the last record in each dataset. Regardless of the number of attributes, the GEDM features an almost steady runtime as similarity can be simply "looked up", whilst the runtime required when using SSCDMs increased linearly with the number of attributes and was always higher than the runtime associated with GEDMs.

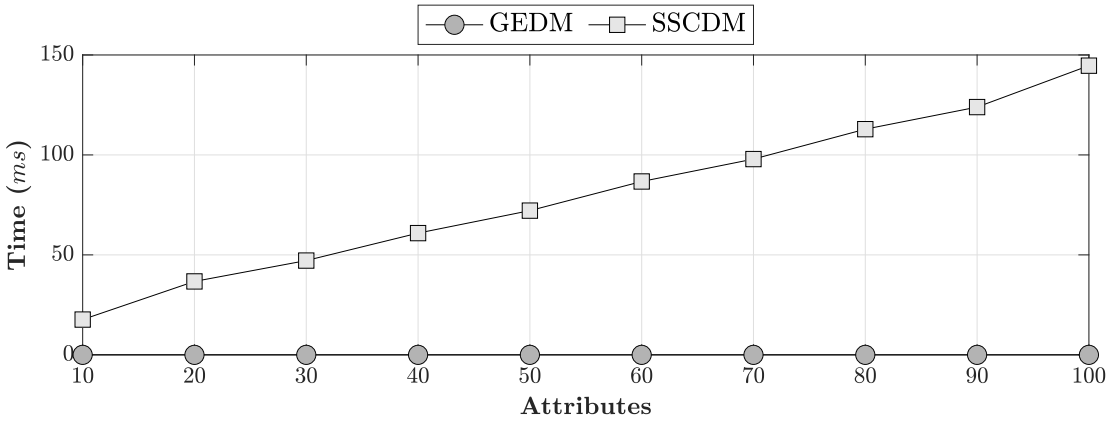


FIGURE 8.6: Time (ms) for TPDM to determine the similarity between two records using the SSCDM and the GEDM concepts

8.6.6.3 Comparison of SSCDMs and GEDMs In Terms of Accuracy

The clustering configuration results obtained using the secure clustering algorithms founded on the SSCDM concept (SNNC and SDBSCAN) were compared with those obtained using the equivalent algorithms founded on the concept of GEDMs (S-NNC and S-DBSCAN). The correctness of the clusters produced in each case was again compared with those obtained using standard equivalent algorithms (NNC and DBSCAN). The same threshold parameters (ϵ, $MPts$ and σ) were used as listed in Tables 6.1 and 6.2. It was found that the SSCDM approach produced clustering configurations *comparable* with those produced using GEDMs and standard equivalents, but not identical (columns 5 and 7 of Table 8.1 and columns 4 and 6 of Table 8.2). The reason for these

difference, as described in Sub-section 8.6.3, was due to the noise added to the MUOPE scheme which accumulated because of the "chain feature" used to determine similarity.

8.6.6.4 Comparison of SSCDM and GEDM In Terms of Memory Resources

Both GEDMs and SSCDMs are 2D matrices. As described in Chapter 6, the GEDM's two dimensions are correlated with the number of records in the dataset; however, due to the similarity around their leading diagonal, only the upper or lower triangle of the GEDM is required. The SSCDM's first dimension is correlated with the number of data records (-1), whilst the second dimension is correlated with the number of attributes. The number of elements within GEDMs and SSCDMs associated with a range of datasets featuring different numbers of records and attributes is presented in Figure 8.7. From the figure, it can be clearly seen that the GEDM is more suited to datasets where the number of attributes is larger than the number of records, whilst the SSCDM is suited to datasets where the number of records is larger than the number of attributes (the more usual case).

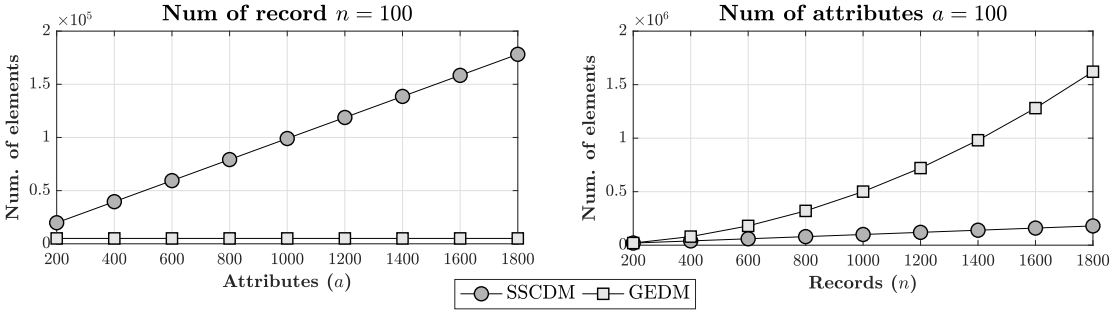


FIGURE 8.7: Number of elements in GEDMs and SSCDMs for different sizes of data

8.7 Summary

This chapter has presented a mechanism, founded on SCDMs, whereby the data to be outsourced is entirely replaced by a proxy for the data, the SSCDM, that supports the required operations for a number of data mining algorithms. The SSCDM data proxy is generated by the data owner in the form of CDMs that are then encrypted using the MUOPE scheme to give SCDMs. Multiple SCDMs can then be combined, using a secure binding process dictated by the nature of the data partitioning features within the global dataset under consideration. The SSCDM is the only "data" sent to a TPDM.

This chapter also presented the SecureCL algorithm, a high-level algorithm directed at collaborative secure data clustering. Two realisations of SecureCL were considered, SDBSCAN and SNNC. The experimental evaluation demonstrated that the SecureCL approach provides the following advantages:

1. The data clustering is entirely delegated to a TPDM without the requirement of resorting to computationally expensive SMPC protocols or secret key sharing.

2. The quality of the data clustering is comparable to that produced using standard, unencrypted methods.

3. No data owner participation is required once the data proxy encryption has been completed.

4. Non-honest data owner attacks, and OAs, are precluded since the third party does not have access to data to conduct the OAs, and data owners are not involved in the process of data clustering.

5. The solution can be scaled to a large number of participants.

In the following chapter the work presented so far is extended to encompass secure neural networks. An approach is presented that allows model training and query prediction to be securely delegated to a TPDM with minimal data owner participation. The following chapter also presents the Modified Liu's Scheme (MLS) that: (i) addresses the cyphertext inflation problem associated with homomorphic multiplication and (ii) facilitates secure data comparisons over cyphertexts without decryption.

Chapter 9

Secure Neural Network Using Modified Liu's Scheme

9.1 Introduction

In the thesis so far a range of secure data clustering mechanisms have been presented. The fundamental idea was the usage of a data proxy coupled with an encryption mechanism or mechanisms. The data proxy in all cases considered so far has been some form of distance matrix, culminating in the Super Secure Chain Distance Matrix (SSCDM) concept presented and evaluated in the preceding chapter. The first form of distance matrix considered, the UDM, did not feature any encryption. However, the subsequent, EUDM, EDM, GEDM, SCDM and SSCDM concepts featured data encryption using OPE schemes. More specifically Cryptographic Ensembles were used where two encryption schemes were used in tandem; one to allow homomorphic addition and multiplication, and one to allow comparison of records. For the first, Liu's FHE scheme was adopted. For the second, two alternative OPE schemes were proposed: (i) the Frequency and Distribution Hiding Order Preserving Encryption (FDH-OPE) scheme for the single data owner scenario, and (ii) the Multi-User Order Preserving Encryption (MUOPE) scheme for the multiple data owner scenario.

However, the use of Cryptographic Ensembles is not ideal, it would be better to use a single encryption mechanism. A further disadvantage with respect to FHE schemes, such as Liu's FHE scheme adopted with respect to many of the solutions presented in the foregoing chapters, is the cyphertext inflation problem where the number of sub-cyphertext increases with each homomorphic multiplication, and consequently there is a corresponding increase in memory demand (as discussed in Sub-section 3.3.1.4 of Chapter 3). The work presented in this chapter was directed at the identification of an encryption scheme that is both fully homomorphic and addresses the cyphertext inflation problem. The fundamental idea was to adapt Liu's existing FHE scheme. The advantage offered was that a far greater range of PPDM algorithms could be considered than the clustering approaches mostly considered so far in this thesis. The chapter focuses on a single data owner scenario as depicted in Figure 1.2-(a).

In summary the work presented in this chapter was directed at providing an answer to the following subsidiary research question, originally posed in Chapter 1:

- Is it possible to tailor (or modify) any proposed HE scheme so that it can support the operations required by a range of data mining algorithms, as opposed to only a small number of specific algorithms?

In more detail, this chapter presents the Modified Liu's Scheme (MLS), a modification of Liu's original FHE scheme that addresses the cyphertext inflation problem and also provides for secure data comparison over cyphertexts without decryption while maintaining the same FHE properties. To address the cyphertext inflation problem a *dimensionality reduction* mechanism was devised that used the *trapdoor* concept. The proposed MLS, as far as the author is aware, is the first FHE scheme that provides an order preserving feature. To illustrate the utility of MLS this chapter also presents the Secure Neural Network (SecureNN) algorithm which uses Back-Propagation (BP) for learning. SecureNN allows model training and query classification to be securely delegated to a TPDM with minimal data owner participation. Using SecureNN, the non-linear activation function is estimated using the low degree polynomial based on Taylor series expansion [195]. For this to operate a limited amount of data owner participation is required.

The remainder of this chapter is organised as follows. Section 9.2 presents MLS and detail concerning the key generation, trapdoor calculation, the associated encryption and decryption algorithms and dimensionality reduction algorithm. In Section 9.3 some preliminaries concerning the Neural Network activation functions and learning methods are briefly presented. Section 9.4 discusses methods for approximating the Sigmoid function using a low degree polynomial founded on Taylor series expansion and using the friendly computation function introduced in [196]. Privacy preserving BPNN, SecureNN, is fully described in Section 9.5. Section 9.6 reports on the evaluation of SecureNN and the experimental results obtained. Finally, Section 9.7 summarises and concludes the chapter.

9.2 Modified Liu's Scheme

The proposed Modified Liu's Scheme (MLS), utilised by the proposed SecureNN approach, is a new scheme that modifies Liu's original FHE scheme presented in [53]. The modifications incorporated into the MLS have two primary objectives:

1. To address the cyphertext inflation problem; the exponential increase in the number of sub-cyphertexts that occurs whenever homomorphic multiplication ($\otimes$) is applied.
2. To provide an order preserving feature in the generated cyphers so as to allow encrypted data comparison in a similar manner to the proposed FDH-OPE and MUOPE schemes presented earlier.

The first is achieved using the concept of *trapdoors*. The second using conditions imposed in key generation coupled with what is referred to as the *ω-concept*, the idea of including a "gap" between sub-cyphertexts so that data ordering is preserved (but not data equality). The original FHE properties of Liu's scheme, that supports addition ($\oplus$), multiplication over cyphertexts ($\otimes$), and multiplication of cyphertexts with plaintexts values ($\circledast$) as discussed in Sub-section 3.3.1.2, were maintained. Also, as in the case of the original Liu's FHE scheme, the proposed MLS does not require any noise management technique and thus there is no limitation on the number of multiplications. The message space and cyphertext space are as defined for the original scheme; $\mathbb{R}$ and $\mathbb{R}^m$ respectively. The following sub-sections, Sub-sections 9.2.1 to 9.2.5, provide detail concerning MLS key generation, trapdoor calculation, data encryption/decryption and the proposed sub-cyphertext dimensionality reduction algorithm.

9.2.1 Key Generation

The same Secret Key (SK) configuration as used in Liu's original FHE scheme is used in MLS, $SK(m)=\Big[(k_1,s_1,t_1), \ldots, (k_m,s_m,t_m)\Big]$. The first step required to generate the secret key is to randomly select real numbers for SK in such a way that the following conditions are satisfied:

1. The number of sub-cyphertexts generated by the MLS (m) is $m \geqslant 3$
2. The k_m, t_m and s_m is selected in such a way that $k_m + t_m + s_m \neq 0$
3. k_i and s_i are positive values ($1 \leqslant i \leqslant m$) and the GCDs (Greatest Common Divisors) for k_i and s_i are > 1 and not equal to s_i or k_i.
4. There exists only one element q ($1 \leqslant q < m$) such that $t_q \neq 0$. This condition was introduced in [21] for facilitating secure data comparison in a secure k-Means data clustering context. In MLS $t_q = (s_q + k_q) \times \omega$, where ω is the numeric gap between cyphertexts included so that ordering is preserved. The ω value adopted with respect to this chapter was 10^ρ selected to create a large gap that permits increasing the number of cyphertexts that can be generated for the same plaintext value.

The values for the sub-keys $K = \{k_1,\ldots,k_{m-1}\}$ and $S = \{s_1,\ldots,s_{m-1}\}$ are split into a "secret key" and "shared key" part. The secret parts of the key are kept locally by the data owner, while the shared parts are used to calculate *trapdoors* that allow the desired sub-cyphertext "dimensionality reduction" (see Sub-sections 9.2.2 and 9.2.5). The list of random numbers $R = \{r_1, \ldots, r_{m-1}\}$, used for encryption purposes together with the secret key (SK), are all positive numbers between 0 and ω. These random values are generated on every occasion the data encryption function is called, and selected in such a way that r_q, corresponding to element q in the SK, is greater than the other generated random values.

9.2.2 Trapdoors Calculation

Trapdoors, as noted above, are used for "reducing" the number of sub-cyphertexts after each homomorphic multiplication ($\otimes$). There is one set of trapdoors, $Trap = \{trap_1, \ldots, trap_m\}$, associated with a single secret key; there is a one-to-one correspondence between the two. The last element of the set $Trap$, $trap_m$, as will be demonstrated later, is of particular significance and is designated as the kst value and is calculated separately; thus for practical purposes $Trap = \{trap_1, \ldots, trap_{m-1}\}$. The process for producing the trapdoor list is given by Algorithm 29. The input is the secret key list (SK(m)) generated as described in Sub-section 9.2.1. The algorithm commences by calculating the "secret key" parts ($secretS$, $secretK$), as the Greatest Common Divisor (GCD) of the sub-keys $S = \{s_1, \ldots, s_{m-1}\}$ and $K = \{k_1, \ldots, k_{m-1}\}$ respectively, to be retained locally by the data owner (lines 2 and 3). The list $Trap$ and shared lists ($SharedS$ and $SharedK$) are then defined in lines 4 and 5 as sets of $m-1$ elements. The set $Trap$ holds the trapdoor values whilst $SharedS$ and $SharedK$ hold the shared part of the secret key used to calculate the trapdoor held in $Trap$. The algorithm then loops from $i = 1$ to $i = m - 1$ (lines 6 to 10) to calculate the shared part of the key, $SharedS$ and $SharedK$, that are then used to calculate the trapdoors as per the equations given in lines 7 to 9. The kst value is calculated in line 11 as the sum of k_m, s_m and t_m. The algorithm exits with the trapdoor list $Trap$ and the kst value (line 12). The set

$Trap$ and kst can be shared with a TPDM so that the TPDM can reduce the number of sub-cyphertexts after each homomorphic multiplication as described in Sub-section 9.2.5 below.

Algorithm 29 MLS trapdoor calculation

1: **procedure** TRAPDOORSCALCULATION($SK(m)$)
2: $secretS$= GCD($s_1, \ldots, s_{m-1}$)
3: $secretK$= GCD($k_1, \ldots, k_{m-1}$)
4: `Declare` $Trap$ `as a set of` $m-1$ `elements to hold trapdoor values`
5: `Declare` $SharedS$ and $SharedK$ `as a set of` $m-1$ `elements`
6: **for** $i = 1$ to $i = m-1$ **do**
7: $sharedS_i = \frac{s_i}{secretS}$
8: $sharedK_i = \frac{k_i}{secretK}$
9: $trap_i = \frac{sharedS_i}{sharedK_i}$
10: **end for**
11: $kst = k_m + s_m + t_m$
12: **Exit** with $Trap$ and kst
13: **end procedure**

9.2.3 Encryption

The MLS encryption function uses $SK(m)$ to convert a value v to m sub-cyphertexts $E = \{e_1, \ldots, e_m\}$. The MLS encryption is given in Algorithm 30. The inputs are the value to be encrypted v and secret key list $SK(m)$. The algorithm commences by generating a random number list $R = \{r_1, \ldots, r_{m-1}\}$ in such a way that the r_q is bigger than the remaining random numbers (as discussed with respect to the key generation process in Sub-section 9.2.1). The cyphertext (E) is then dimensioned in line 3 as a set of m elements, $E = \{e_1, \ldots, e_m\}$, that are calculated in lines 4 to 8. The variable l, associated with each cyphertext, is then initialised (line 9). This variable is the cypher level counter, the number of times that dimensionality reduction has been applied to the cyphertext. There is no limit for the number of levels supported by the MLS, however, the value of l is required for decryption purposes (see Sub-section 9.2.4 below).

The MLS encryption function associated with the conditions defined by the key generation conditions presented in Sub-section 9.2.1 preserves the order of the plaintext value. The proof of correctness for these features are given in Appendix A. The feature of preserving the data ordering in the qth sub-cyphertext (e_q) can be used to calculate the absolute value of the cyphertexts, when the cypher is negative this can be derived by first comparing the e_q with the qth cyphertext of zero and then multiplying the cyphertext by (-1) using the $\circledast$ HE property, as included in the Liu's original FHE scheme (see Sub-section 3.3.1.2).

9.2.4 Decryption

The decryption function decodes cyphertext E to its plaintext equivalent v, following the process shown by the pseudo code given in Algorithm 31. The inputs are: the cyphertext to be decrypted E, the secret key list $SK(m)$ and secret part of the sub-keys $secretS$ and $secretK$. If the level number ($E.l$) is not zero, the dimensionality of the sub-cyphertext has been reduced. The algorithm starts by calculating a new sub-cyphertext value for each sub-cypher e_i in E (lines 2 to 6). The algorithm then calculates t and

Algorithm 30 MLS encryption

1: **procedure** ENCRYPT($v, SK(m)$)
2: `Uniformly generate m-1 arbitrarily random numbers` $R = \{r_1, ..., r_{m-1}\}$
3: `Declare` E `as a set of` m `elements`
4: $e_1 = \frac{(k_1 \times t_1 \times v + s_1 + k_1 \times (r_1 - r_{m-1}))}{s_1}$
5: **for** $i = 2$ to $m - 1$ **do**
6: $e_i = \frac{(k_i \times t_i \times v + s_i + k_i \times (r_i - r_{i-1}))}{s_i}$
7: **end for**
8: $e_m = k_m + s_m + t_m$
9: $E.l = 0$
10: **Exit** with E
11: **end procedure**

s in lines 7 and 8 that are used in line 9 with the $SK(m)$ to calculate the decoded value v. The algorithm will exit in line 10 with the decoded value v. As the MLS modifies Liu's original FHE scheme encryption and decryption algorithm to facilitate the dimensionality reduction and order preserving feature, a mathematical proof for the correctness of the decryption algorithm is provided in Appendix A.

Algorithm 31 MLS decryption

1: **procedure** DECRYPT($E, SK(m), secretS, secretK$)
2: **if** $E.l \neq 0$ **then**
3: **for** $i = 1$ to $i = m$ **do**
4: $e_i = \frac{(e_i \times (secretS^{E.l} / secretK^{E.l}))}{t^{E.l}}$
5: **end for**
6: **end if**
7: $t = \sum_{i=1}^{m-1} t_i$
8: $s = \frac{e_m}{(k_m + s_m + t_m)}$
9: $v = \frac{(\sum_{i=1}^{m-1}((e_i \times s_i) - (s \times s_i))/k_i)}{t}$
10: **Exit** with v
11: **end procedure**

9.2.5 Sub-cyphertexts Dimensionality Reduction

MLS, as in the case of Liu's scheme, performs cyphertext multiplication ($\otimes$) by determining the outer product of the two cyphertexts to be multiplied. Given two plaintext values v_1 and v_2, which are encrypted using MLS with the $SK(m)$ to give $E_1 = \{e_{1_1}, \ldots e_{1_m}\}$ and $E_2 = \{e_{2_1}, \ldots, e_{2_m}\}$ respectively; the cyphertext multiplication, $E_1 \otimes E_2$, is implemented as: $\{e_{1_1}, \ldots, e_{1_m}\} \otimes \{e_{2_1}, \ldots, e_{2_m}\} = \{e_{1_1} \times e_{2_1}, \ldots, e_{1_1} \times e_{2_m}, \ldots e_{1_m} \times e_{2_1}, \ldots, e_{1_m} \times e_{2_m}\}$. Therefore, for one multiplication, the number of sub-cyphertext is increased from m to m^2 and continues to exponentially increase with each multiplication operation. This cyphertext inflation, as noted earlier, causes a computational overhead and also leads to a scalability issue. Using the proposed MLS the number of the generated sub-cyphertext, after a multiplication operation has been applied, is "reduced" back to m using trapdoor information that allows re-encryption of the cyphertext (without prior decryption); this is referred to, in this thesis, as "dimensionality reduction".

Algorithm 32 presents the pseudo code for the dimensionality reduction process. The algorithm takes as inputs: (i) a sequence of sub-cyphers $E = \{e_1, \ldots, e_{m^2}\}$, (ii) a set of

trapdoors $Trap$ and (iii) the kst value. The algorithm commences (line 2) by declaring a reduced cyphertext set RE of length m. Next, an index j for the cyphertext set E and index z for reduced cyphertexts RE are declared (line 3). The algorithm then loops through E (lines 4 to 15). Each iteration commences (line 5) with the creation of a temporary cypher, $Temp$, made up of the consecutive m sub-cyphers in E starting with index j. The mth sub-cypher in $Temp$, $temp_m$, and the trapdoor value kst are used to calculate the value for the parameter s (line 6). The algorithm then (line 7) defines the variable $subCypher$ in which to hold the current sub-cyphertext value once calculated. Next the algorithm loops through $Temp$ (lines 8 to 11) and determines the new sub-cyphertext value and, on completion, assigns the value to the zth element of RE which holds the cyphertexts as calculated so far. The values of index z and j are then updated in lines 13 and 14. The loop continues until the set RE is calculated. In line 16, the cypher level counter is incremented by one, $E.l + 1$. At the end of the process the newly calculated cyphertext, of length m, is returned (line 17). In the remainder of this chapter the multiplication of two cyphertexts, followed by dimensionality reduction, is indicated using the operator $\bigotimes$. Recall that the operator $\circledast$ is used to refer to multiplying a cyphertext with a plaintext value.

Algorithm 32 Dimensionality reduction process

```
1: procedure DimensionalityReduction(E, Trap, kst)
2:     RE set of m elements to hold a re-encrypted cyphertexts
3:     j = 1, z = 1
4:     while j < |E| do
5:         Temp = Copy sub-cyphertext in E started by jth index of length
   m
6:         s = temp_m / kst (temp_m ∈ Temp)
7:         subCypher = 0
8:         for i = 1 to i = m − 1 do
9:             t = temp_i − s ⊛ trap_i (trap_i ∈ Trap)
10:            subCypher = subCypher + t
11:        end for
12:        re_z = subCypher (re_z ∈ RE)
13:        z = z + 1
14:        j = j + m
15:    end while
16:    RE.l = E.l + 1
17:    Exit with RE
18: end procedure
```

The correctness of the sub-cyphertext dimensionality reduction algorithm given in Algorithm 32 is evaluated by decrypting the cyphertext result using the decryption algorithm (Algorithm 31). The results of decryption have to match the result of the basic multiplication operator ($\times$). The mathematical proof of the correctness of the sub-cyphertext dimensionality reduction process is provided in Appendix A.

9.3 Neural Network Preliminaries: Activation Functions and Learning Methods

The cyphertext inflation problem, addressed using MLS, is not a significant issue with respect to the PPDM processes described in this thesis so far where the processes considered have been restricted to data clustering and on one occasion Nearest Neighbour classification. For example in the case of the Secure k-Means clustering algorithm described earlier, division (multiplication by a fraction) was used to determine the cluster centroids, but this did not present a restrictive overhead. It would of course be preferable if the encryption techniques presented in this thesis could also be applied to a more sophisticated range of PPDM activities. A good exemplar of such a PPDM activity is secure Neural Network (NN) training and utilisation. Therefore the remainder of this chapter is devoted to how the MLS presented in the previous section can be used to achieve secure NN. This section looks at some necessary preliminaries concerning NNs; so as facilitate a complete understanding. The next section considers the various elements required to realise secure NN using the MLS. The following section presents the SecureNN algorithm, which is then evaluated in the next section.

An Artificial Neural Network (ANN) comprises several neurons that can be connected in different ways. The multiple-layer feed-forward NN orders the neurons into different layers. The first layer is the input layer and the last layer is the output layer; the layers between are the hidden layers. Each neuron has a threshold coefficient *bias* and a *weighted* connection to all neurons in the next layer. The output of the ith neuron is determined as per Equation 9.1 and Equation 9.2 where w_{ji} is the weight of the connection between the ith and jth neuron and p is the number of neurons in the predecessor layer where the ith neuron is located.

$$y_i = f(x_i) \tag{9.1}$$

$$x_i = \theta_i + (\sum_{j=1}^{j=p} w_{ji} \times y_j) \tag{9.2}$$

The function f in Equation 9.1 is the activation function; a mathematical equation used to determine whether a neuron will be activated ("fired") or not, based on the relevance of the neuron's input for the model prediction purposes. There are several activation functions that can be used. Popular activation functions include: (i) Binary Step, (ii) Hyperbolic Tangent (Tanh), (iii) Rectified Linear Unit (ReLu) and (iv) the Sigmoid function. These are summarised in Figure 9.1.

Regardless of the specific activation function adopted, the weights and biases for the activation function, associated with each neuron in the hidden layers and the output layer, is derived using a learning (training) process. The adopted learning process varies the weights and biases associated with an activation function so as to maximise query classification effectiveness. The most commonly encountered learning process is the Back Propagation (BP) algorithm [197], and this was therefore the learning process adopted with respect to the SecureNN process described in more detail later in this chapter. BP is a supervised learning method that varies the weights and biases to minimise the error according to the difference between NN prediction (the output of the last layer) and the target output. Gradient decent is used to propagate the prediction error back from the output layer of the NN model, through the different neurons which were involved in generating that output, back to the input. During this process the original weights and biases associated with each neuron are adjusted. The BP learning may proceed in one of

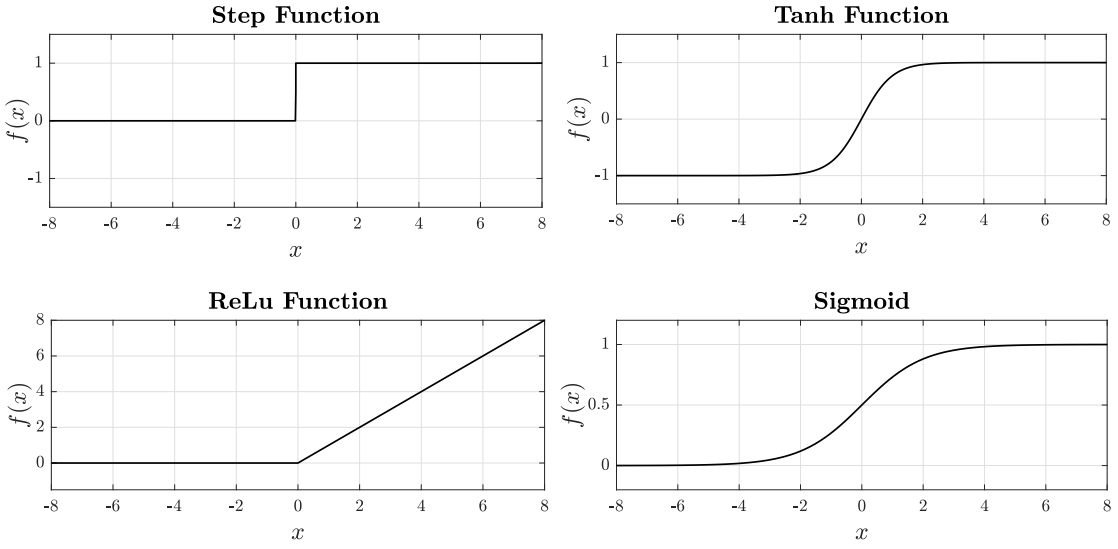


FIGURE 9.1: Popular activation functions

two basic ways: pattern mode or batch mode. In pattern mode BP weight updating is performed after the presentation of each training example. In the batch mode of the BP weight updating is performed after the presentation of all the training examples (thus after a whole "epoch"). With respect to the proposed SecureNN process, described in detail later in this chapter, pattern mode was used. Note that the gradient decent requires the activation function to be differentiable (inconstant). Of the four example activation functions listed earlier Binary Step activation is not differentiable. Of the remainder, the Sigmoid activation is the most frequently adopted and was therefore the activation function adopted with respect to the proposed SecureNN process presented in this chapter. On completion of the training the NN can be used to assign class labels to previously unseen examples. The derived activation functions collectively determine the output of the ANN model as indicated by Equation 9.2.

9.4 Approximation of Sigmoid Activation Function

The Sigmoid activation function, given in Equation 9.3 and plotted in Figure 9.1, is a non-linear function that cannot be *directly* calculated using the mathematical properties of FHE schemes [198]. The reader may find it useful to refer back to Sub-section 3.2.4 of Chapter 3 where the HE limitations were discussed. The operation of the Sigmoid activation function can be *approximated*, up to a certain accuracy, using a polynomial approximation method that uses Taylor series expansions [195] or Chebyshev polynomials [159]. Essentially, the idea of approximating the Sigmoid activation function, as argued in the literature [195], was to improve the efficiency of training NNs. This polynomial approximation can be theoretically implemented using the HE properties. In practice, this approximation needs to be done using a high degree polynomial for accurate results to be obtained, which in turn increases the extensive amount of HE multiplication that in turn increases the amount of noise and the size of cyphertexts.

$$f(x) = Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{9.3}$$

In the following sub-sections two Sigmoid approximations are discussed. The first is the *TaylorLinear*; the privacy preserving approximation of the Sigmoid activation

function using Taylor series expansion. The second is the *FriendlyFunction*, introduced in [196], that uses piecewise-linear approximation. With respect to the proposed SecureNN, *TaylorLinear* approximation was used for training the SecureNN, with limited data owner participation, and the *FriendlyFunction* to provide query classification (prediction) services once the NN had been trained.

9.4.1 Sigmoid Approximation Using Taylor Series Expansion (*TaylorLinear*)

The Taylor series expansion is used to linearly approximate the e^{-x} term, which is a part of the Sigmoid activation function given in Equation 9.3. The e^{-x} term can be approximated as per Equation 9.4 where d is the degree of polynomial selected by the data owner according to the required accuracy set against execution time.

$$e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \cdots + \frac{(-1)^d x^d}{d!} \tag{9.4}$$

Using these parameters the *TaylorLinear* approximates Sigmoid(x'), where x' is encrypted using MLS, as follows:

1. Using the MLS homomorphic properties, the TPDM calculates the approximation of $1 + e^{|x'|}$ using the Taylor polynomial given in Equation 9.5 where d represents the maximum degree of the function. A range of d values were experimented with $d = 3, 5$, and 7, the results are presented later in this chapter.

2. The data owner performs the "inversion" of the $1 + e^{|x'|}$ value to arrive at the approximated value $\frac{1}{1+e^{|x'|}}$.

3. If $x' \geqslant 0$ the TPDM will calculate the activation function $\frac{1}{1+e^{-x'}}$ as $1 - \frac{1}{1+e^{|x'|}}$. Otherwise the activation function is as approximated in step 2.

$$1+e^{x'} = 1 \oplus (1 \oplus x' \oplus \frac{1}{2!} \circledast (x' \bigotimes x') \oplus \frac{1}{3!} \circledast (x' \bigotimes x' \bigotimes x') \oplus \cdots \oplus \frac{1}{d!} \circledast (x' \bigotimes \ldots\ldots \bigotimes x')) \tag{9.5}$$

The absolute value of x', $|x'|$, used in step 1 is calculated by multiplying x' with -1, using $\circledast$, when cypher x' is less than the MLS cypher of zero. The comparison of x' with zero (in step 3) is conducted, using the MLS properties, by comparing the qth cypher of x' with qth sub-cypher of zero encrypted. Step 3 also relies on a mathematical rule associated with the S function that allows the calculation of Sigmoid(x) and Sigmoid(-x) as per Equation 9.6.

$$Sigmoid(-x) = 1 - Sigmoid(x) \tag{9.6}$$

From the above, *TaylorLinear* requires some data owner participation (step 2) but this participation is minimal compared with alternative approaches used to approximating Sigmoid, such as those given in [112] and [99], or approximating the ReLU as given in [199]. The accuracy of using the *TaylorLinear* was evaluated by calculating the error associated with the estimations, calculated as the difference between the Sigmoid function and the estimated functions; this is reported on later in the chapter. Figure 9.2 shows a comparison of Sigmoid activation and its approximation using *TaylorLinear* with a range of values for d, $d = \{3, 5, 7\}$, and different value function input x; from the figure the magnitude of error can be observed in each case. The symbol φ^d is used to

refer to the *TaylorLinear* approximation function where d is the degree of polynomial. The figure also shows the error (δ) associated with the estimations. The experiments show that the error of *TaylorLinear* when $d = 3$ is $\delta \in [-0.024, 0.024]$, greater than when $d = 5$ and $d = 7$, $\delta \in [-0.0048, 0.0048]$ and $\delta \in [-0.0010, 0.0010]$ respectively.

9.4.2 Friendly Activation Functions (*FriendlyFunction*)

A secure activation function is also required in the context of the provision of query classification/prediction services. *TaylorLinear* can again be used for this purpose. Alternatively, the *FriendlyFunction* piecewise-linear approximation [196] may be used, which will return 0 when $x < -0.5$, 1 when $x > 0.5$ and $x + 0.5$ when $-0.5 \leqslant x \leqslant 0.5$. This offers the advantage, using the proposed MLS, that it can operate over encrypted data without any data owner participation. However, it is not as accurate as *TaylorLinear*. This is illustrated in Figure 9.3 which provides a comparison of different values of the activation function input x, using the *FriendlyFunction* and the standard Sigmoid activation function, and also shows the amount of error, δ, calculated as the difference between the Sigmoid function and the *FriendlyFunction* estimation. The experiments demonstrated that the *FriendlyFunction* provides the worst case; $\delta \in [-0.40, 0.40]$.

From the foregoing it can therefore be concluded that *TaylorLinear* approximation provides a better fit with the Sigmoid function than *FriendlyFunction* approximation, but requires some undesirable data owner participation; whilst the *FriendlyFunction* does not provide as good an approximation, but requires no data owner participation.

9.5 Privacy Preserving Back-Propagation NN

The proposed SecureNN approach comprises a multi-layer feed-forward network with BP learning which is both trained and used over encrypted data. The scenario considered with respect to this chapter is the single data owner scenario where the data that used to develop SecureNN belongs to a single data owner and only the data owner is allowed to use (query) the model. In feed-forward neural networks, as noted earlier in Section 9.3, each neuron has a weighted connection to all neurons in the next layer. Neurons in different layers are of different types. For example, neurons in the input layer are distinguished by one input and one output (which is the same value). Neurons in the hidden layers are more complex; they receive multiple inputs, compute the weighted summation of these inputs, operate an activation function on the summation and then output the value of the function. For the proposed SecureNN this function is a *TaylorLinear* function. The proposed privacy preserving BP training mechanism is given in Algorithm 33; the notation used is presented in Table 9.1. Recall that the operators $\bigotimes$, $\otimes$, $\circledast$, $\oplus$ and $\ominus$ indicate cyphertexts multiplication coupled with dimensionality reduction, cyphertexts multiplication without dimensionality reduction, multiplication of a cyphertext with a plaintext value, cyphertext addition and cyphertext subtraction, respectively. The inputs are:

(i) a set of n training examples (records), $D' = \{r'_1, \ldots, r'_n\}$, where each record r'_i features a set of a attributes $\{r'_{i,1}, \ldots, r'_{i,a}\}$;

(ii) a maximum number of epochs, $maxEpoch$;

(iii) a learning rate η;

(iv) a momentum μ;

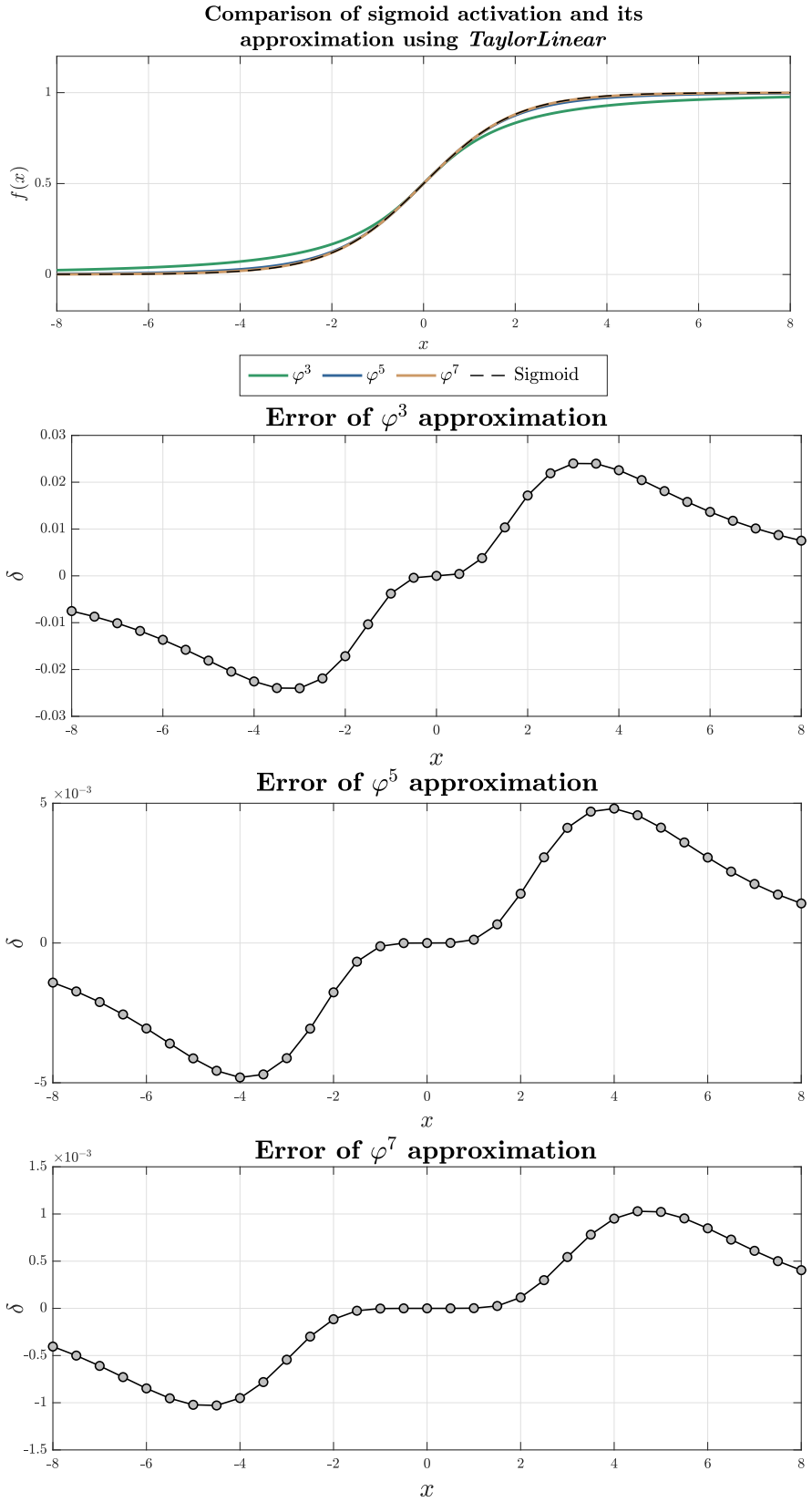


FIGURE 9.2: Comparison of the *TaylorLinear* (φ) approximations, and their error of estimation, with the Sigmoid activation function

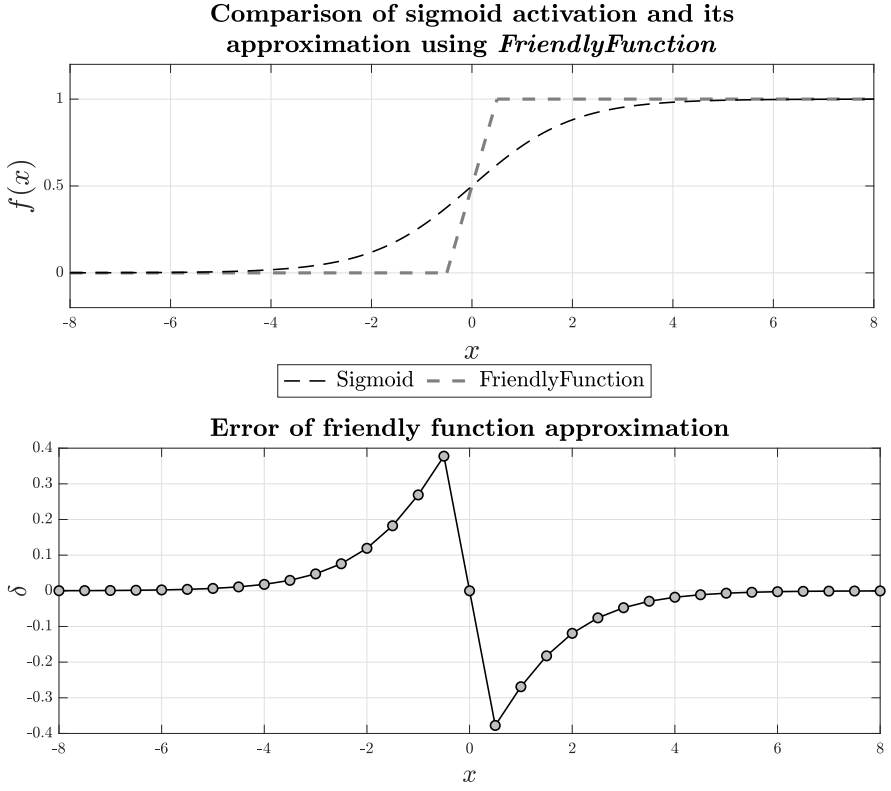


FIGURE 9.3: Comparison of the *FriendlyFunction* approximations, and their error of estimation, with the Sigmoid activation function

(v) a set of n records that represent the target class labels for the training examples, $T' = \{t'_1, \ldots, t'_n\}$, each featuring c attributes $t'_i = \{t'_{i,1}, \ldots, t'_{i,c}\}$ where c is the number of classes, in such a way that only one element in t'_i is 1 which represents the target class label;

(vi) an input-hidden-output network topology, $L = \{l_1, \ldots, l_b\}$, defined by the data owner, where $|L| = b$ indicates the number of layers, $\sum_{i=1}^{i=b} l_i$ indicates the total number of neurons, and l_i indicates the number of neurons in the ith layer;

(vii) an error threshold ϵ'; and

(viii) the d value for the *TaylorLinear* approximation (φ^d).

The training data D', set of class labels T' and the error threshold ϵ' are all encrypted using the proposed MLS. The output is a set of weights W' and a set of biases Θ', for the network described by L, encrypted using MLS.

As in the case of standard BP learning [197], the proposed privacy preserving BP learning is comprised of two stages: (i) feed-forward (lines 8 to 15 of Algorithm 33) and (ii) error BP (lines 16 to 34). The algorithm commences, line 2, by defining the sets W' and Θ' and initialising them with random values encrypted using MLS; and then, line 3, defining the sets $\Delta W'$ and $\Delta\Theta'$ and initialising them with the value 0 encrypted using MLS. The overall error value so far (the overall loss function), $overallError'$, is initialised with the MLS encrypted equivalent of 0 (line 4). The variable $oneE'$ is assigned the MLS encrypted equivalent of 1 (line 5). The training is then commenced (line 6), the algorithm iterates until the specified maximum number of epochs is reached ($maxEpoch$), or the $overallError'$ value becomes less than ϵ'. On each iteration each sample (record) in D' is processed in turn.

Table 9.1: Notation used in Algorithm 33

Symbol	Definition
$r'_{i,j}$	The encrypted values of the jth attribute in the ith record of the encrypted training set D'.
$t'_{i,j}$	The encrypted target class value of the jth neuron in the output layer for the ith encrypted data sample $r'_i \in D'$.
${y'}_i^j$	The encrypted output of the ith neuron in the jth layer.
Θ'	The set of encrypted bias values $\{{\theta'}_1^1, {\theta'}_2^1, \dots\}$.
${\theta'}_i^j$	The encrypted bias value of the ith neuron in the jth layer.
W'	The set of encrypted weight values $\{{w'}_1^1, {w'}_2^1 \dots\}$.
${w'}_{x\ y}^{i}$	The encrypted weight connecting the xth neuron in the ith layer with the yth neuron in the following layer ($i+1$th layer).
${\delta'}_i^j$	The encrypted error value corresponding to the ith neuron in the jth layer.
$\Delta W'$	The set of encrypted weight differences in the network, one to one correspondence with W'.
$\Delta {w'}_{x\ y}^{i}$	The encrypted value of change in weight that connects the xth neuron in the ith layer with the yth neuron in the following layer (the $i+1$th layer).
$\Delta\Theta'$	The set of encrypted bias differences in the network, one to one correspondence with Θ'.
$\Delta {\theta'}_i^j$	The encrypted value of change in bias for the ith neuron in the jth layer.

During the feed forward stage the output for the input layer neurons are matched to the attribute values in r'_{sample} (lines 8 to 10 of Algorithm 33); a is the number of attributes in the attribute set (the number of values in each record/sample and thus the number of neurons in input layer). The remaining layers, the hidden layers and the output layer, are then processed in lines 11 to 15 and outputs assigned to the neurons; b is the number of layers in the topology as specified by the data owner. The outputs of the neurons in the hidden layers are calculated by multiplying the output of neurons in the previous layer with the weights connecting the two layer neurons and then adding the value of the neuron bias. The results are used as input for the *TaylorLinear* approximation of the Sigmoid function with degree d, φ^d, line 13.

Once the feed forward stage is completed the BP is commenced. In the BP stage, the network weights and biases are adjusted to minimise the error function. With respect to the proposed SecureNN the BP used *pattern mode*, or what is also sometimes referred to as the *online method*, where the weights and bias updates are applied after the presentation of each training sample. The gradient descent ($\frac{\partial Error}{\partial y}$), the derivation of the error with respect to the NN output (prediction), for each neuron in each layer is calculated starting with the output layer and moving backwards to the input layer following a process similar to the BP process presented in [197]. However, the mathematical operations are replaced with homomorphic equivalent operation using the properties of the proposed MLS. The calculated gradient is then used to update weights and biases. The process is commenced with the output layer in lines 16 to 19 of Algorithm 33, by calculating the gradients δ; c is the number of classes featured in the input data that also matches the number of neurons in the output layer. Next, lines 20 to 24, the gradient of the intermediate neurons are calculated, starting with the layer immediately preceding

the output layer; l_j is the number of neurons in the jth layer as specified by the data owner in network topology L. The gradients are then used to update the weights and biases as per the equations in lines 27 and 28 for updating biases, and the equations in lines 30 and 31 for updating the weights. For SecureNN, the momentum μ is used to accelerate the learning process. As the iterative process of incremental adjustment continues the weights and biases will gradually converge to the a locally optimal set of values that minimises the loss function (error); in the best case scenario globally optimal values will be reached.

Once a single record/sample has been processed the associated error is determined (line 35) which is then included in the overall error so far (line 36). The difference between the NN predictions and target labels T, the loss or error function, for each training sample is calculated using Equation 9.7; where n is the number of data samples in D'. Once an epoch has been completed an end of epoch overall error is calculated (line 38) which is compared to the threshold ϵ', if this is less than the threshold the algorithm completes with the current set of weights and biases. Otherwise the process repeats.

$$overallError = \frac{1}{2 \times n} \sum_{s=1}^{s=n} \sum_{i=1}^{i=c} (y_i - t_i)^2 \tag{9.7}$$

As noted in the introduction of this chapter the SecureNN is directed to a single data owner scenario where one data owner outsources their data. In the proposed SecureNN the query prediction/classification service is provided to the same data owner who owns the MLS encryption key and thus the QO in this case is also the data owner. As a consequence, there is no need for QO authorisation or query control. Algorithm 34 shows the Query prediction service process. The inputs are the query record ($QRec$) belonging to the data owner, the approximation function to be used and the degree of polynomial d used to approximate the $TaylorLinear$. The algorithm commences with the data owner pre-processing and encrypting the $QRec$ using the MLS key used to encrypt the data to give $QRec'$ (lines 2 and 3). The TPDM is then fed-forward through the NN using the encrypted $QRec'$ as specified in lines 8 to 15 of Algorithm 33. The activation function to be adopted is as specified by the data owner in the $ApproximationFunction$ (lines 4 and 5). The TPDM then determines the class label of the Query record from the output layer neurons and sends the result back to the data owner (lines 6 and 7).

9.6 Experimental Results

This section reports on the analysis and evaluation of the proposed MLS encryption and the SecureNN process. For the evaluation, two categories of datasets were used; synthetic datasets and benchmark datasets taken from the UCI data repository [50]. The synthetic datasets were used to evaluate the performance of the proposed MLS while the UCI datasets were used to evaluate the SecureNN process. The UCI dataset were selected so that a variety of sizes (number of records and number of attributes) and number of classes could be considered. Both MLS and SecureNN were implemented in the Java programming language. All experiments were run on an iMac (3.8 GHz Intel Core i5) running under the macOS High Sierra operating system with 8GB of RAM, and conducted using TCV; the results presented in the following sub-sections are therefore average values. Table 9.2 listed the UCI datasets used, the number of samples (records) in each dataset, the network (input-hidden-output layer) topology, the learning rate η

Algorithm 33 Secure NN Training

procedure SECURETRAINING(D',$maxEpoch$,η,μ,T',L,ϵ',d)
  `Initialise` W' `and` Θ' `randomly and encrypt values using MLS`
  `Initialise` $\Delta W'$= Encrypt(0) `and` $\Delta\Theta'$=Encrypt(0)
  $overallError'$= Encrypt(0) ▷ Algorithm 30
  $oneE'$= Encrypt(1)
  for $epoch = 1$ to $epoch = maxEpoch$ **do**
    for $sample = 1$ to $sample = n$ **do**
      for $i = 1$ to $i = a$ **do**
        ${y'}_i^1 = r'_{sample,i}$ (where $r'_{sample} \in D'$)
      end for
      for $j = 2$ to $j = b$ **do**
        for $i = 1$ to $i = l_j$ **do**
          ${y'}_i^j = \varphi^d(({w'}_{1\,i}^{j-1} \bigotimes {y'}_1^{j-1} \oplus \cdots \oplus {w'}_{l_{j-1}\,i}^{j-1} \bigotimes {y'}_{l_{j-1}}^{j-1}) \oplus {\theta'}_i^j)$
        end for
      end for
      for $att = 1$ to $att = c$ **do**
        $e' = ({y'}_{att}^b \ominus t'_{sample,att})$ where $t'_{sample} \in T'$
        ${\delta'}_{att}^b = e' \bigotimes ({y'}_{att}^b \bigotimes (oneE' \ominus {y'}_{att}^b))$
      end for
      for $j = b - 1$ to $j = 2$ **do**
        for $i = 1$ to $i = l_j$ **do**
          ${\delta'}_i^j = {y'}_i^j \bigotimes (oneE' \ominus {y'}_i^j) \bigotimes [({w'}_{i\,1}^j \bigotimes {\delta'}_1^{j+1}) \oplus \cdots \oplus ({w'}_{i\,l_{j+1}}^j \bigotimes {\delta'}_{l_{j+1}}^{j+1})]$
        end for
      end for
      for $j = b$ to $j = 2$ **do**
        for $i = 1$ to $i = l_j$ **do**
          ${\Delta\theta'}_i^j = (\eta \circledast {\delta'}_i^j) \oplus (\mu \circledast {\Delta\theta'}_i^j)$
          ${\theta'}_i^j = {\theta'}_i^j \oplus {\Delta\theta'}_i^j$
          for $k = 1$ to $k = l_{j-1}$ **do**
            ${\Delta w'}_{i\,k}^j = (\eta \circledast {\delta'}_i^j \bigotimes {y'}_k^{j-1}) \oplus (\mu \circledast {\Delta w'}_{i\,k}^j)$
            ${w'}_{i\,k}^j = {w'}_{i\,k}^j \oplus {\Delta w'}_{i\,k}^j$
          end for
        end for
      end for
      $Error' = [({y'}_1^b \ominus t'_{sample,1}) \bigotimes ({y'}_1^b \ominus t'_{sample,1})] \oplus \cdots \oplus [({y'}_c^b \ominus t'_{sample,c}) \bigotimes ({y'}_c^b \ominus t'_{sample,c})]$
      $overallError' = overallError' \oplus Error'$
    end for
    $overallError' = \dfrac{1}{2 \times n} \circledast overallError'$
    if $overallError' < \epsilon'$ **then**
      Exit with W' and Θ'
    end if
  end for
  Exit with W' and Θ'
end procedure

Algorithm 34 Secure query classification process using SecureNN

1: **procedure** QUERYPREDICTION($QRec$, $ApproximationFunction$, d)
2: **Data owner:** pre-process $QRec$
3: **Data owner:** $QRec'$= Encrypt ($QRec$) ▷ Algorithm 30
4: **TPDM:** The activation function ← as specified in $ApproximationFunction$.
5: **TPDM:** Feed-forward trained NN using $QRec'$. ▷ Algorithm 33
6: **TPDM:** $PredictedLabel$= Determine the predicted class label by comparing cyphertexts output of the output layer.
7: **Exit** with $PredictedLabel$
8: **end procedure**

and the momentum μ parameter settings that were used for the experimentations. The number of epochs was fixed at $MaxEpoch = 100$ in all cases, and the NN weights and biases were initialised randomly from the range $[-0.4, 0.4]$ which were then encrypted using MLS. In practice these parameter settings would be pre-defined by the data owner. The datasets were normalised using *Minmax* data normalisation. The SecureNN was trained using the *TaylorLinear* $d = 3$ and φ^3, because this was the poorest approximation as depicted in Figure 9.2. In the prediction stage the φ^3 and *FriendlyFunction* were used.

In the following sub-sections the results of the experimental evaluation are presented. The objectives of the evaluation were as follows:

1. **MLS performance**: To evaluate the performance of MLS encryption by measuring the runtimes required for: (i) MLS key generation; (ii) MLS data encryption; (iii) MLS data decryption; (iv) MLS homomorphic addition ($\oplus$); multiplication with plaintext ($\circledast$) and multiplication with cyphertext ($\bigotimes$); and (v) MLS data comparison. The results are presented and discussed in Sub-section 9.6.1.

2. **SecureNN evaluation**: To evaluate the operation of SecureNN in terms of: (i) data owner participation, (ii) the computational overhead of SecureNN in comparison with standard NN (SecureNN efficiency), (iii) effectiveness (query classification/prediction accuracy) and (iv) security. Each is discussed in further detail in Sub-sections 9.6.2 to 9.6.5.

TABLE 9.2: Experiment datasets and neural network parameters

No. UCI Dataset	Num. of Sample	Topology $\{input, hidden, output\}$	η	μ
1. Banknote Auth.	1372	{4, 5, 2}	0.01	0.7
2. Blood Trans.	748	{4, 5, 2}	0.02	0.8
3. Breast Cancer	198	{33, 10, 2}	0.20	0.9
4. Breast Tissue	106	{9, 6, 6}	0.20	0.9
5. Chronic Kidney	400	{24, 5, 2}	0.20	0.9
6. Dermatology	366	{34, 5, 6}	0.20	0.8
7. Ecoli	336	{8, 5, 8}	0.30	0.8
8. Iris	150	{4, 5, 3}	0.20	0.7
9. Libras Mov.	360	{90, 10, 15}	0.30	0.5
10. Parkinsons	195	{22, 5, 2}	0.30	0.9
11. Pima Disease	768	{8, 5, 2}	0.20	0.9
12. Seeds	210	{7, 5, 3}	0.20	0.9

9.6.1 MLS Performance Evaluation

In this sub-section the evaluation of the proposed MLS is presented. MLS was evaluated by analysing the performance of the various supported MLS operations and, where appropriate, comparing this performance with the performance using Liu's original FHE scheme presented in Sub-section 3.3.1. Performance was measured in terms of the runtime required to: (i) generate the MLS key, (ii) encrypt data, (iii) decrypt data, (iv) utilise the FHE mathematical properties ($\oplus$,$\bigotimes$,$\circledast$) and (v) secure comparison. In the experiments the number of sub-cyphertexts (m) considered was $m = \{3, 9, 15\}$. The recorded runtimes to generate the MLS key were $1.16ms$, $1.37ms$ and $1.44ms$ for $m = 3, 9$ and 15 respectively. These results demonstrated that the runtimes for generating the MLS keys increased with number of sub-cyphertexts m, the number of elements in the secret key list $SK(m)$, this was to be expected.

The performance associated with: MLS encryption and decryption, the HE mathematical properties ($\oplus$,$\bigotimes$ and $\circledast$) and order preserving properties were also measured in terms of runtimes to perform the operations in the context of different sizes of data records and the different numbers of sub-cyphertexts featured in MLS. The results were shown in Figure 9.4. As in the case of Liu's original FHE scheme, data encryption, decryption and the homomorphic operations feature "linear" processing time in relation to the size of the data and number of sub-cyphertexts m. However, the runtimes were negligible; using MLS a record with $1,000$ attributes can be encrypted in $0.85ms$ when $m = 15$, and decrypted in $0.52ms$. The HE mathematical properties ($\oplus$,$\bigotimes$ and $\circledast$) were more expensive, in terms of runtime, than encryption and decryption although the multiplication runtime was much higher than the addition because of dimensionality reduction. The runtime associated with the HE mathematical operations increased with number of attribute featured in the data and the number of sub-cyphertexts in the MLS. However, the times reported, as shown in Figure 9.4, were again negligible. The secure comparison of two MLS cyphertexts can be achieved by comparing the qth sub-cyphertexts, therefore, regardless of the number of sub-cyphertexts (the value of m) the recorded data comparison runtime was steady at $0.2ms$.

The performance of MLS was also compared to the Liu's original FHE scheme. The comparison considered the runtime required for the two schemes to: (i) generate the key, (ii) perform encryption and decryption considering datasets that featured different numbers of attributes and different numbers of sub-cyphertext and (iii) perform the HE mathematical properties supported by both schemes. The results for the Liu's original FHE scheme were presented in Sub-section 3.3.1.4. The performance for key generation, encryption, decryption and additions were comparable to that reported for MLS. However, the runtimes for MLS multiplication that included dimensionality reduction ($\bigotimes$) were much higher than the original Liu's FHE scheme multiplication $\otimes$. The difference is the cost of the MLS feature to reduce the number of Sub-cyphertexts from m^2 to m.

9.6.2 Data Owner Participation

This sub-section considers the amount of data owner participation required to: (i) prepare the data prior to network generation and (ii) train the network using the proposed SecureNN framework. The results for data owner preparation are presented in Table 9.3. The data preparation comprised: Minmax data normalisation (column 2); data encryption, excluding MLS key generation and trapdoors calculation (column 3); and preparation of the training and testing samples to facilitate stratified TCV (column 4). Inspection of the table shows that the data owner participation in preparing data for network generation was negligible and did not introduce any overhead on behalf of

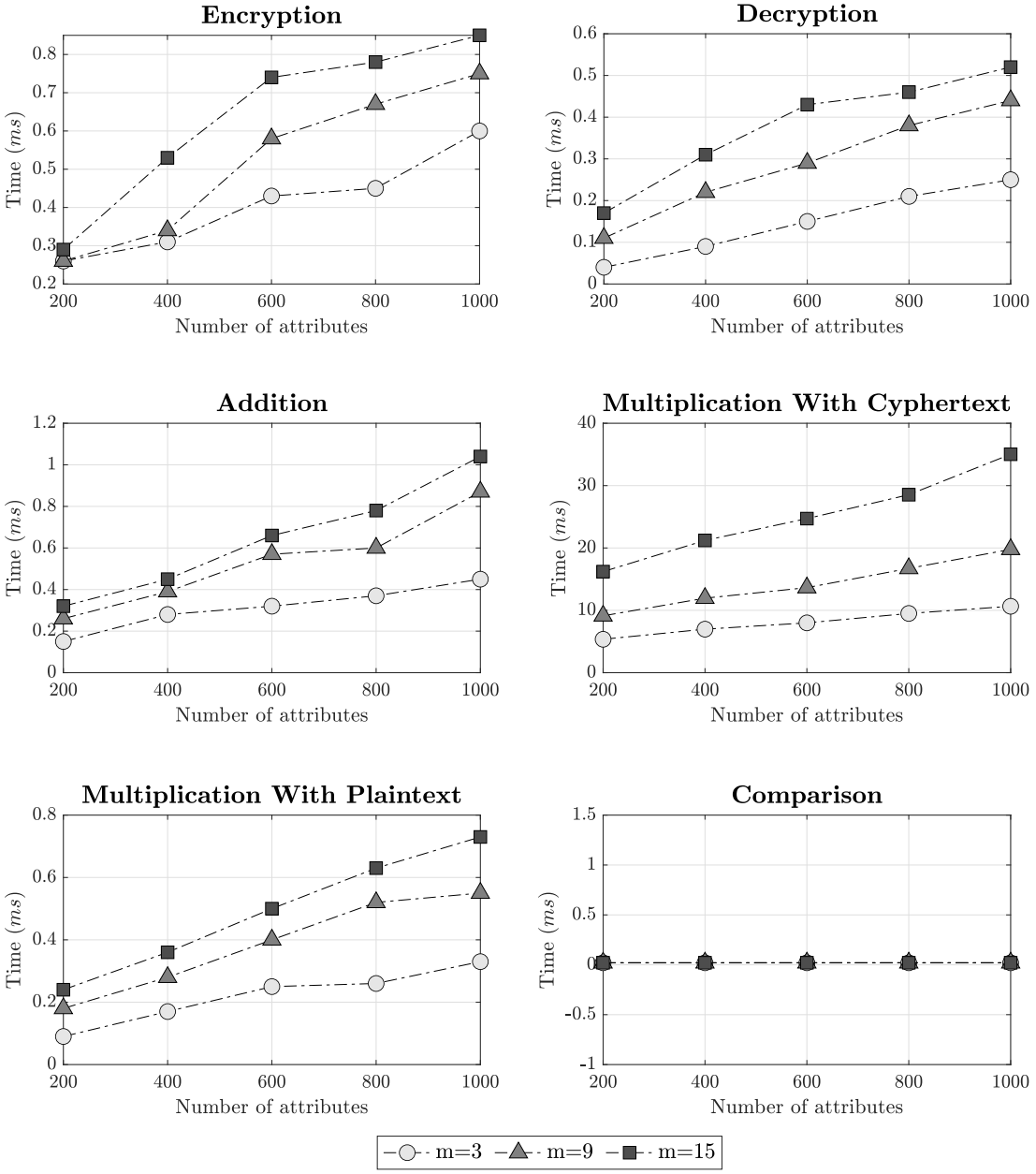


FIGURE 9.4: The MLS performance evaluation for different value of m and different numbers of attributes in a record, **[top left]** Encrypt(v); **[top right]** Decrypt(E); **[middle left]** addition: $E_1 \oplus E_2$; **[middle right]** multiplication: $E_1 \bigotimes E_2$; **[bottom left]** multiplication $c \circledast E_1$ and **[bottom right]** cyphertexts comparison. The values correspond to runtime is averaged over 10 iterations

the data owner. The largest dataset "Libras Mov" only required, on average, 3.86ms for data normalisation, 3.84ms for data encryption and 6.98ms for stratified CV data preparation.

The data owner participation with respect to network training is given in column 4 of Table 9.4 measured in terms of the average runtime over all TCV. Recall that data owner involvement in the model training is limited to division (inversion) operations with respect to the *TaylorLinear* approximation of the Sigmoid activation function (whenever it is encountered). Query prediction services, provided by a TPDM using the constructed SecureNN model, can be conducted using two different methods differing in

Table 9.3: Runtime for data owner data preparation and standard NN

No. UCI Dataset	Preparation (ms)	Data Encryption (ms)	CV time (ms)	Standard NN Exe. time (ms)
1. Banknote Auth.	1.08	1.59	5.63	632.64
2. Blood Trans.	1.10	1.08	1.95	365.82
3. Breast Cancer	2.21	2.80	1.96	268.58
4. Breast Tissue	1.20	0.47	0.64	124.32
5. Chronic Kidney	1.51	2.17	3.95	275.72
6. Dermatology	1.75	2.58	3.92	375.04
7. Ecoli	0.74	0.95	1.84	313.13
8. Iris	0.52	0.28	0.64	101.71
9. Libras Mov.	3.86	3.84	6.98	1202.13
10. Parkinsons	1.93	0.84	1.35	159.41
11. Pima Disease	1.79	3.00	3.04	410.20
12. Seeds	0.75	0.96	1.02	141.08

Table 9.4: Runtime for a SecureNN machine learning operating statistics

No. UCI Dataset	SecureNN		
	Total Exe. time (Sec.)	Data Miner proc. (Sec.)	Data owner partic. (Sec.)
1. Banknote Auth.	487.13	249.49	237.64
2. Blood Trans.	283.69	155.21	128.49
3. Breast Cancer	195.32	132.67	62.65
4. Breast Tissue	67.25	36.41	30.84
5. Chronic Kidney	189.05	118.36	70.69
6. Dermatology	263.81	172.58	91.23
7. Ecoli	240.77	141.09	99.69
8. Iris	065.69	34.45	31.24
9. Libras Mov.	873.85	656.75	217.11
10. Parkinsons	91.50	58.73	32.77
11. Pima Disease	312.84	178.40	134.44
12. Seeds	094.69	53.22	41.47

the way the activation function is approximated and the amount of data owner involvement, *TaylorLinear* and *Friendly Function*. *TaylorLinear* approximation requires data owner participation. As in the case of training the model; the time complexity for data owner participation using *TaylorLinear* will be in the order of $O(|neurons|)$. The data owner will decrypt approximated values for the activation function, inverse the value, encrypt the results and then return it to the TPDM. The data owner participation for predicting one query record label in the "Libras Mov" dataset, using a network topology of $\{90, 10, 15\}$ (see Table 9.2) was 0.14ms. *FriendlyFunction* approximation can be entirely conducted using the homomorphic operations facilitated by the MLS properties, therefore no data owner participation was required.

9.6.3 SecureNN Efficiency

The total runtime for training each network using SecureNN is given in column 2 of Table 9.4. Column 5 of Table 9.3 gives the runtime to train the same network without using any encryption. Note that the runtimes given in column 2 are in seconds (*Sec.*), whilst those given in column 5 are in milli-seconds (ms). As expected, training a NN over encrypted data introduces a computational overhead. The difference is due to computation complexity of the FHE mathematical properties and the linear approximation of the Sigmoid function using *TaylorLinear*. However, it is argued here, that this is not an unacceptable overhead, even for the largest dataset, the "Libras Mov" dataset, the network was trained in $873.85Sec.$

With respect to the computational overhead of using a trained SecureNN when it goes into usage, compared to standard NN (over plaintext neural parameters), further experiments were conducted using a single machine, which indicated that the runtime was negligible. The "Libras Mov" dataset was considered in this respect as it had the largest number of neurons in the NN topology and the largest number of class labels (see Table 9.2). Using *TaylorLinear* approximation where $d = 3$, $1,641,256$ predictions could be made per hour, whilst when using *FriendlyFunction* approximation $4,143,012$ predictions could be made per hour. Using standard NN, coupled with the standard using Sigmoid function, $655,463,103$ predictions can be made per hour. It can therefore be concluded that the standard NN is more efficient than the SecureNN using *TaylorLinear* and *FriendlyFunction*, although the *FriendlyFunction* is more efficient than the linear estimation using *TaylorLinear*.

9.6.4 Accuracy

The classification accuracy obtained using standard NN was compared with the accuracy of the proposed SecureNN approach using both *TaylorLinear* and *FriendlyFunction* approximation using the same network topology and parameters in all cases. The intuition was that the SecureNN should produce comparable results to those obtained using the standard NN; if so the SecureNN could be said to be operating correctly. The effectiveness evaluation metrics used were: (i) Precision (**P**), Recall (**R**) and the **F1** measures [194] and (ii) a comparison of the value of the loss function calculated for different numbers of epochs. To provide a precise and fair comparison, the performance measures were calculated over the same test set for all activation functions (Sigmoid, *TaylorLinear* and *FriendlyFunction*). This approach was previously used in [159, 200] for comparing performances of different activation functions.

Table 9.5 shows the **P**, **R** and **F1** values obtained when using the standard and SecureNN frameworks. From the table it can be seen that:

Precision: For ten of the datasets considered the precision (P) values obtained for all three approaches was more-or-less equal. For the remaining two cases, "Blood Trans" and "Libras Mov", the values obtained using *TaylorLinear* were comparable with the standard approach; whilst using *FriendlyFunction* the precision values obtained were slightly lower.

Recall: In terms of recall (R), the values obtained were similar in nine cases. In the remaining three cases, two cases, "Breast Tissue" and "Libras Mov", the *TaylorLinear* produced comparable results to Sigmoid, whilst *FriendlyFunction* produced slightly lower values. In the case of the "Blood Trans" dataset, the recall values obtained using *TaylorLinear* and the Sigmoid function were equal, however the value obtained using *FriendlyFunction* was slightly higher.

F1: With respect to the F1 values obtained, these were comparable in ten cases; while in one case,"Breast Tissue", *TaylorLinear* and *FriendlyFunction* produced identical values slightly lower than the Sigmoid function. In the remaining case, "Libras Mov", the *TaylorLinear* was comparable to the Sigmoid function and *FriendlyFunction* was slightly lower.

The overall average values for precision were 0.82, 0.82 and 0.79 for Standard NN using the Sigmoid function, SecureNN using *TaylorLinear* and SecureNN using *FriendlyFunction*, respectively. The overall average values for recall were 0.80, 0.80 and 0.79, respectively; and the average F1 values were 0.80, 0.80 and 0.78. Thus it can be concluded that the SecureNN approach, coupled with *TaylorLinear* approximation, produced comparable results to Standard NN (without encryption). The results produced using *FriendlyFunction* approximation were not as good, because the approximation was coarser than the values produced using *TaylorLinear* approximation (as show in Figures 9.2 and 9.3).

The loss function values obtained using SecureNN coupled with *TaylorLinear* and *FriendlyFunction* were compared with those produced using standard NN with respect to different numbers of epochs. Figure 9.5 shows the average loss function for the UCI datasets for standard and SecureNNs for different epochs from 10 to 190 increasing in steps of 20. From the figure it can be seen that in all cases the *FriendlyFunction* produced the worst performance. The operation of the proposed SecureNN framework coupled with *TaylorLinear* approximation and Standard NN was comparable. Thus it can be concluded that *TaylorLinear* approximation can approximate the value of the Sigmoid function while at the same time maintaining the overall accuracy of the trained models.

TABLE 9.5: Prediction accuracies using: (i) standard NN with Sigmoid activation, (ii) SecureNN with *TaylorLinear* approximation and (iii) SecureNN with *FriendlyFunction* approximation.

DataSet	Standard NN			Secure NN					
				TaylorLinear			*FriendlyFunction*		
	P	R	F1	P	R	F1	P	R	F1
1. Banknote Auth.	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
2. Blood Trans.	**0.73**	**0.54**	0.51	**0.71**	**0.54**	0.51	**0.61**	**0.65**	0.55
3. Breast Cancer	0.65	0.65	0.66	0.65	0.65	0.66	0.66	0.67	0.67
4. Breast Tissue	0.66	**0.64**	**0.64**	0.61	**0.61**	**0.57**	0.60	**0.57**	**0.57**
5. Chronic Kidney	0.97	0.98	0.98	0.97	0.98	0.98	0.97	0.98	0.98
6. Dermatology	0.94	0.94	0.94	0.96	0.96	0.97	0.95	0.94	0.95
7. Ecoli	0.62	0.59	0.61	0.62	0.62	0.62	0.58	0.59	0.59
8. Iris	0.97	0.97	0.97	0.98	0.98	0.98	0.97	0.97	0.97
9. Libras Mov.	**0.78**	**0.76**	**0.76**	**0.77**	**0.77**	**0.77**	**0.70**	**0.67**	**0.68**
10. Parkinsons	0.85	0.82	0.84	0.85	0.83	0.85	0.82	0.83	0.83
11. Pima Disease	0.75	0.72	0.74	0.75	0.70	0.72	0.75	0.71	0.72
12. Seeds	0.94	0.94	0.95	0.94	0.94	0.95	0.91	0.91	0.91
Average	**0.82**	**0.80**	**0.80**	**0.82**	**0.80**	**0.80**	**0.79**	**0.79**	**0.78**

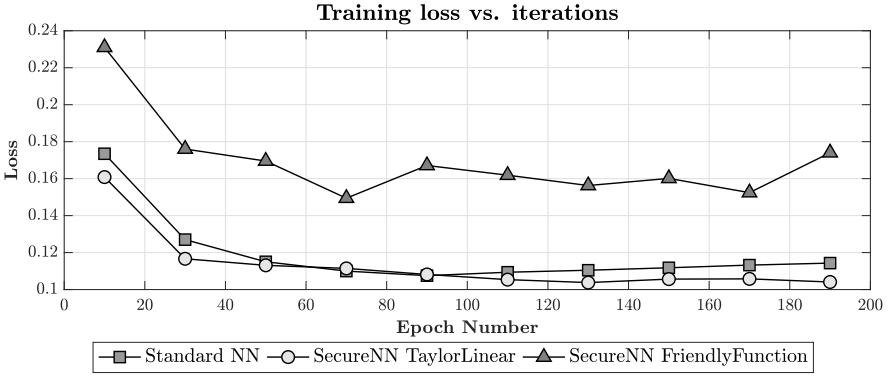


FIGURE 9.5: Loss function for three NN for different number of epochs

9.6.5 Security

Using the proposed SecureNN framework the TPDM is considered to be a "Honest but Curious" party; thus the semi-honest security model where the TPDM executes the stated algorithm without deviation and does not fail to provide the required service is again applicable. At the same time the TPDM is curious, in the sense that it would "look" at the available data and calculated results. The security of the proposed SecureNN framework was thus evaluated in terms of the semi-honest model by identifying potential attacks that can be instigated during model training and provision of the query prediction service. Model training was performed on encrypted data, encrypted data labels and encrypted NN parameters, thus the only two potential forms of attack are: a Cyphertext Only Attack (COAs) available whenever adversaries have access to such cyphertexts or a Model Inversion Attack (MIA); as in the case of earlier proposed encryption schemes.

The proposed MLS, as in the case of Liu's original FHE scheme, is a probabilistic scheme that produces different cyphertexts for the same plaintext value each time it is applied, even when using the same secret key. This feature means that MLS cyphers are semantically secure, hence accessing cyphertexts does not provide any useful information with respect to the associated plaintext values from the perspective of an adversary. COAs are more likely to succeed when attackers have background knowledge of the data frequency of the original data values. Knowledge associated with the ordering feature of some order preserving encryption schemes might allow an adversary to infer the ranges containing dense data. Alternatively, frequency analysis could allow attackers to highlight cyphertexts with the same frequency as plaintexts (if such plaintexts were available) and then identify cyphertexts that have the same frequency. However, this will not be possible in the case of MLS, because different cyphertexts are produced for the same plaintext values that makes such inference hard. The entire model training was conducted over MLS cyphers and no decryption took place at the TPDM side which implies even more security. Hence it is argued that the proposed SecureNN framework, founded on MLS, is secure with respect to COAs.

With respect to the query classification/prediction services provided by a TPDM, in the context of the SecureNN framework, there were two issues of concern: (i) the presence of sensitive information in prediction requests (query), and (ii) the protection of the knowledge embedded in the trained model to avoid an MIA. In the context of the first concern, prediction requests are sent in encrypted form by the data owner, the TPDM performs the requested inference over the encrypted data and then produces an encrypted prediction that can only be decrypted by the data owner who is also the query

owner. In the context of the second concern, all weights and biases are encrypted using MLS which, as noted above, is encrypted and has semantic security features; therefore MIAs cannot be instigated without access to the required encryption key.

9.7 Summary

This chapter has presented the Modified Liu's Scheme (MLS), a novel FHE scheme that addresses the cyphertexts inflation problem and supports secure data comparison. In Liu's original FHE scheme, the number of sub-cyphertexts increased exponentially with each homomorphic multiplication from the initial m sub-cyphertexts to m^2 for one multiplication. In MLS, the number of sub-cyphertexts is reduced back to m using a dimensionality reduction algorithm that allows the re-encryption of the cyphertexts resulting from homomorphic multiplication, without decryption, using the concept of *trapdoors*. The ordering of data is also preserved using MLS. The idea relies on introducing conditions over the key generation process and usage of the ω-concept; the adding of a "gap" between generated cyphers in order to allow the inclusion of some noise without compromising the data ordering. Mathematical proofs are presented in Appendix A for correctness of the data encryption/decryption, dimensionality reduction and order preservation processes.

The chapter also presented the SecureNN framework that utilised the MLS to allow for privacy preserving multi-layer Neural Network (NN) learning with back-propagation. Using SecureNN, coupled with MLS a NN model was training and the activation function approximated with little or no data owner participation. The model is trained using encrypted data and encrypted network parameters; similarly the model can be used to provide secure query prediction services for the same data owner. The evaluation demonstrated that the training of networks and their usage does not entail any significant computational overhead over the data owner while maintaining a comparable accuracy to that obtained using standard NN with Sigmoid activation. This makes the framework ideally suited to PPDM whereby model training and prediction services are delegated to a TPDM (with limited data owner participation).

In the following chapter, the thesis is concluded with a summary and an overview of the main finding in terms of the original research question and subsidiary research questions postulated in Chapter 1. The chapter also presents some potential directions whereby the work presented in this thesis can be extended in the future.

Chapter 10

Conclusion and Future Work

10.1 Introduction

This concluding chapter presents an overall summary of work presented in this thesis together with the main findings and some suggested directions for future work. The chapter commences, Section 10.2, with a summary of the material presented. Section 10.3 presents the main findings and contributions of the work in the context of the research question, and subsidiary research questions, presented in Chapter 1. The chapter is then concluded, in Section 10.4, with a review of potential areas for future research that build upon, and extend, the work presented in the thesis.

10.2 Summary of Thesis

This section gives a summary of the work presented in this thesis. The thesis commenced with an introductory chapter, Chapter 1, that establish: the motivations for the research, the research question and subsidiary research questions, the main contributions of the research and the research methodology. The central idea presented in this thesis is that Privacy Preserving Data Mining (PPDM), using a third party, can be implemented using a proxy for the data that has been encrypted in such a manner so that order preservation is maintained and/or simple arithmetic operations are supported.

The main motivation for the idea of cryptography-based PPDM using a TPDM (Third Party Data Miner) was the rise in the usage of cloud storage and the associated computing services, together with the increased demand, on behalf of data owners, to securely apply the techniques of machine learning and data mining to their data. The challenge of involving a TPDM was that it imposed a data privacy preserving requirement. The proposed solution was to use a data proxy coupled with cryptography as a means for providing privacy preservation and preventing unauthorised use of confidential data. The focus throughout the thesis was on Homomorphic Encryption (HE) and Property Preserving Encryption (PPE) schemes that allow some operations to be applied over cyphertexts, without decryption, in such a way that the results obtained will match the results obtained as if the operations were applied over plaintext equivalents. The main objective was to conduct the PPDM in such a way that data owner participation was minimised (in comparison with previously proposed systems).

Two categories of data owner scenario were considered throughout the thesis: (i) the single data owner scenario where the data to be outsourced belongs to a single data owner and (ii) the multiple data owners scenario where the data to be outsourced belongs to more than one data owner and which then leads to the idea of collaborative data mining over the union of the datasets. Any number of machine learning algorithms

could have been considered in the context of PPDM, however the research focus was on data clustering using established approaches such as the k-Means, NNC and DBSCAN algorithms and data classification using kNN, a natural progression from NNC, and finally neural networks that utilise back-propagation.

Chapter 2 then presented a number of preliminaries concerning the notation used throughout the thesis. The chapter also provided a compressive review of PPDM techniques, including: data modification, Secure Multi-Party Computation(SMPC), secret sharing and Homomorphic Encryption (HE). In Chapter 3 a literature review concerning cryptography, with emphasis on HE schemes, was presented. The material presented encompassed: HE scheme properties, categories of HE scheme, the important limitations of HE schemes with respect to PPDM and examples of two encryption schemes adopted with respect to the work presented later in the thesis, Liu's Fully HE (FHE) scheme and the Paillier scheme.

The following six chapters, presented different approaches for achieving PPDM using data proxies coupled with cryptographic approaches, that did not require data proxy decryption by the TPDM. The chapters considered both the single data owner and the multiple data sources scenarios. The proposed solutions were directed at, or illustrated with, different data mining algorithms from established data clustering algorithms to more sophisticated neural network classification models. Each of the chapters was structured in the similar manner comprising: an introduction of the proposed solution, a review of the adopted encryption schemes, one or more illustrative secure data mining algorithms founded on the proposed approach and an evaluation of the proposed approach.

The first proposed approach was presented in Chapter 4 and was founded on the idea of Updateable Distance Matrices (UDMs). The approach was directed at the single data owner scenario and illustrated using a secure k-Means clustering, the Sk-Means algorithm was proposed. The reported evaluation demonstrated that Sk-Means, founded on the use of UDMs, dramatically reduced the data owner participation compared to other comparable approaches from the literature. Data owner participation was limited to updating the UDM. It was observed that the need to update the UDM was consequent on the nature of k-Means clustering algorithm used to illustrate the use of UDMs, that necessitated repeated recalculation of cluster centroids. However, more importantly, it was noted that the UDM concept had a significant security concern associated with, given that the UDM elements were exchanged in plaintexts and that a UDM was simply a large collection of linear equations that might permit reverse engineering of aspects of the original dataset.

Chapter 5 therefore sought to address the security concerns associated with UDMs by proposing the concept of Encrypted Updatable Distance Matrices (EUDMs) and Encrypted Distance Matrices (EDMs) coupled with the idea of Cryptographic Ensembles whereby more than one encryption scheme was used to maintain security. More specifically Liu's FHE scheme and the proposed Frequency and Distribution Hiding Order Preserving Encryption (FDH-OPE). Liu's FHE scheme was used to encrypt the dataset while the FDH-OPE scheme was used to encrypt the distance matrix to arrive at encrypted version. As a result the TPDM had access to the order of the distances instead of the real distances values as in the case of unencrypted UDMs. Any algorithm that utilised the proposed approach would therefore be a "double blind" algorithm because the Cryptographic Ensemble comprised two encryption schemes. The distinction between an EUDM and an EDM was that, as the name suggests, the first could be updated while the second could not. The first could therefore support clustering algorithms, such as k-Means that required the recalculation of cluster centroids. The second

required less memory resource than the first. The application of EUDMs and EDMs was presented in the context of a number of different data clustering algorithms; Double Blind Secure k-Means (DBSk-Means) and Double Blind Secure NNC (DBSNNC). The evaluations indicated that the Cryptographic Ensemble and EUDM/EDM collectively provided a solution for both reducing the data owner participation while preserving the data privacy without compromising the accuracy of final clustering configuration.

The next chapter, Chapter 6, was the first to consider the multiple data owners scenario; collaborative data clustering whereby the clustering process was entirely delegated to the TPDM. The work presented in the chapter was founded on the EDM concept presented in the previous chapter, Chapter 5, and introduced a process that allowed multiple EDMs, generated by individual data owners, to be "pooled" to form a single Global EDM (a GEDM). The pooling process was presented with respect to horizontally partitioned data only. For collaborative clustering to operate correctly an alternative Cryptographic Ensemble was required to allow multiple data sources and hence the FDH-OPE scheme from the previous chapter was replaced with a proposed Multi-Users Order Preserving Encryption (MUOPE) scheme that preserved the ordering of data distributed across multiple sources. Extensive evaluations were conducted using different UCI datasets and synthetic datasets of various size. The evaluations demonstrated that: (i) the usage of GEDMs did not introduce a significant overhead on behalf of data owners, (ii) the clustering configurations obtained using the GEDM matched configurations obtained using standard algorithms and (iii) the GEDM provided the potential for large scale collaborative data clustering involving many data owners. However, the GEDM approach featured a significant memory resource requirement as the dimensions of the GEDM were correlated with the number of data records in the global dataset in question; the union of the datasets belonging to the participating data owners.

In Chapter 7, the Secure Chain Distance Matrices (SCDM) concept was presented. The central idea underpinning SCDMs was to reduce the memory resources required to conduct secure data clustering while maintaining the minimum data owner participation. The memory resource was measured in terms of the number of elements in the adopted distance matrix representation. The SCDM reduced the memory resource requirements compared to other approaches presented in Chapters 4 and 5. Data privacy was preserved using a Cryptographic Ensemble comprised of Liu's FHE scheme and the proposed FDH-OPE scheme. The significance of the proposed SCDM concept was that it was applicable with respect to a range of data mining algorithms that required distance comparison. Usage of the SCDM concept was illustrated with respect to a number of well-established data clustering algorithms, namely k-Means, NNC and DBSCAN. The utility of the SCDM concept was further illustrated using kNN data classification and collaborative querying (class labelling). The reported evaluation indicted a slight increase in runtime required to cluster and classify encrypted data records using the SCDM compared to earlier distance matrix-based approaches. The reason was the "chain feature" used to calculate the similarity between data records. The reported evaluation comparing the usage of the SCDMs and the EUDMs in the context of secure data clustering demonstrated that: (i) the complexity of the data preparation process was lower for SCDMs compared to EUDMs, (ii) the clustering algorithms that utilised the EUDM were more efficient than the algorithms that utilised the SCDM concept and (iii) the SCDMs and EUDMs produced comparable results. The main issue of adopting the SCDM in the context of collaborative data classification was how to process queries that belonged to authorised Query Owners (QOs) whilst: (i) preserving data owner key confidentiality when encrypting a QO's query and (ii) determining the similarity between the outsourced datasets and the new query records without involving the data owner

or QO in the process. To this end the chapter presented the Secure Query Cyphering (SQC) protocol, and a secure binding algorithm. The SQC protocol operated between the data owner and QOs and was designed to allow QOs to encrypt their query without accessing the data owner's secret key. The binding process allowed the use of SCDMs to derive the similarity between outsourced data records and query records in the context of k Nearest Neighbour (kNN) classification. The evaluation of the kNN coupled with the SCDM concept demonstrated that the accuracy of the resulting classification was comparable to the accuracy produced using standard kNN.

Chapter 8 extended the SCDM concept presented in Chapter 7 by considering the multiple data owners scenario and the potential for collaborative data mining. This was achieved by "binding" multiple SCDMs to form a single Super SCDM (SSCDM). In the chapter it was noted that SSCDMs could be constructed with respect to different data partitionings, horizontal, vertical and arbitrary. The chapter also considered the possibility of updating SSCDMs by allowing record deletion and/or addition whenever corresponding datasets were modified. The application of the SSCDM concept was again illustrated with respect to secure data clustering. Two implementations were considered: Secure DBSCAN and Secure NNC. Extensive experimentation was conducted to evaluate the usage of SSCDMs and to compare the operation of SSCDMs with GEDMs (as presented in Chapter 6). The evaluation indicted that SSCDMs provided a scalable and efficient approach for collaborative data clustering without resorting to a well-established SMPCs. In terms of accuracy, the SSCDM approaches produced comparable results to those produced using standard algorithms, while producing comparable results to those produced using GEDMs.

Chapter 9 investigated the idea of using a single encryption scheme, rather than the Cryptographic Ensembles from earlier chapters. A novel FHE scheme, Modified Liu's Scheme (MLS), was presented; a modification of Liu's FHE scheme that also addressed the issues of the increasing number of sub-cyphertexts that results when performing multiplications using Liu's original scheme. The proposed MLS addressed this issue while maintaining the same homomorphic properties as Liu's original scheme. Instead of illustrating MLS in terms of clustering or kNN, as in the case of earlier chapters, a more ambitious machine learning algorithm was considered; namely a Secure Neural Network (SecureNN) that allowed model training and prediction over encrypted data. The proposed SecureNN framework required only minimal data owner participation during training and usage that was limited to performing a division required to estimate the activation function. Extensive evaluations of MLS, in terms of the proposed SecureNN framework, indicted that the networks generated achieve a classification accuracy comparable with that of identical networks generated without encryption.

10.3 Main Findings and Contributions

The main findings and contributions of the work presented in this thesis are given in this section. The discussion is firstly presented in the light of the subsidiary research questions and then in terms of the overriding research question that this thesis seeks to address, as presented in Chapter 1 (Section 1.3). Each of the subsidiary research question is considered in turn as follows:

1. "*What is the most appropriate HE scheme that satisfies the requirements for outsourcing data mining activities to a TPDM in the context of PPDM and the single data owner scenario?*"

The most appropriate HE schemes are those that *efficiently* support *most* of the required operations over cyphertexts without decryption so as to reduce the data owner participations, and thus the outsourcing of the entire data analysis activities. HE schemes, as presented in Chapter 3, can be divided into three categories: (i) Partial HE (PHE) schemes that support limited homomorphic addition or homomorphic multiplication but not both, (ii) SomeWhat HE (SWHE) schemes that support unlimited homomorphic addition and a limited number of multiplication and (iii) Fully HE (FHE) schemes that support both unlimited addition and multiplication. The FHE schemes can be categorised further in to: schemes that incorporate noise management techniques and schemes that are noise-free. The noise management techniques were considered in Chapter 3, and feature high computational cost associated with the re-encryption of cyphertexts whenever the noise reaches a certain threshold. Many data mining algorithms require distance calculations, similarity determination (data comparison) and the evaluation of linear and non-linear functions. This imposes a requirement for schemes that support both addition and multiplication. **Therefore the most appropriate HE scheme that satisfies the requirements for outsourcing data mining activities is one that supports addition and multiplication over encrypted data with appropriate noise reduction where necessary**. The intuition underpinning the use of FHE schemes was supported by the effective implementation of a wide range of data mining algorithms that made use of such schemes. Noise free FHE schemes allow the secure operation of sophisticated data mining algorithms that comprises extensive homomorphic multiplications as demonstrated through the implementation of a neural network algorithm.

2. "*Is HE the best form of cryptography to provide an effective solution to the PPDM problem?*"

   HE schemes, including FHE schemes, do not provide an entire solution to PPDM problem. There are limitations associated with FHE scheme (as discussed in Chapter 3). The most significant of which is the need for data comparison operations which are frequently required with respect to many data mining algorithms. Comparison between data records and comparison with a threshold value cannot be directly performed over HE cyphertexts. **Therefore, HE schemes provide only a partial solution and need to be coupled with additional secure mechanisms to allow comparison.** Note that any proposed mechanism needs to satisfy the security requirements for the considered data owner scenario, while at the same time maintaining the accuracy of the final data mining results.

3. "*Following on from 2, if HE is not the most appropriate form of cryptography, what is the most appropriate form of encryption required to provide a better solution?*"

   The mathematical properties associated with current HE schemes do not provide a comprehensive solution to the PPDM problem given that, as noted above, there is no HE scheme that supports comparison over cyphertexts. One proposed solution is to incorporate periodic recourse to data owners, during the data mining process, so that the data owners can perform the data operations (on unencrypted data) that the adopted HE scheme does not support. However, in this approach the amount of data owner participation is significant, calling in to question the advantages that DMaaS has to offer for single data owner scenario. **Therefore, HE schemes need either: (i) to be coupled with an Order Preserving**

Encryption scheme, a Cryptographic Ensemble, that supports data ordering and thus comparison between cyphertexts; or (ii) be modified so that the idea of order preservation over generated cyphertexts, to allow direct comparison, is maintained. The thesis explored both solutions. The proposed MLS is argued to be the most appropriate form of encryption for PPDM.

4. "*What is the most effective way of supporting the required operations not supported by an encryption scheme (without decrypting the data) so as to, where necessary, minimise the number of data owner interactions?*"

   The proposed answer to this question was to provide a proxy for data; a proxy that supported efficient updating of the proxy whenever new data records were required to be added or deleted. **The particular form of data proxy proposed in this thesis was the idea of distance matrices of various forms. These matrices held the distance between every record to every other record in outsourced data and acted as a proxy for the real data. A range of different forms of distance matrices were considered, including: UDMs, EUDMs, EDM and SCDMs.** The distances held in a distance matrix are calculated, on start-up, by the data owner and, in the case of the later forms of distance matrix considered, encrypted in such a way that data comparison (by a TPDM) was supported.

5. "*Given potential solutions to above, what is the most appropriate mechanism, or mechanisms, for evaluating these solutions?*"

   The mechanism for evaluating the proposed secure data mining algorithms and encryption schemes was primarily by comparing the results obtained with those obtained using equivalent standard algorithms. The operation of the standard algorithms applied to unencrypted datasets was compared with proposed secure algorithms using the same algorithm parameters. Well-established evaluation measures were used to compare data clustering configurations, such as the Silhouette Coefficient or the number of produced clusters and number of iterations (in the case of iterative data clustering). Classification algorithms were compared using confusion matrix measures such as Precision, Recall and F1. The intuition was that any proposed secure algorithms should produce equivalent (or comparable) results to those produced using standard equivalent algorithms; if so the secure algorithms could be said to be operating correctly.

6. "*Can the proposed single data owner scenario solutions be extended to support scalable collaborative data mining (the multiple data owners scenario) while keeping the data owner participation at a minimum?*"

   As noted above, the mechanisms presented in this thesis to address single data owner scenarios were founded on the idea of proxy data, namely the use of distance matrices. **It was found that by extended the distance matrix idea by allowing the combining of multiple matrices and calculating a new super-matrix that the multiple data owners scenario could be addressed.** A number of variation of this proposed solution were presented, of note were GEDMs and SSCDMs generated by pooling a number of EDMs or binding different SCDMs respectively. The conducted experimental analysis demonstrated that the binding and pooling methods could provide for secure scalable collaborative data mining.

7. "*Is it possible to tailor (or modify) any proposed HE scheme so that it can support the operations required by a range of data mining algorithms, as opposed to only a small number of specific algorithms?*"

   The challenge of modifying HE schemes so as to address the issue of unsupported operations was addressed towards the end of the thesis in Chapter 9. Secure comparison over cyphertexts and scalable multiplication were considered by focusing on Liu's FHE scheme. The central idea was to modify Liu's FHE key generation process and encryption function so as to incorporate the idea of OPE while maintaining the FHE properties of the original scheme. **The proposed MLS demonstrated that the key values, used to encrypt sub-cyphertexts, could be generated in such a way that data ordering was maintained while at the same time supporting the HE properties of the original scheme.** The proposed MLS uses what was referred to as the ω-concept; the idea of creating a "gap" between generated cyphertexts and maintained the probabilistic nature of the scheme while preserving the data ordering.

Returning to the central research question of the thesis:

"Using cryptography is it possible to securely, effectively and efficiently delegate data analysis to a third party data miner while minimising any required interaction with data owners?"

From the foregoing answers to the subsidiary questions it can be stated that it is possible to "effectively and efficiently delegate the data analysis" to a TPDM with limited (or no) data owner participation while maintaining an appropriate level of security.

For the completeness, the main contributions of the work presented in this thesis, from Chapter 1, are restated here:

1. Improvement in data security by obviating the need for sending data in any form to TPDMs and/or sharing it with other participating parties using the concept of Super Secure Chain Distance Matrices (SSCDMs) and idea of virtual lists.

2. Reducing the amount of data owner participation when undertaking secure data mining operations, and in some cases avoiding it entirely, when the data mining is undertaken by a TPDM using the concept of distance matrices.

3. Introducing the idea of Cryptographic Ensembles that comprise an HE scheme and a bespoken OPE scheme, therefore providing a solution to secure data clustering involving distance comparison.

4. Six secure data clustering algorithms, in the context of the single data owner scenario, that operated over encrypted data, without requiring decryption and without delegating keys to non-colluding parties:

   (a) Secure k-Means using an Updatable Distance Matrices (UDMs) (Sk-Means).

   (b) Double Blind Secure k-Means using an Encrypted UDMs (EUDMs) (DBSk-Means).

   (c) Double Blind Secure Nearest Neighbour Clustering (DBSNNC) using Encrypted Distance Matrices (EDMs).

   (d) Sk-Means, SNNC and SDBSCAN using Secure Chain Distance Matrices (SCDMs).

5. Four Secure collaborative data clustering algorithms, addressing the multiple data owners scenario, that operated over encrypted data without decryption and without sharing data between participants or delegating keys to non-colluding parties:
   (a) Secure DBSCAN (S-DBSCAN) and Secure NNC (S-NNC) using Global EDMs (GEDMs).
   (b) Secure DBSCAN (SDBSCAN) and Secure NNC (SNNC) using Super SCDMs (SSCDMs).
6. A Secure Neural Network algorithm (SecureNN) that facilitates model training and query classification/prediction with only very limited data owner participation.
7. A novel Order Preserving Encryption scheme, the Multi-Users Order Preserving Encryption (MUOPE) scheme, that supported collaborative data clustering and facilitated preservation of the ordering of data distributed across multiple data owners.
8. The Frequency and Distribution Hiding OPE (FDH-OPE) scheme, an amalgamation of two order preserving encryption schemes, the nonlinear order preserving scheme [51] and Zheli et al.'s scheme [52], each of which facilitates information hiding and the reduction of information leakage.
9. A "dimensionality reduction" algorithm that addressed the scalability issue of the exponentially increasing size of cyphertexts with the application of each multiplication operation as in the case Liu's FHE scheme.
10. The Modified Liu's Scheme (MLS) that supported secure data comparison while preserving the same HE mathematical properties as Liu's original scheme [53].
11. Scalable collaborative data clustering that allowed a very large number of parties (multiple data owners) to conduct data analysis over the union of their data without sharing private data and without recourse to the expensive SMPC protocols.
12. The Secure Query Cyphering (SQC) protocol that preserved a query record's privacy, while at the same time preserving the data owner key confidentiality when classifying the query record using a proposed Secure kNN (SkNN) algorithm.

10.4 Future Work

The work presented in this thesis has proposed a number of cryptographic mechanisms to securely delegate data clustering and classification to a TPDM while reducing the data owner participation. The work considered two scenarios, the single data owner scenario where a data owner wants to delegate their data analysis to a TPDM who provides DMaaS, and the multiple data owners scenario where a number of data owners wish to collaboratively so as to develop (say) a data mining model over the union of their datasets. In this section, a number of potential future directions of research are suggested. The future work can be divided into the **PPDM context** and the **alternative domains context**. The former is concerned with aspects where the current work directed at PPDM can be extended, improved or applied. The later is concerned with potential application domains, other than PPDM, where the encryption schemes and mechanisms proposed in this thesis may be applied. Each is discussed in further detail in the following two sub-sections.

10.4.1 Future Work In The Context of PPDM

In the context of PPDM, the work presented in this thesis can be further extended as follows.

1. **Improvement of the security of proposed SkNN data classification algorithm:** Data classification using the proposed SkNN algorithm operated over unencrypted data labels. Therefore, the determination of the major class label (the SkNN stage 2) was not secure given that the TPDM would know the classes of the outsourced encrypted data and the classes of the query records. Deterministic encryption schemes, or other PPDM techniques, might provide a solution. It is thought that adopting such methods to preserve the security of the data label would be better than confiding the label values as plaintext.

2. **Alternative evaluation:** A limiting factor of the evaluations presented in this thesis was that they involved implementations on a single machine, therefore no real consideration was given to the real communication overhead when the TPDM is hosted in the cloud. Evaluating the proposed solutions by implementing the TPDM as real CSPs would demonstrate the expected communication overhead when outsourcing the data analysis to CSPs.

3. **Alternative data mining algorithms:** A conjectured fruitful avenue for further research is to consider a wider range of machine learning algorithms. For example to consider the application of MLS, and the other proposed schemes, to implement (say) Support Vector Machine (SVM) learning, deep neural network learning using Convolutional Neural Networks (CNNs) and logistic regression.

4. **Real application domains:** The work presented in this thesis has focused on well-establish benchmark datasets from the UCI data mining repository and simulated datasets. There was good reason for this as it provided a ready mechanism for comparison. However, what might be termed "real" datasets drawn from real life applications should also be considered in the context of future work.

10.4.2 Future Work In The Context of Alternative Domain

The application domain considered in this thesis was PPDM. A number of encryption schemes where proposed: the FDH-OPE scheme, MUOPE scheme and MLS. It is conjectured that these schemes have application in other domains than PPDM. It is suggested that an investigation of alternative application domains for the proposed schemes may have merit. For example, the HE and OPE schemes introduced in this thesis could be utilised to provide for secure searching and retrieval over encrypted data (an application domain considered in [201, 202]). Alternatively, secure signal processing (see for example [203, 204]), secure face matching (see for example [205]), secure database as a service provision and private database processing (see for example [206–208]). Investigation of these alternative domains would thus provide for a further opportunity to extend the investigation presented in this thesis.

Similarly the range of secure clustering and classification mechanisms presented in this thesis are likely to have alternative forms of application than PPDM. One specific example is the preservation of location privacy in the context of spatial query systems. Spatial queries pose a serious threat to user location privacy as the location of a query may reveal sensitive information about (say) a mobile user (as discussed in [209]). A mobile user can send a query request to a Location Based Service (LBS) requesting the k nearest points of interest given their current location. It is suggested that such requests

can be sent in a secure manner by adopting the proposed SkNN approach (presented in Chapter 7). The idea can be extended to location privacy with respect to GeoFences[1] as described in [209–212].

[1]GeoFence is a location-based service in which an App or other software uses GPS, RFID, Wi-Fi or cellular data to trigger a pre-programmed action when a mobile device or RFID tag enters or exits a virtual boundary set up around a geographical location, known as a geofence.

References

[1] Kuan-Ching Li, Hai Jiang, Laurence T. Yang, and Alfredo Cuzzocrea. *Big data: algorithms, analytics, and applications*. CRC Press, Taylor and Francis Group, 2015.

[2] Microsoft TechNet. Cloud computing survey the results, 2018. URL `http://download.microsoft.com/documents/uk/technet/cloud-power/ITPro_survey_v5.pdf`.

[3] Gururaj Ramachandra, Mohsin Iftikhar, and Farrukh Khan. A comprehensive survey on security in cloud computing. *Procedia Computer Science*, 110:465–472, 2017. ISSN 1877-0509.

[4] R. L. Lagendijk, Z. Erkin, and M. Barni. Encrypted signal processing for privacy protection: Conveying the utility of homomorphic encryption and multiparty computation. *IEEE Signal Processing Magazine*, 30(1):82–105, Jan 2013. ISSN 1053-5888.

[5] Cristina Vargas. Cloud market in 2019 and predictions for 2022, 2019. URL `https://www.skyhighnetworks.com/cloud-security-blog/microsoft-azure-closes-iaas-adoption-gap-with-amazon-aws/`.

[6] Lawrence O Gostin. National health information privacy: regulations under the Health Insurance Portability and Accountability Act. *The Journal of the American Medical Association (JAMA)*, 285(23):3015–3021, 2001.

[7] Anup Kumar Das et al. European Union's general data protection regulation, 2018: A brief overview. *Annals of Library and Information Studies (ALIS)*, 65(2): 139–140, 2018.

[8] Amazon. Amazon S3 Service Level Agreement., 2018. Available at `https://aws.amazon.com/s3/sla/`, [Online; accessed 18-March-2019].

[9] Google. Google Compute Engine Service Level Agreement (SLA), 2018. Available at `https://cloud.google.com/compute/sla`, [Online; accessed 18-March-2019].

[10] Microsoft. SLA for Azure machine learning service, 2018. Available at `https://azure.microsoft.com/en-gb/support/legal/sla/machine-learning-service/v1_0/`, [Online; accessed 18-March-2019].

[11] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. Trusted execution environment: what it is, and what it is not. In *13th International Symposium on Parallel and Distributed Processing with Applications (ISPA)*, volume 1, pages 57–64. IEEE, 2015.

[12] Pierangela Samarati. Protecting respondents identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001. ISSN 1041-4347.

[13] Tharam Dillon, Chen Wu, and Elizabeth Chang. Cloud computing: issues and challenges. In *24th International Conference on Advanced Information Networking and Applications*, pages 27–33. IEEE, 2010.

[14] Arvind Narayanan and Vitaly Shmatikov. Robust De-anonymization of large sparse datasets. In *2008 Symposium on Security and Privacy*, pages 111–125. IEEE, 2008.

[15] Latanya Sweeney. Matching known patients to health records in Washington state data. Data Privacy Lab, Institute for Quantitative Social Science, Harvard University, July 2013. URL `http://thedatamap.org/risks.html`.

[16] Latanya Sweeney, Akua Abu, and Julia Winn. Identifying participants in the personal genome project by name. Data Privacy Lab, Institute for Quantitative Social Science, Harvard University, April 2013. URL `http://dataprivacylab.org/projects/pgp/`.

[17] Oded Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2004.

[18] Omkant Pandey and Yannis Rouselakis. Property preserving symmetric encryption. In David Pointcheval and Thomas Johansson, editors, *31st International Conference on the Theory and Applications of Cryptographic Techniques*, pages 375–391. Springer, 2012.

[19] Mohammad Shahriar Rahman, Anirban Basu, and Shinsaku Kiyomoto. Towards outsourced privacy-preserving multiparty DBSCAN. In *22nd Pacific Rim International Symposium on Dependable Computing*, pages 225–226. IEEE, 2017.

[20] Zekeriya Erkin, Thijs Veugen, Tomas Toft, and Reginald L Lagendijk. Privacy-preserving user clustering in a social network. *1st International Workshop on Information Forensics and Security*, pages 96–100, 2009. ISSN 978-1-4244-5280-4.

[21] Dongxi Liu, Elisa Bertino, and Xun Yi. Privacy of outsourced k-means clustering. In *9th Symposium on Information, Computer and Communications Security*, pages 123–134. ACM, 2014.

[22] Xiaoyan Liu, Zoe L Jiang, Siu-Ming Yiu, Xuan Wang, Chuting Tan, Ye Li, Zechao Liu, Yabin Jin, and Junbin Fang. Outsourcing two-party privacy preserving k-means clustering protocol in wireless sensor networks. In *11th International Conference on Mobile Ad-hoc and Sensor Networks*, pages 124–133. IEEE, 2015.

[23] Youwen Zhu, Rui Xu, and Tsuyoshi Takagi. Secure k-NN query on encrypted cloud database without key-sharing. *International Journal of Electronic Security and Digital Forensics*, 5(3-4):201–217, 2013.

[24] H. Hu, J. Xu, C. Ren, and B. Choi. Processing private queries over untrusted data cloud through privacy homomorphism. In *27th International Conference on Data Engineering (ICDE)*, pages 601–612. IEEE, 2011.

[25] Michael Beye, Zekeriya Erkin, and Reginald L Lagendijk. Efficient privacy preserving k-means clustering in a three-party setting. In *International Workshop on Information Forensics and Security*, pages 1–6. IEEE, 2011.

[26] Fang-Yu Rao, Bharath K Samanthula, Elisa Bertino, Xun Yi, and Dongxi Liu. Privacy-preserving and outsourced multi-user k-means clustering. *1st Conference on Collaboration and Internet Computing*, pages 80–89, 2015. ISSN 978-1-5090-0089-0.

[27] Carmit Hazay, Tomas Toft, Gert Læssøe Mikkelsen, and Tal Rabin. Efficient RSA key generation and threshold Paillier in the two-party setting. In *Cryptographersâ Track at the RSA Conference*, volume 7178 of *Lecture Notes in Computer Science*, pages 313–331. Springer, 2012.

[28] Jiawei Yuan and Shucheng Yu. Privacy preserving back-propagation neural network learning made practical with cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, 25(1):212–221, 2014.

[29] Oded Goldreich. Secure multi-party computation. *PhD thesis, Weizmann Institute of Science*, 1998.

[30] Andrew Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164. IEEE, 1982.

[31] Chris Clifton, Murat Kantarcioglu, Jaideep Vaidya, Xiaodong Lin, and Michael Y Zhu. Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):28–34, 2002.

[32] Wenliang Du, Mikhail J Atallah, and Eugene H Spafford. A study of several specific secure two-party computation problems. *PhD thesis, Purdue University, West Lafayette*, 2001.

[33] Deepti Mittal, Damandeep Kaur, and Ashish Aggarwal. Secure data mining in cloud using homomorphic encryption. In *2014 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pages 1–7. IEEE, 2014.

[34] Yanguang Shen, Junrui Han, and Huifang Shan. The research of privacy-preserving clustering algorithm. In *3rd International Symposium on Intelligent Information Technology and Security Informatics*, pages 324–327. IEEE, 2010.

[35] Mohamed Ouda, Sameh Salem, Ihab Ali, and El-Sayed Saad. Privacy-preserving data mining in homogeneous collaborative clustering. *International Arab Journal of Information Technology*, 12(6):604–612, 2015. ISSN 16833198.

[36] K Anil Kumar and C Pandu Rangan. Privacy preserving DBSCAN algorithm for clustering. In *3rd International Conference on Advanced Data Mining and Applications*, pages 57–68. Springer, 2007.

[37] Dongjie Jiang, Anrong Xue, Shiguang Ju, Weihe Chen, and Handa Ma. Privacy-preserving DBSCAN on horizontally partitioned data. In *International Symposium on IT in Medicine and Education (ITME)*, pages 1067–1072. IEEE, 2008.

[38] Somesh Jha, Luis Kruger, and Patrick McDaniel. Privacy preserving clustering. In *10th European Symposium on Research in Computer Security*, pages 397–417. Springer, 2005.

[39] Paul Bunn and Rafail Ostrovsky. Secure two-party k-means clustering. In *14th conference on Computer and Communications Security*, pages 486–497. ACM, 2007.

[40] Jinfei Liu, Li Xiong, Jun Luo, and Joshua Zhexue Huang. Privacy preserving distributed DBSCAN clustering. *Transactions on Data Privacy*, 6(1):69–85, 2013. ISSN 18885063.

[41] S. Xu, X. Cheng, S. Su, K. Xiao, and L. Xiong. Differentially private frequent sequence mining. *IEEE Transactions on Knowledge and Data Engineering*, 28 (11):2910–2926, 2016. ISSN 1041-4347.

[42] Li Liu, Murat Kantarcioglu, and Bhavani Thuraisingham. The applicability of the perturbation based privacy preserving data mining for real-world data. *Data and Knowledge Engineering*, 65(1):5–21, 2008.

[43] H. Xu, S. Guo, and K. Chen. Building confidential and efficient query services in the cloud with RASP data perturbation. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):322–335, 2014. ISSN 1041-4347.

[44] Man Lung Yiu, Ira Assent, Christian S Jensen, and Panos Kalnis. Outsourced similarity search on metric data assets. *IEEE Transactions on Knowledge and Data Engineering*, 24(2):338–352, 2012.

[45] Bharath K. Samanthula, Yousef Elmehdwi, and Wei Jiang. k-Nearest Neighbor classification over semantically secure encrypted relational data. *IEEE Transactions on Knowledge and Data Engineering*, 27(5):1261–1273, 2015. ISSN 1041-4347.

[46] James MacQueen. Some methods for classification and analysis of multivariate observations. In *5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297. Oakland, CA, USA, 1967.

[47] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996.

[48] Thomas Cover and Peter Hart. Nearest neighbour pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[49] Justin Zhan. Privacy-preserving collaborative data mining. *IEEE Computational Intelligence Magazine*, 3(2):31–41, 2008.

[50] M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

[51] Dongxi Liu and S. Wang. Nonlinear order preserving index for encrypted database query in service cloud environments. *Concurrency Computation: Practice and Experience*, 25(13):1967–1984, 2013. ISSN 15320626.

[52] Zheli Liu, Xiaofeng Chen, Jun Yang, Chunfu Jia, and Ilsun You. New order preserving encryption model for outsourced databases in cloud environments. *Journal of Network and Computer Applications*, 59:198–207, 2016. ISSN 1084-8045.

[53] Dongxi Liu. Homomorphic encryption for database querying. *Patent*, (2865127A4), 10 2015.

[54] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *16th International Conference on the Theory and Applications of Cryptographic Techniques*, pages 223–238. Springer, 1999.

[55] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *2000 SIGMOD International Conference on Management of Data*, pages 439–450. ACM, 2000.

[56] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. *Journal of Cryptology*, 15(3):177–206, 2002. ISSN 09332790.

[57] SR de M Oliveira and Osmar R Zaïane. Toward standardization in privacy-preserving data mining. In *2nd International Workshop on Data Mining Standards, Services and Platforms*, pages 7–17, 2004.

[58] Chris Clifton, Murat Kantarcioglu, and Jaideep Vaidya. Defining privacy for data mining. In *National science foundation workshop on next generation data mining*, pages 126–133. Citeseer, 2002.

[59] Alpa Shah and Ravi Gulati. Privacy preserving data mining: Techniques classification and implications- a survey. *International Journal of Computer Applications*, 137(12):40–46, 2016.

[60] Sam Smith. Data and privacy: Now you see me; new model for data sharing; modern governance and staticians. *Significance*, 11(4):10–17, October 2014.

[61] Jian Pei, Yufei Tao, Jiexing Li, and Xiaokui Xiao. Privacy preserving publishing on multiple Quasi-identifiers. In *25th International Conference on Data Engineering (ICDE)*, pages 1132–1135. IEEE, 2009.

[62] Khushbu Agrawal and Vandan Tewari. Privacy-preservation in collaborative association rule mining for outsourced data. *International Journal of Distributed and Cloud Computing*, 5(2):29–34, 2017. ISSN 23216840.

[63] Ali Gholami and Erwin Laure. Security and privacy of sensitive data in cloud computing: A survey of recent developments. *arXiv ePrint Archive Cornell University Library*, abs/1601.01498, 2016.

[64] Stefano Ceri, Mauro Negri, and Giuseppe Pelagatti. Horizontal data partitioning in database design. In *1982 SIGMOD International Conference on Management of Data*, pages 128–136. ACM, 1982.

[65] Shamkant Navathe, Stefano Ceri, Gio Wiederhold, and Jinglie Dou. Vertical partitioning algorithms for database design. *ACM Transactions on Database Systems*, 9(4):680–710, 1984. ISSN 15574644.

[66] Khaled El Emam, Elizabeth Jonker, Luk Arbuckle, and Bradley Malin. A systematic review of re-identification attacks on health data. *PLOS ONE*, 6(12):1–12, 12 2011.

[67] Kamalkumar Macwan and Sankita Patel. k-NMF anonymization in social network data publishing. *The Computer Journal*, 61(4):601–613, 02 2018. ISSN 0010-4620.

[68] Grigorios Loukides and Aris Gkoulalas-Divanis. Utility-preserving transaction data anonymization with low information loss. *Expert Systems with Applications*, 39(10):9764–9777, 2012.

[69] Benjamin CM Fung, Ke Wang, Lingyu Wang, and Patrick CK Hung. Privacy-preserving data publishing for cluster analysis. *Data and Knowledge Engineering*, 68(6):552–575, 2009.

[70] Ke Wang, Benjamin C. M. Fung, and Guozhu Dong. Integrating private databases for data analysis. In Paul Kantor, Gheorghe Muresan, Fred Roberts, Daniel D. Zeng, Fei-Yue Wang, Hsinchun Chen, and Ralph C. Merkle, editors, *International Conference on Intelligence and Security Informatics*, pages 171–182. Springer, 2005.

[71] Mark Last, Tamir Tassa, Alexandra Zhmudyak, and Erez Shmueli. Improving accuracy of classification models induced from anonymized datasets. *Information Sciences*, 256:138–161, 2014. ISSN 0020-0255.

[72] Benjamin CM Fung, Ke Wang, and S Yu Philip. Anonymizing classification data for privacy preservation. *IEEE Transactions on Knowledge and Data Engineering*, 19(5):711–725, May 2007. ISSN 1041-4347.

[73] Graham Cormode and Divesh Srivastava. Anonymized data: generation, models, usage. In *2009 SIGMOD International Conference on Management of Data*, pages 1015–1018. ACM, 2009.

[74] Dong Li, Xianmang He, LongBin Cao, and Huahui Chen. Permutation anonymization. *Journal of Intelligent Information Systems*, 47(3):427–445, 2016.

[75] Xiaokui Xiao and Yufei Tao. Anatomy: Simple and effective privacy preservation. In *32nd International Conference on Very Large Data Bases*, pages 139–150. ACM, 2006.

[76] Latanya Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):571–588, 2002.

[77] Pierangela Samarati and Latanya Sweeney. Generalizing data to provide anonymity when disclosing information. In *17th Symposium on Principles of Database Systems (PODS)*, volume 98, pages 188–201. ACM, 1998.

[78] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[79] Roberto J Bayardo and Rakesh Agrawal. Data privacy through optimal k-anonymization. In *21st International conference on data engineering (ICDE)*, pages 217–228. IEEE, 2005.

[80] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *23rd International Conference on Data Engineering (ICDE)*, pages 106–115. IEEE, 2007.

[81] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *22nd International Conference on Data Engineering (ICDE)*, pages 24–24. IEEE, 2006.

[82] Raymond Chi-Wing Wong, Jiuyong Li, Ada Wai-Chee Fu, and Ke Wang. (α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In *12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 754–759. ACM, 2006.

[83] Xiaokui Xiao and Yufei Tao. M-invariance: towards privacy preserving republication of dynamic datasets. In *2007 SIGMOD International Conference on Management of Data*, pages 689–700. ACM, 2007.

[84] Wei Jiang and Chris Clifton. Privacy-preserving distributed k-anonymity. In *19th Annual International Federation for Information Processing (IFIP)*, pages 166–177. Springer, 2005.

[85] Sheng Zhong, Zhiqiang Yang, and Rebecca N Wright. Privacy-enhancing k-anonymization of customer data. In *24th SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 139–147, 2005.

[86] M. Ercan Nergiz and Chris Clifton. Thoughts on k-anonymization. *Data and Knowledge Engineering*, 63(3):622–645, 2007. ISSN 0169-023X.

[87] Charu C. Aggarwal. On k-anonymity and the curse of dimensionality. In *31st International Conference on Very Large Data Bases*, pages 901–909. ACM, 2005.

[88] Jiuyong Li, Raymond Chi-Wing Wong, Ada Wai-Chee Fu, and Jian Pei. Anonymization by local recoding in data with attribute hierarchical taxonomies. *IEEE Transactions on Knowledge and Data Engineering*, 20(9):1181–1194, 2008.

[89] Slawomir Goryczka, Li Xiong, and Benjamin CM Fung. m-privacy for collaborative data publishing. *IEEE Transactions on Knowledge and Data Engineering*, 26(10): 2520–2533, 2013.

[90] Ardo van den Hout and Elsayed AH Elamir. Statistical disclosure control using post randomisation: Variants and measures for disclosure risk. *Journal of Official Statistics*, 22(4):711–731, 2006.

[91] J Kim and W Winkler. Multiplicative noise for masking continuous data. *Statistics*, 1, 2003.

[92] Keng-Pei Lin. Privacy-preserving kernel k-means clustering outsourcing with random transformation. *Knowledge and Information Systems*, 49(3):885–908, 2016. ISSN 02191377.

[93] Kanishka Bhaduri, Mark D Stefanski, and Ashok N Srivastava. Privacy-preserving outlier detection through random nonlinear data distortion. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(1):260–272, 2011. ISSN 1941-0492.

[94] Igor V Anikin and Rinat M Gazimov. Privacy preserving DBSCAN clustering algorithm for vertically partitioned data in distributed systems. In *2017 International Siberian Conference on Control and Communications*, pages 1–4. IEEE, 2017.

[95] Kun Liu, Hillol Kargupta, and Jessica Ryan. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):92–106, 2005.

[96] Nabil R Adam and John C Worthmann. Security-control methods for statistical databases: a comparative study. *ACM Computing Surveys (CSUR)*, 21(4):515–556, 1989.

[97] Tianqing Zhu, Gang Li, Wanlei Zhou, and S Yu Philip. Differentially private data publishing and analysis: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 29(8):1619–1638, 2017. ISSN 1041-4347.

[98] Keke Chen and Ling Liu. Privacy-preserving multiparty collaborative mining with geometric data perturbation. *IEEE Transactions on Parallel and Distributed Systems*, 20(12):1764–1776, 2009. ISSN 1045-9219.

[99] Claudio Orlandi, Alessandro Piva, and Mauro Barni. Oblivious neural network computing via homomorphic encryption. *EURASIP Journal on Information Security*, 2007(1):1–11, 2007.

[100] Zhengli Huang, Wenliang Du, and Biao Chen. Deriving private information from randomized data. In *2005 SIGMOD International Conference on Management of Data*, pages 37–48. ACM, 2005.

[101] Charu C Aggarwal. A general survey of privacy-preserving data mining models and algorithms. In Charu C Aggarwal and S Yu Philip, editors, *Privacy-Preserving Data Mining: Models and Algorithms*, chapter 2, pages 11–52. Springer Science and Business Media, 2008.

[102] Xu Wei-jiang, Huang Liu-sheng, Luo Yong-long, Yao Yi-fei, and Jing Wei-wei. Privacy-preserving DBSCAN clustering over vertically partitioned data. In *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE)*, pages 850–856. IEEE, 2007.

[103] Mark Shaneck, Yongdae Kim, and Vipin Kumar. Privacy preserving nearest neighbor search. *6th IEEE International Conference on Data Mining*, pages 541–545, 2006. ISSN 0-7695-2702-7.

[104] Christian Cachin. Efficient private bidding and auctions with an oblivious third party. In *6th ACM Conference on Computer and Communications Security*, pages 120–127. ACM, 1999.

[105] Qiuhui Tong, Xiu Li, and Bo Yuan. Efficient distributed clustering using boundary information. *Neurocomputing*, 275:2355–2366, 2018. ISSN 0925-2312.

[106] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *39th annual ACM Symposium on Theory of Computing*, pages 21–30. ACM, 2007.

[107] Adi Shamir. How to share a secret. *Communications ACM*, 22(11):612–613, November 1979. ISSN 0001-0782.

[108] George Robert Blakley et al. Safeguarding cryptographic keys. In *National Computer Conference*, pages 313–317, 1979.

[109] Dan Boneh and Matthew Franklin. Efficient generation of shared RSA keys. In *17th Annual International Cryptology Conference*, pages 425–439. Springer, 1997.

[110] Zekeriya Erkin, Thijs Veugen, Tomas Toft, and Reginald L Lagendijk. Privacy-preserving distributed clustering. *EURASIP Journal on Information Security*, 2013(1):1–15, 2013. ISSN 16874161.

[111] Mohammed Golam Kaosar, Russell Paulet, and Xun Yi. Fully homomorphic encryption based two-party association rule mining. *Data and Knowledge Engineering*, 76:1–15, 2012.

[112] Mauro Barni, Claudio Orlandi, and Alessandro Piva. A privacy-preserving protocol for neural-network-based computation. In *8th workshop on Multimedia and Security*, pages 146–151. ACM, 2006.

[113] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *33rd International Conference on Machine Learning (ICML)*, pages 201–210, 2016.

[114] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333. ACM, 2015.

[115] Dorothy Elizabeth Robling Denning. *Cryptography and data security*. Addison-Wesley Longman Publishing, 1982.

[116] Joppe W Bos, Kristin Lauter, and Michael Naehrig. Private predictive analysis on encrypted medical data. *Journal of Biomedical Informatics*, 50:234–243, 2014.

[117] Y. Li, W. Dai, Z. Ming, and M. Qiu. Privacy protection for preventing data over-collection in smart city. *IEEE Transactions on Computers*, 65(5):1339–1350, May 2016. ISSN 0018-9340.

[118] Hui Suo, Jiafu Wan, Caifeng Zou, and Jianqi Liu. Security in the internet of things: a review. In *2012 International Conference on Computer Science and Electronics Engineering (ICCSEE)*, volume 3, pages 648–651. IEEE, 2012.

[119] Dongxi Liu. Practical fully homomorphic encryption without noise reduction. *International Association for Cryptologic Research (IACR) Cryptology ePrint Archive*, 2015:468, 2015.

[120] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[121] Yin Hu. Improving the efficiency of homomorphic encryption schemes. *PhD thesis, Worcester Polytechnic Institute*, 2013.

[122] Aviad Kipnis and Eliphaz Hibshoosh. Efficient methods for practical fully homomorphic symmetric-key encrypton, randomization and verification. *International Association for Cryptologic Research (IACR) Cryptology ePrint Archive*, 2012:637, 2012.

[123] Ravi Shankar Dhakar, Amit Kumar Gupta, and Prashant Sharma. Modified RSA encryption algorithm (MREA). In *2nd International Conference on Advanced Computing and Communication Technologies*, pages 426–429. IEEE, 2012.

[124] Ronald Rivest, Len Adleman, and Michael Dertouzos. On data banks and privacy homomorphisms. In Richard A. DeMillo, editor, *Foundations of Secure Computation*, chapter 2, pages 169–180. Academic Press, 1978.

[125] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions On Computation Theory*, 6(3):13–48, 2014.

[126] Jing Li and Licheng Wang. Noise-free symmetric fully homomorphic encryption based on noncommutative rings. *International Association for Cryptologic Research (IACR) Cryptology ePrint Archive*, 2015:641, 2015.

[127] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[128] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *29th International Conference on the Theory and Applications of Cryptographic Techniques*, pages 24–43. Springer, 2010.

[129] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. *Society for Industrial and Applied Mathematics Journal on Computing (SICOMP)*, 43(2):831–871, 2014.

[130] Craig Gentry. A fully homomorphic encryption scheme. *PhD thesis, Stanford University Stanford*, 2009.

[131] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *32nd Annual Cryptology Conference*, pages 868–886. Springer, 2012.

[132] Joppe W Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In *14th IMA International Conference on Cryptography and Coding*, pages 45–64. Springer, 2013.

[133] Pedro Silveira Pisa, Michel Abdalla, and Otto Carlos Muniz Bandeira Duarte. Somewhat homomorphic encryption scheme for arithmetic operations on large integers. In *4th Global Information Infrastructure and Networking Symposium (GIIS)*, pages 1–8. IEEE, 2012.

[134] Yarkin Doröz and Berk Sunar. Flattening NTRU for evaluation key free homomorphic encryption. *International Association for Cryptologic Research (IACR) Cryptology ePrint Archive*, 2016:315, 2016.

[135] Koji Nuida. Candidate constructions of fully homomorphic encryption on finite simple groups without ciphertext noise. *International Association for Cryptologic Research (IACR) Cryptology ePrint Archive*, 2014.

[136] Gustavus J Simmons. Symmetric and asymmetric encryption. *ACM Computing Surveys (CSUR)*, 11(4):305–330, 1979.

[137] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *2nd Theory of Cryptography Conference*, pages 325–341. Springer, 2005.

[138] Nigel P Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *13th International Workshop on Public Key Cryptography*, pages 420–443. Springer, 2010.

[139] Masahiro Yagisawa. Fully homomorphic public-key encryption based on discrete logarithm problem. *International Association for Cryptologic Research (IACR) Cryptology ePrint Archive*, 2016:54, 2016.

[140] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21 (2):120–126, 1978.

[141] Ernest F Brickell and Yacov Yacobi. On privacy homomorphisms. In *4th International Conference on the Theory and Applications of Cryptographic Techniques*, pages 117–125. Springer, 1987.

[142] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *International Association for Cryptologic Research (IACR) Cryptology ePrint Archive*, 2012:144, 2012.

[143] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *33rd Annual Cryptology Conference*, pages 75–92. Springer, 2013.

[144] Tan Soo Fun and Azman Samsudin. A survey of homomorphic encryption for outsourced big data computation. *KSII Transactions on Internet and Information Systems*, 10(8):3826–3851, 2016. ISSN 22881468.

[145] Riza Aditya, Colin Boyd, Ed Dawson, and Kapali Viswanathan. Secure E-voting for preferential elections. In *2nd International Conference on Electronic Government*, pages 246–249. Springer, 2003.

[146] Marta Gomez-Barrero, Julian Fierrez, and Javier Galbally. Variable-length template protection based on homomorphic encryption with application to signature biometrics. In *4th International Workshop on Biometrics and Forensics (IWBF)*, pages 1–6. IEEE, 2016.

[147] Julien Bringer, Hervé Chabanne, Malika Izabachene, David Pointcheval, Qiang Tang, and Sébastien Zimmer. An application of the Goldwasser-Micali cryptosystem to biometric authentication. In *12th Australasian Conference on Information Security and Privacy*, pages 96–106. Springer, 2007.

[148] Raluca Ada Popa, Catherine Redfield, Nickolai Zeldovich, and Hari Balakrishnan. CryptDB: protecting confidentiality with encrypted query processing. In *23rd ACM Symposium on Operating Systems Principles*, pages 85–100. ACM, 2011.

[149] Martin R Albrecht, Pooya Farshim, Jean-Charles Faugere, and Ludovic Perret. Polly cracker, revisited. In *17th International Conference on the Theory and Application of Cryptology and Information Security*, pages 179–196. Springer, 2011.

[150] Jacob Alperin-Sheriff and Chris Peikert. Practical bootstrapping in quasilinear time. In *33rd Annual Cryptology Conference*, pages 1–20. Springer, 2013.

[151] Jacob Alperin Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In *34th Annual Cryptology Conference*, pages 297–314. Springer, 2014.

[152] Shai Halevi and Victor Shoup. Algorithms in HElib. In *34th Annual Cryptology Conference*, pages 554–571. Springer, 2014.

[153] Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. In *30th International Conference on the Theory and Applications of Cryptographic Techniques*, pages 129–148. Springer, 2011.

[154] Ana Costache and Nigel P Smart. Which ring based somewhat homomorphic encryption scheme is best? In *Cryptographers Track at the RSA Conference*, pages 325–340. Springer International Publishing, 2016.

[155] H. Kumarage, I. Khalil, A. Alabdulatif, Z. Tari, and X. Yi. Secure data analytics for cloud-integrated internet of things applications. *IEEE Cloud Computing*, 3(2): 46–56, Mar 2016. ISSN 2325-6095.

[156] William Kahan. IEEE standard 754 for binary floating-point arithmetic. *Lecture Notes on the Status of IEEE*, 754(94720-1776):11, 1996.

[157] Louis J. M. Aslett, Pedro M. Esperança, and Chris C. Holmes. Encrypted statistical machine learning: new privacy preserving methods. *arXiv ePrint Archive Cornell University Library*, 2015.

[158] Louis Jm Aslett, Pedro M Esperança, and Chris C Holmes. A review of homomorphic encryption and software tools for encrypted statistical machine learning. *arXiv ePrint Archive Cornell University Library*, 2015.

[159] Ehsan Hesamifard, Hassan Takabi, Mehdi Ghasemi, and Rebecca N Wright. Privacy-preserving Machine Learning as a Service. *Proceedings on Privacy Enhancing Technologies*, 2018(3):123–142, 2018.

[160] Anthony Meehan, Ryan KL Ko, and Geoff Holmes. Deep learning inferences with hybrid homomorphic encryption. *arXiv ePrint Archive Cornell University Library*, 2018.

[161] Liangliang Xiao, Osbert Bastani, and I-Ling Yen. An efficient homomorphic encryption protocol for multi-user systems. *International Association for Cryptologic Research (IACR) Cryptology ePrint Archive*, 2012:193, 2012.

[162] Yongge Wang. Octonion algebra and noise-free fully homomorphic encryption (FHE) schemes. *arXiv ePrint Archive Cornell University Library*, 2016.

[163] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2014.

[164] Christine Jost, Ha Lam, Alexander Maximov, and Ben Jm Smeets. Encryption performance improvements of the Paillier cryptosystem. *International Association for Cryptologic Research (IACR) Cryptology ePrint Archive*, 2015:864, 2015.

[165] Christine Tex, Martin Schäler, and Klemens Böhm. Towards meaningful distance-preserving encryption. In *30th International Conference on Scientific and Statistical Database Management*, pages 2–14. ACM, 2018.

[166] Stanley RM Oliveira and Osmar R Zaiane. Privacy preserving clustering by data transformation. *Journal of Information and Data Management*, 1(1):37–51, 2010.

[167] Wai Kit Wong, David Wai-lok Cheung, Ben Kao, and Nikos Mamoulis. Secure kNN computation on encrypted databases. In *2009 SIGMOD International Conference on Management of Data*, pages 139–152, New York, NY, USA, 2009. ACM.

[168] Kun Liu, Chris Giannella, and Hillol Kargupta. An attacker's view of distance preserving maps for privacy preserving data mining. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *10th European conference on Principle and Practice of Knowledge Discovery in Databases (PKDD)*, pages 297–308. Springer, 2006.

[169] E Onur Turgay, Thomas B Pedersen, Yücel Saygın, Erkay Savaş, and Albert Levi. Disclosure risks of distance preserving data transformations. In *20th International Conference on Scientific and Statistical Database Management*, pages 79–94. Springer, 2008.

[170] Youwen Zhu, Zhikuan Wang, and Yue Zhang. Secure k-NN query on encrypted cloud data with limited key-disclosure and offline data owner. In *10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 401–414. Springer, 2016.

[171] Google. The encrypted BigQuery client, 2015. URL `https://github.com/google/encrypted-bigquery-client`.

[172] Timo Schindler and Christoph Skornia. Secure parallel processing of big data using order-preserving encryption on Google BigQuery. *arXiv ePrint Archive Cornell University Library*, 2016.

[173] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order preserving encryption for numeric data. In *2004 SIGMOD International Conference on Management of Data*, pages 563–574. ACM, 2004.

[174] Tingjian Ge and Stan Zdonik. Fast, secure encryption for indexing in a column-oriented DBMS. In *23rd International Conference on Data Engineering (ICDE)*, pages 676–685. IEEE, 2007.

[175] Hakan Hacigümüş, Bala Iyer, Chen Li, and Sharad Mehrotra. Executing SQL over encrypted data in the database-service-provider model. In *2002 SIGMOD international conference on Management of data*, pages 216–227. ACM, 2002.

[176] Dongxi Liu and Shenlu Wang. Programmable order-preserving secure index for encrypted database query. In *5th International Conference on Cloud Computing, Cloud Computing*, pages 502–509. IEEE, 2012.

[177] Z. Xia, X. Wang, X. Sun, and Q. Wang. A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. *IEEE Transactions on Parallel and Distributed Systems*, 27(2):340–352, 2016. ISSN 1045-9219.

[178] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O Neill. Order-preserving symmetric encryption. In *28th International Conference on the Theory and Applications of Cryptographic Techniques*, pages 224–241. Springer, 2009.

[179] Dae Hyun Yum, Duk Soo Kim, Jin Seok Kim, Pil Joong Lee, and Sung Je Hong. Order-preserving encryption for non-uniformly distributed plaintexts. In *12th International Workshop on Information Security Applications*, pages 84–97. Springer, 2011.

[180] Raluca Ada Popa, Frank H Li, and Nickolai Zeldovich. An ideal-security protocol for order-preserving encoding. In *34th Symposium on Security and Privacy*, pages 463–477. IEEE, 2013.

[181] K Srinivasa Reddy and Sirandas Ramachandram. A novel dynamic order-preserving encryption scheme. In *1st International Conference on Networks and Soft Computing*, pages 92–96. IEEE, 2014.

[182] Savvas Savvides. Order-preserving encryption in Java. `https://github.com/ssavvides/jope`, 2015.

[183] Alexandra Boldyreva, Nathan Chenette, and Adam ONeill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *31st Annual Cryptology Conference*, pages 578–595. Springer, 2011.

[184] Liangliang Xiao and I-Ling Yen. Security analysis for order preserving encryption schemes. In *46th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE, 2012.

[185] Vladimir Kolesnikov and Abdullatif Shikfa. On the limits of privacy provided by order-preserving encryption. *Bell Labs Technical Journal*, 17(3):135–146, 2012.

[186] Cong Wang, Ning Cao, Kui Ren, and Wenjing Lou. Enabling secure and efficient ranked keyword search over outsourced cloud data. *IEEE Transactions on Parallel and Distributed Systems*, 23(8):1467–1479, 2011.

[187] Hasan Kadhem, Toshiyuki Amagasa, and Hiroyuki Kitagawa. A secure and efficient order preserving encryption scheme for relational databases. In *2nd International Conference on Knowledge Management and Information Sharing (KMIS)*, pages 25–35, 2010.

[188] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20: 53–65, 1987. ISSN 03770427.

[189] Belur V Dasarathy. *Nearest Neighbor (NN) norms: NN pattern classification techniques*. IEEE Computer society press, 1991.

[190] David Adedayo Adeniyi, Zhaoqiang Wei, and Y Yongquan. Automated web usage data mining and recommendation system using k-Nearest Neighbor (kNN) classification method. *Applied Computing and Informatics*, 12(1):90–108, 2016.

[191] Zhenyun Deng, Xiaoshu Zhu, Debo Cheng, Ming Zong, and Shichao Zhang. Efficient kNN classification algorithm for big data. *Neurocomputing*, 195:143–148, 2016.

[192] BL Deekshatulu, Priti Chandra, et al. Classification of heart disease using k-nearest neighbor and genetic algorithm. *Procedia Technology*, 10:85–94, 2013.

[193] Q Peter He and Jin Wang. Fault detection using the k-nearest neighbor rule for semiconductor manufacturing processes. *IEEE Transactions on Semiconductor Manufacturing*, 20(4):345–354, 2007.

[194] John Makhoul, Francis Kubala, Richard Schwartz, and Ralph Weischedel. Performance measures for information extraction. In *Proceedings of DARPA Broadcast News Workshop*, pages 249–252. Herndon, VA, Morgan Kaufmann, 1999.

[195] Fevzullah Temurtas, Ali Gulbag, and Nejat Yumusak. A study on neural networks using taylor series expansion of sigmoid activation function. In *4th International Conference on Computational Science and Its Applications*, pages 389–397. Springer, 2004.

[196] Payman Mohassel and Yupeng Zhang. SecureML: A system for scalable privacy-preserving machine learning. In *38th IEEE Symposium on Security and Privacy (SP)*, pages 19–38. IEEE, 2017.

[197] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, Institute for Cognitive Science, California University, San Diego, 1985.

[198] Thore Graepel, Kristin Lauter, and Michael Naehrig. ML confidential: machine learning on encrypted data. In *15th International Conference on Information Security and Cryptology*, pages 1–21. Springer, 2012.

[199] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. CryptoDL: Deep neural networks over encrypted data. *arXiv ePrint Archive Cornell University Library*, 2017.

[200] Bekir Karlik and A Vehbi Olgac. Performance analysis of various activation functions in generalized MLP architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122, 2011.

[201] Miran Kim, Yongsoo Song, and Jung Hee Cheon. Secure searching of biomarkers through hybrid homomorphic encryption scheme. *BMC Medical Genomics*, 10(2): 42, 2017.

[202] Henning Perl, Yassene Mohammed, Michael Brenner, and Matthew Smith. Fast confidential search for bio-medical data using bloom filters and homomorphic cryptography. In *8th International Conference on E-Science*, pages 1–8. IEEE, 2012.

[203] Thomas Shortell and Ali Shokoufandeh. Secure signal processing using fully homomorphic encryption. In Sebastiano Battiato, Jacques Blanc-Talon, Giovanni Gallo, Wilfried Philips, Dan Popescu, and Paul Scheunders, editors, *16th International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 93–104, Cham, 2015. Springer International Publishing.

[204] R. L. Lagendijk, Z. Erkin, and M. Barni. Encrypted signal processing for privacy protection: Conveying the utility of homomorphic encryption and multiparty computation. *IEEE Signal Processing Magazine*, 30(1):82–105, Jan 2013. ISSN 1053-5888.

[205] Vishnu Naresh Boddeti. Secure face matching using fully homomorphic encryption. In *9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–10. IEEE, 2018.

[206] Gizem S Çetin, Hao Chen, Kim Laine, Kristin Lauter, Peter Rindal, and Yuhou Xia. Private queries on encrypted genomic data. *BMC Medical Genomics*, 10(2): 45, 2017.

[207] Ming Li, Shucheng Yu, Ning Cao, and Wenjing Lou. Authorized private keyword search over encrypted data in cloud computing. In *31st International Conference on Distributed Computing Systems*, pages 383–392. IEEE, 2011.

[208] Dan Boneh, Craig Gentry, Shai Halevi, Frank Wang, and David J Wu. Private database queries using somewhat homomorphic encryption. In *11th International Conference on Applied Cryptography and Network Security*, pages 102–118. Springer, 2013.

[209] X. Yi, R. Paulet, E. Bertino, and V. Varadharajan. Practical k nearest neighbor queries with location privacy. In *30th International Conference on Data Engineering (ICDE)*, pages 640–651. IEEE, March 2014.

[210] Jens Mathias Bohli, Dan Dobre, Ghassan O Karame, and Wenting Li. PrivLoc: preventing location tracking in geofencing services. In *7th International Conference on Trust and Trustworthy Computing*, pages 143–160. Springer, 2014.

[211] Mary R Schurgot, David A Shinberg, and Lloyd G Greenwald. Experiments with security and privacy in IoT networks. In *16th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–6. IEEE, 2015.

[212] Christoph Bösch. An efficient privacy-preserving outsourced geofencing service using bloom filter. In *2018 IEEE Vehicular Networking Conference (VNC)*, pages 1–8. IEEE, 2018.

Appendix A

Modified Liu's Scheme: Proof of Correctness

A.1 Overview

This appendix presents mathematical proofs of correctness for the MLS algorithms presented in Chapter 9. The appendix provides proofs for: (i) the scheme encryption and decryption algorithms (Algorithm 30 and 31), (ii) the dimensionality reduction algorithm (Algorithm 32) and (iii) the secure data comparison. In the interest of simplicity of the mathematical proofs, the number of MLS sub-cyphertexts used in the proofs was set to 3 ($m = 3$) and the first sub-cypher was used to preserve the data order ($q = 1$). However, the proofs can, of course, be generalise for different numbers of sub-cyphertexts.

The appendix is organised as follows. Sub-appendix A.2 presents the correctness of the data encryption and decryption algorithms. This is followed by Sub-appendix A.3 that presents mathematical proofs of the dimensionality reduction algorithm that reduces the number of sub-cyphertexts after each HE multiplication using the concept of *trapdoors*. Finally, Sub-section A.4 presents the proof of secure data comparison.

A.2 MLS Encryption and Decryption Algorithms

The proposed MLS FHE scheme was a modification of Liu's scheme. This sub-appendix presents proofs of the correctness of this modification in that it should be the case that cyphertext generated using Algorithm 30 can be reversed to plaintext using the decryption algorithm (Algorithm 31). As noted, the MLS examples presented in this Appendix considers $m = 3$, therefore a plaintext value v is encrypted to cyphertext $E = \{e_1, e_2, e_3\}$. According to the prescribed criteria for key generation (Sub-section 9.2.1 given in Chapter 9) there exists only one element q ($1 \leqslant q < m$) such that $t_q \neq 0$. In the MLS used in this Appendix $q = 1$, thus $t_1 \neq 0$ and $t_2 = t_3 = 0$. The values of each sub-cyphertext is calculated as:

$$\begin{aligned}
Encrypt(v, SK(3)) = &\ \{e_1, e_2, e_3\}\ where \\
e_1 = &\ \frac{k_1 \times t_1 \times v + s_1 + k_1 \times (r_1 - r_2)}{s_1} = \frac{k_1 \times t_1 \times v + s_1 + k_1 \times r_1 - k_1 \times r_2}{s_1} \\
e_2 = &\ \frac{k_2 \times t_2 \times v + s_2 + k_2 \times (r_2 - r_1)}{s_2} = \frac{s_2 + k_2 \times r_2 - k_2 \times r_1}{s_2} \\
e_3 = &\ (k_3 + s_3 + t_3) = kst
\end{aligned}$$

To decrypt this cyphertext, the steps outlined in Algorithm 31 are followed as shown in Equation A.1. According to the prescribed criteria for key generation $t_1 \neq 0$ and $t_2 = 0$ thus $t = t_1$.

$$\begin{aligned} t &= t_1 \\ s &= \frac{e_m}{k_m + s_m + t_m} = \frac{kst}{kst} = 1 \\ v &= \frac{\frac{(e_1 \times s_1) - (s \times s_1)}{k_1} + \frac{(e_2 \times s_2) - (s \times s_2)}{k_2}}{t_1} \\ &= \frac{\frac{(e_1 \times s_1) - s_1}{k_1} + \frac{(e_2 \times s_2) - s_2}{k_2}}{t_1} \quad where\ s = 1 \\ &= \frac{1}{t_1} \times \left(\boldsymbol{\frac{s_1}{k_1} \times (e_1 - 1) + \frac{s_2}{k_2} \times (e_2 - 1)} \right) \end{aligned} \tag{A.1}$$

Note that the value of s is not equal to 1 when the cyphertexts are reduced, as the last cyphertext (e_m) is changed when the dimensionality reduction algorithm is applied. The values of $\frac{s_1}{k_1} \times (e_1 - 1)$ and $\frac{s_2}{k_2} \times (e_2 - 1)$ are as calculated as demonstrated by Equations A.2 and A.3:

$$\begin{aligned} \frac{s_1}{k_1} \times (e_1 - 1) &= \frac{s_1}{k_1} \times \left(\frac{k_1 \times t_1 \times v + s_1 + k_1 \times r_1 - k_1 \times r_2}{s_1} - 1 \right) \\ &= \frac{s_1}{k_1} \times \left(\frac{k_1 \times t_1 \times v + s_1 + k_1 \times r_1 - k_1 \times r_2 - s_1}{s_1} \right) = \frac{k_1 \times t_1 \times v + k_1 \times r_1 - k_1 \times r_2}{k_1} \end{aligned} \tag{A.2}$$

$$\begin{aligned} \frac{s_2}{k_2} \times (e_2 - 1) &= \frac{s_2}{k_2} \times \left(\frac{s_2 + k_2 \times r_2 - k_2 \times r_1}{s_2} - 1 \right) = \frac{s_2}{k_2} \times \left(\frac{s_2 + k_2 \times r_2 - k_2 \times r_1 - s_2}{s_2} \right) \\ &= \frac{k_2 \times r_2 - k_2 \times r_1}{k_2} \end{aligned} \tag{A.3}$$

Equations A.2 and A.3 are then used to calculate the decryption in Equation A.1 as follows:

$$\begin{aligned} v &= \frac{1}{t_1} \times \left(\frac{k_1 \times t_1 v + k_1 \times r_1 - k_1 \times r_2}{k_1} + \frac{k_2 \times r_2 - k_2 \times r_1}{k_2} \right) \\ &= \frac{k_1 \times t_1 \times v + k_1 \times r_1 - k_1 \times r_2}{k_1 \times t_1} + \frac{k_2 \times r_2 - k_2 \times r_1}{k_2 \times t_1} \\ &= v + \frac{r_1}{t_1} - \frac{r_2}{t_1} + \frac{r_2}{t_1} - \frac{r_1}{t_1} \\ &= v \end{aligned} \tag{A.4}$$

The result of decrypting the cyphertexts using Algorithm 31 is the value of plaintext v.

A.3 Dimensionality Reduction Algorithm

As in the case of Liu's original FHE scheme, MLS performs cyphertext multiplication by determining the outer product of the two cyphertexts to be multiplied. Therefore the number of sub-cyphertexts is increased from m to m^2 for one multiplication. Equation A.5 shows the HE multiplication of two cyphertexts E_1 (encrypt the value of v_1) and E_2 (encrypt the value of v_2), both encrypted using MLS to $E_1 = \{e_{1_1}, \ldots, e_{1_m}\}$ and $E_2 = \{e_{2_1}, \ldots, e_{2_m}\}$, respectively. In MLS, the number of sub-cyphertext of a generated cyphertext can be "reduced" back to m using a *trapdoor* concept which re-encrypts the cyphertext (without prior decryption). This process is referred to as "dimensionality reduction". This sub-appendix presents the correctness of the dimensionality reduction, Algorithm 32 called after each HE multiplication. The correctness is evaluated by multiplying two cyphertext using the HE properties ($\otimes$) and then applying dimensionality reduction (Algorithm 32) to reduce the number of sub-cyphertexts. The resulting cyphertext, from Algorithm 32, when decrypted should match the standard (or basic) multiplication to demonstrate that the dimensionality reduction is operating correctly.

$$E_1 \otimes E_2 = \{e_{1_1} \times e_{2_1}, \ldots, e_{1_1} \times e_{2_m}, \ldots e_{1_m} \times e_{2_1}, \ldots, e_{1_m} \times e_{2_m}\} \tag{A.5}$$

Assuming a MLS where $m = 3$ and $q = 1$. The sub-cyphertexts for encrypting v_1 and v_2, using Algorithm 30, are calculated as follows:

$$\begin{aligned} Encrypt(v_1, SK(3)) = &\ \{e_{1_1}, e_{1_2}, e_{1_3}\}\ where \\ e_{1_1} = &\ \tfrac{k_1 t_1 v_1 + s_1 + k_1(r_1 - r_2)}{s_1} = \tfrac{k_1 t_1 v_1 + s_1 + k_1 r_1 - k_1 r_2}{s_1} \\ e_{1_2} = &\ \tfrac{k_2 t_2 v_1 + s_2 + k_2(r_2 - r_1)}{s_2} = \tfrac{s_2 + k_2 r_2 - k_2 r_1}{s_2} \\ e_{1_3} = &\ (k_3 + s_3 + t_3) = kst \end{aligned} \tag{A.6}$$

$$\begin{aligned} Encrypt(v_2, SK(3)) = &\ \{e_{2_1}, e_{2_2}, e_{2_3}\}\ where \\ e_{2_1} = &\ \tfrac{k_1 t_1 v_2 + s_1 + k_1(r_1 - r_2)}{s_1} = \tfrac{k_1 t_1 v_2 + s_1 + k_1 r_1 - k_1 r_2}{s_1} \\ e_{2_2} = &\ \tfrac{k_2 t_2 v_2 + s_2 + k_2(r_2 - r_1)}{s_2} = \tfrac{s_2 + k_2 r_2 - k_2 r_1}{s_2} \\ e_{2_3} = &\ (k_3 + s_3 + t_3) = kst \end{aligned} \tag{A.7}$$

Multiplying $Encrypt(v_1, SK(3))$ and $Encrypt(v_2, SK(3))$ gives $E_1 \otimes E_2 = E = \{e_{1_1}e_{2_1}$, $e_{1_1}e_{2_2}$, $e_{1_1}e_{2_3}$, $e_{1_2}e_{2_1}$, $e_{1_2}e_{2_2}$, $e_{1_2}e_{2_3}$, $e_{1_3}e_{2_1}$, $e_{1_3}e_{2_2}$, $e_{1_3}e_{2_3}\}$. Using the proposed dimensionality reduction algorithm (Algorithm 32) the resulting cypher E can be re-encrypted without being first decrypted whilst reducing the number of sub-cyphertexts to only 3 sub-cyphertexts; when decrypted this should give $v_1 \times v_2$. Following the steps outlined in Algorithm 32, every consecutive three sub-cyphers in E will be associated with a trapdoor in $Trap$ which will be used to calculate one cypher in the reduced cypher $RE = \{re_1, re_2, re_3\}$. For example, re_1 is calculated from $\{e_{1_1}e_{2_1}, e_{1_1}e_{2_2}, e_{1_1}e_{2_3}\}$. Applying Algorithm 32, the reduced sub-cyphers will be as follows:

$$\begin{aligned} re_1 = &\ (((e_{1_1}e_{2_1}) - (\tfrac{e_{1_1}e_{2_3}}{kst})) \times trap_1) + (((e_{1_1}e_{2_2}) - (\tfrac{e_{1_1}e_{2_3}}{kst})) \times trap_2) \\ re_2 = &\ (((e_{1_2}e_{2_1}) - (\tfrac{e_{1_2}e_{2_3}}{kst})) \times trap_1) + (((e_{1_2}e_{2_2}) - (\tfrac{e_{1_2}e_{2_3}}{kst})) \times trap_2) \\ re_3 = &\ (((e_{1_3}e_{2_1}) - (\tfrac{e_{1_3}e_{2_3}}{kst})) \times trap_1) + (((e_{1_3}e_{2_2}) - (\tfrac{e_{1_3}e_{2_3}}{kst})) \times trap_2) \end{aligned}$$

According to Trapdoor calculation algorithm (Algorithm 29) the values for $trap_1$ and $trap_2$ are $\frac{s_1 \times secretK}{secretS \times k_1}$ and $\frac{s_2 \times secretK}{secretS \times k_2}$; and the value for kst will be $k_3 + s_3 + t_3$. After calling cypher reduction, the level of RE is one ($RE.l = 1$). The RE cypher can then be decrypted using Algorithm 31 to give $v_1 \times v_2$. Following Algorithm 31, as the cyphertext level value in RE is not equal to zero, a new sub-cyphertext value for each sub-cypher re_i in RE is calculated (lines 3 to 5) the new RE sub-cyphertexts are:

$$\begin{aligned} re_1 = &\ \tfrac{re_1 \times \frac{secretS}{secretK}}{t} = \tfrac{1}{t}\left[\tfrac{s_1}{k_1}(e_{1_1}e_{2_1} - \tfrac{e_{1_1}e_{2_3}}{kst}) + \tfrac{s_2}{k_2}(e_{1_1}e_{2_2} - \tfrac{e_{1_1}e_{2_3}}{kst})\right] \\ re_2 = &\ \tfrac{re_2 \times \frac{secretS}{secretK}}{t} = \tfrac{1}{t}\left[\tfrac{s_1}{k_1}(e_{1_2}e_{2_1} - \tfrac{e_{1_2}e_{2_3}}{kst}) + \tfrac{s_2}{k_2}(e_{1_2}e_{2_2} - \tfrac{e_{1_2}e_{2_3}}{kst})\right] \\ re_3 = &\ \tfrac{re_3 \times \frac{secretS}{secretK}}{t} = \tfrac{1}{t}\left[\tfrac{s_1}{k_1}(e_{1_3}e_{2_1} - \tfrac{e_{1_3}e_{2_3}}{kst}) + \tfrac{s_2}{k_2}(e_{1_3}e_{2_2} - \tfrac{e_{1_3}e_{2_3}}{kst})\right] \end{aligned}$$

The resulting cyphers are then used, in lines 7 to 9 of Algorithm 31, and processed as follows:

$$
\begin{aligned}
t = &\quad t_1 + t_2 = t_1 + 0 = t_1 \\
s = &\quad \tfrac{re_3}{(k_3+s_3+t_3)} = \tfrac{re_3}{kst} \\
v = &\quad \frac{\frac{((re_1 \times s_1)-(\frac{re_3}{kst} \times s_1))}{k1} + \frac{((re_2 \times s_2)-(\frac{re_3}{kst} \times s_2))}{k2}}{t} \\
= &\quad \frac{\frac{s_1 \times (re_1 - \frac{re_3}{kst})}{k1} + \frac{s_2 \times (re_2 - \frac{re_3}{kst})}{k2}}{t} \\
= &\quad \tfrac{1}{t}[\tfrac{s_1}{k_1} \times (re_1 - \boldsymbol{\frac{re_3}{kst}}) + \tfrac{s_2}{k_2} \times (re_2 - \boldsymbol{\frac{re_3}{kst}})]
\end{aligned}
\tag{A.8}
$$

The values for $\frac{re_3}{kst}$, re_1 and re_2 must then be calculated.
The values for $\frac{re_3}{kst}$ are given by:

$$
\begin{aligned}
\tfrac{re_3}{kst} = &\quad \tfrac{1}{t}[\tfrac{s_1}{k_1 kst}(e_{1_3}e_{2_1} - \tfrac{e_{1_3}e_{2_3}}{kst}) + \tfrac{s_2}{k_2 kst}(e_{1_3}e_{2_2} - \tfrac{e_{1_3}e_{2_3}}{kst})] \\
= &\quad \tfrac{1}{t}[\tfrac{s_1 e_{1_3} e_{2_1}}{k_1 kst} - \tfrac{s_1 e_{1_3} e_{2_3}}{k_1 kst^2} + \tfrac{s_2 e_{1_3} e_{2_2}}{k_2 kst} - \tfrac{s_2 e_{1_3} e_{2_3}}{k_2 kst^2}]
\end{aligned}
$$

Where:

$$
\begin{aligned}
\tfrac{s_1 e_{1_3} e_{2_1}}{k_1 kst} = &\quad \tfrac{s_1}{k_1 kst}(kst)(\tfrac{k_1 t_1 v_2 + s_1 + k_1 r_1 - k_1 r_2}{s_1}) \quad \textit{Equations A.6 and A.7} \\
= &\quad \tfrac{k_1 t_1 v_2 + s_1 + k_1 r_1 - k_1 r_2}{k_1} = t_1 v_2 + \tfrac{s_1}{k_1} + r_1 - r_2 \\
\tfrac{s_1 e_{1_3} e_{2_3}}{k_1 kst^2} = &\quad \tfrac{s_1}{k_1 kst^2} kst\ kst = \tfrac{s_1}{k_1} \\
\tfrac{s_2 e_{1_3} e_{2_2}}{k_2 kst} = &\quad \tfrac{s_2}{k_2 kst} e_{1_3} e_{2_2} = \tfrac{s_2}{k_2 kst} kst(\tfrac{s_2 + k_2 r_2 - k_2 r_1}{s_2}) \\
= &\quad \tfrac{s_2 + k_2 r_2 - k_2 r_1}{k_2} = \tfrac{s_2}{k_2} + r_2 - r_1 \\
\tfrac{s_2 e_{1_3} e_{2_3}}{k_2 kst^2} = &\quad \tfrac{s_2}{k_2 kst^2} kst\ kst = \tfrac{s_2}{k_2}
\end{aligned}
$$

In other words:

$$
\begin{aligned}
\tfrac{re_3}{kst} = &\quad \tfrac{1}{t}[t_1 v_2 + \tfrac{s_1}{k_1} + r_1 - r_2 - \tfrac{s_1}{k_1} + \tfrac{s_2}{k_2} + r_2 - r_1 - \tfrac{s_2}{k_2}] = \tfrac{1}{t} t_1 v_2 \\
= &\quad v_2 \quad \textit{where } t = t_1
\end{aligned}
$$

The values for re_1 and re_2 (in Equation A.8) are given by:

$$
re_1 = \quad \tfrac{1}{t}[\tfrac{s_1 e_{1_1} e_{2_1}}{k_1} - \tfrac{s_1 e_{1_1} e_{2_3}}{k_1 kst} + \tfrac{s_2 e_{1_1} e_{2_2}}{k_2} - \tfrac{s_2 e_{1_1} e_{2_3}}{k_2 kst}]
$$

The operands in the above are calculated, retrospectively, as follows:

$$
\begin{aligned}
\tfrac{s_1 e_{1_1} e_{2_1}}{k_1} = &\quad \tfrac{s_1}{k_1}[(\tfrac{k_1 t_1 v_1 + s_1 + k_1 r_1 - k_1 r_2}{s_1})(\tfrac{k_1 t_1 v_2 + s_1 + k_1 r_1 - k_1 r_2}{s_1})] \\
= &\quad \tfrac{s_1}{k_1}[(\tfrac{k_1 t_1 v_1}{s_1} + 1 + \tfrac{k_1 r_1}{s_1} - \tfrac{k_1 r_2}{s_1})(\tfrac{k_1 t_1 v_2}{s_1} + 1 + \tfrac{k_1 r_1}{s_1} - \tfrac{k_1 r_2}{s_1})] \\
= &\quad \tfrac{s_1}{k_1}[\tfrac{k_1^2 t_1^2 v_1 v_2}{s_1^2} + \tfrac{k_1 t_1 v_1}{s_1} + \tfrac{k_1^2 t_1 r_1 v_1}{s_1^2} - \tfrac{k_1^2 t_1 r_2 v_1}{s_1^2} + \tfrac{k_1 t_1 v_2}{s_1} + 1 + \tfrac{k_1 r_1}{s_1} - \tfrac{k_1 r_2}{s_1} \\
&\quad + \tfrac{k_1^2 t_1 r_1 v_2}{s_1^2} + \tfrac{k_1 r_1}{s_1} + \tfrac{k_1^2 r_1^2}{s_1^2} - \tfrac{k_1^2 r_1 r_2}{s_1^2} - \tfrac{k_1^2 t_1 r_2 v_2}{s_1^2} - \tfrac{k_1 r_2}{s_1} - \tfrac{k_1^2 r_1 r_2}{s_1^2} + \tfrac{k_1^2 r_2^2}{s_1^2}] \\
= &\quad \tfrac{s_1}{k_1}[\tfrac{k_1^2 t_1^2 v_1 v_2 + k_1 t_1 s_1 v_1 + k_1^2 t_1 r_1 v_1 - k_1^2 t_1 r_2 v_1 + k_1 t_1 s_1 v_2 + s_1^2 + k_1 r_1 s_1 - k_1 r_2 s_1 + k_1^2 t_1 r_1 v_2}{s_1^2} \\
&\quad \tfrac{+ k_1 r_1 s_1 + k_1^2 r_1^2 - k_1^2 r_1 r_2 - k_1^2 t_1 r_2 v_2 - k_1 r_2 s_1 - k_1^2 r_1 r_2 + k_1^2 r_2^2}{s_1^2}]
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{k_1^2t_1^2v_1v_2+k_1t_1s_1v_1+k_1^2t_1r_1v_1-k_1^2t_1r_2v_1+k_1t_1s_1v_2+s_1^2+k_1r_1s_1-k_1r_2s_1+k_1^2t_1r_1v_2}{s_1k_1} \\
&\quad \frac{+k_1r_1s_1+k_1^2r_1^2-k_1^2r_1r_2-k_1^2t_1r_2v_2-k_1r_2s_1-k_1^2r_1r_2+k_1^2r_2^2}{s_1k_1} \\
&= \frac{k_1t_1^2v_1v_2}{s_1}+t_1v_1+\frac{k_1t_1r_1v_1}{s_1}-\frac{k_1t_1r_2v_1}{s_1}+t_1v_2+\frac{s_1}{k_1}+r_1-r_2+\frac{k_1t_1r_1v_2}{s_1} \\
&\quad +r_1+\frac{k_1r_1^2}{s_1}-\frac{k_1r_1r_2}{s_1}-\frac{k_1t_1r_2v_2}{s_1}-r_2-\frac{k_1r_1r_2}{s_1}+\frac{k_1r_2^2}{s_1} \\
&= \frac{k_1t_1^2v_1v_2}{s_1}+t_1v_1+\frac{k_1t_1r_1v_1}{s_1}-\frac{k_1t_1r_2v_1}{s_1}+t_1v_2+\frac{s_1}{k_1}+2r_1-2r_2+\frac{k_1t_1r_1v_2}{s_1} \\
&\quad +\frac{k_1r_1^2}{s_1}-2\frac{k_1r_1r_2}{s_1}-\frac{k_1t_1r_2v_2}{s_1}+\frac{k_1r_2^2}{s_1} \\
\frac{s_1e_{1_1}e_{2_3}}{k_1kst} &= \frac{s_1}{k_1kst}\left(\frac{k_1t_1v_1+s_1+k_1r_1-k_1r_2}{s_1}\right)(kst) \\
&= \frac{1}{k_1}(k_1t_1v_1+s_1+k_1r_1-k_1r_2) \\
&= t_1v_1+\frac{s_1}{k_1}+r_1-r_2 \\
\frac{s_2e_{1_1}e_{2_2}}{k_2} &= \frac{s_2}{k_2}\left(\frac{k_1t_1v_1+s_1+k_1r_1-k_1r_2}{s_1}\right)\left(\frac{s_2+k_2r_2-k_2r_1}{s_2}\right) \\
&= \frac{s_2}{k_2}\left[\left(\frac{k_1t_1v_1}{s_1}+1+\frac{k_1r_1}{s_1}-\frac{k_1r_2}{s_1}\right)\left(1+\frac{k_2r_2}{s_2}-\frac{k_2r_1}{s_2}\right)\right] \\
&= \frac{s_2}{k_2}\left[\frac{k_1t_1v_1}{s_1}+\frac{k_1k_2t_1r_2v_1}{s_1s_2}-\frac{k_1k_2t_1r_1v_1}{s_1s_2}+1+\frac{k_2r_2}{s_2}-\frac{k_2r_1}{s_2}+\frac{k_1r_1}{s_1}+\frac{k_1k_2r_1r_2}{s_1s_2}\right. \\
&\quad \left.-\frac{k_1k_2r_1^2}{s_1s_2}-\frac{k_1r_2}{s_1}-\frac{k_1k_2r_2^2}{s_1s_2}+\frac{k_1k_2r_1r_2}{s_1s_2}\right] \\
&= \frac{k_1t_1s_2v_1}{s_1k_2}+\frac{k_1t_1r_2v_1}{s_1}-\frac{k_1t_1r_1v_1}{s_1}+\frac{s_2}{k_2}+r_2-r_1+\frac{k_1r_1s_2}{s_1k_2}+2\frac{k_1r_1r_2}{s_1} \\
&\quad -\frac{k_1r_1^2}{s_1}-\frac{k_1r_2s_2}{s_1k_2}-\frac{k_1r_2^2}{s_1} \\
\frac{s_2e_{1_1}e_{2_3}}{k_2kst} &= \frac{s_2}{k_2kst}\left(\frac{k_1t_1v_1+s_1+k_1r_1-k_1r_2}{s_1}\right)(kst) \\
&= \left(\frac{k_1t_1s_2v_1+s_1s_2+k_1s_2r_1-k_1s_2r_2}{s_1k_2}\right) \\
&= \frac{k_1t_1s_2v_1}{s_1k_2}+\frac{s_2}{k_2}+\frac{k_1s_2r_1}{s_1k_2}-\frac{k_1s_2r_2}{s_1k_2}
\end{aligned}
$$

In other words (recall $t = t_1$):

$$
\begin{aligned}
re_1 = &\ \frac{1}{t}\left[\frac{k_1t_1^2v_1v_2}{s_1}+t_1v_1+\frac{k_1t_1r_1v_1}{s_1}-\frac{k_1t_1r_2v_1}{s_1}+t_1v_2+\frac{s_1}{k_1}+2r_1-2r_2+\frac{k_1t_1r_1v_2}{s_1}\right. \\
&+\frac{k_1r_1^2}{s_1}-2\frac{k_1r_1r_2}{s_1}-\frac{k_1t_1r_2v_2}{s_1}+\frac{k_1r_2^2}{s_1}-t_1v_1-\frac{s_1}{k_1}-r_1+r_2+\frac{k_1t_1s_2v_1}{s_1k_2} \\
&+\frac{k_1t_1r_2v_1}{s_1}-\frac{k_1t_1r_1v_1}{s_1}+\frac{s_2}{k_2}+r_2-r_1+\frac{k_1r_1s_2}{s_1k_2}+2\frac{k_1r_1r_2}{s_1}-\frac{k_1r_1^2}{s_1} \\
&\left.-\frac{k_1r_2s_2}{s_1k_2}-\frac{k_1r_2^2}{s_1}-\frac{k_1t_1s_2v_1}{s_1k_2}-\frac{s_2}{k_2}-\frac{k_1s_2r_1}{s_1k_2}+\frac{k_1s_2r_2}{s_1k_2}\right] \\
= &\ \frac{k_1t_1v_1v_2}{s_1}+v_2+\frac{k_1r_1v_2}{s_1}-\frac{k_1r_2v_2}{s_1}
\end{aligned}
\tag{A.9}
$$

Recall that $t = t_1 + t_2$, however, as the key generation conditions require that there is only one $t_q \neq 0$ that is t_1 thus $t = t_1$. The value for re_2 (in Equation A.8) is then given by:

$$
\begin{aligned}
re_2 = &\ \frac{1}{t}\left[t_1v_2+\frac{s_1}{k_1}+r_1-r_2+\frac{t_1k_2r_2v_2}{s_2}+\frac{s_1k_2r_2}{k_1s_2}+2\frac{k_2r_1r_2}{s_2}-\frac{k_2r_2^2}{s_2}\right. \\
&-\frac{k_2t_1r_1v_2}{s_2}-\frac{s_1k_2r_1}{k_1s_2}-\frac{k_2r_1^2}{s_2}-\frac{s_1}{k_1}-\frac{s_1k_2r_2}{k_1s_2}+\frac{s_1k_2r_1}{k_1s_2}+\frac{s_2}{k_2} \\
&\left.+2r_2-2r_1+\frac{k_2r_2^2}{s_2}-2\frac{k_2r_1r_2}{s_2}+\frac{k_2r_1^2}{s_2}-\frac{s_2}{k_2}-r_2+r_1\right] \\
= &\ \frac{1}{t}\left[t_1v_2+\frac{t_1k_2r_2v_2}{s_2}-\frac{k_2t_1r_1v_2}{s_2}\right] \\
= &\ v_2+\frac{k_2r_2v_2}{s_2}-\frac{k_2r_1v_2}{s_2}
\end{aligned}
$$

Where:

$$\begin{aligned}
\frac{s_1 e_{1_2} e_{2_1}}{k_1} &= \frac{s_1}{k_1}[(\frac{s_2+k_2r_2-k_2r_1}{s_2})(\frac{k_1t_1v_2+s_1+k_1r_1-k_1r_2}{s_1})] \\
&= \frac{s_1}{k_1}[(1+\frac{k_2r_2}{s_2}-\frac{k_2r_1}{s_2})(\frac{k_1t_1v_2}{s_1}+1+\frac{k_1r_1}{s_1}-\frac{k_1r_2}{s_1})] \\
&= \frac{s_1}{k_1}[\frac{k_1t_1v_2}{s_1}+1+\frac{k_1r_1}{s_1}-\frac{k_1r_2}{s_1}+\frac{t_1k_1k_2r_2v_2}{s_1s_2}+\frac{k_2r_2}{s_2}+\frac{k_1k_2r_1r_2}{s_1s_2}-\frac{k_1k_2r_2^2}{s_1s_2} \\
&\quad -\frac{k_1k_2t_1r_1v_2}{s_1s_2}-\frac{k_2r_1}{s_2}-\frac{k_1k_2r_1^2}{s_1s_2}+\frac{k_1k_2r_1r_2}{s_1s_2}] \\
&= t_1v_2+\frac{s_1}{k_1}+r_1-r_2+\frac{t_1k_2r_2v_2}{s_2}+\frac{s_1k_2r_2}{k_1s_2}+2\frac{k_2r_1r_2}{s_2}-\frac{k_2r_2^2}{s_2} \\
&\quad -\frac{k_2t_1r_1v_2}{s_2}-\frac{s_1k_2r_1}{k_1s_2}-\frac{k_2r_1^2}{s_2}
\end{aligned}$$

$$\begin{aligned}
\frac{s_1 e_{1_2} e_{2_3}}{k_1kst} &= \frac{s_1}{k_1kst}(\frac{s_2+k_2r_2-k_2r_1}{s_2})(kst) = \frac{s_1s_2+s_1k_2r_2-s_1k_2r_1}{k_1s_2} \\
&= \frac{s_1}{k_1}+\frac{s_1k_2r_2}{k_1s_2}-\frac{s_1k_2r_1}{k_1s_2} \\
\frac{s_2 e_{1_2} e_{2_2}}{k_2} &= \frac{s_2}{k_2}(\frac{s_2+k_2r_2-k_2r_1}{s_2})(\frac{s_2+k_2r_2-k_2r_1}{s_2}) \\
&= \frac{s_2}{k_2}(1+\frac{k_2r_2}{s_2}-\frac{k_2r_1}{s_2})(1+\frac{k_2r_2}{s_2}-\frac{k_2r_1}{s_2}) \\
&= \frac{s_2}{k_2}[1+\frac{k_2r_2}{s_2}-\frac{k_2r_1}{s_2}+\frac{k_2r_2}{s_2}+\frac{k_2^2r_2^2}{s_2^2}-\frac{k_2^2r_1r_2}{s_2^2}-\frac{k_2r_1}{s_2}-\frac{k_2^2r_1r_2}{s_2^2}+\frac{k_2^2r_1^2}{s_2^2}] \\
&= [\frac{s_2}{k_2}+r_2-r_1+r_2+\frac{k_2r_2^2}{s_2}-\frac{k_2r_1r_2}{s_2}-r_1-\frac{k_2r_1r_2}{s_2}+\frac{k_2r_1^2}{s_2}] \\
&= [\frac{s_2}{k_2}+2r_2-2r_1+\frac{k_2r_2^2}{s_2}-2\frac{k_2r_1r_2}{s_2}+\frac{k_2r_1^2}{s_2}] \\
\frac{s_2 e_{1_2} e_{2_3}}{k_2kst} &= \frac{s_2}{k_2kst}(\frac{s_2+k_2r_2-k_2r_1}{s_2})(kst) = \frac{s_2}{k_2}+r_2-r_1
\end{aligned}$$

Finally:

$$\begin{aligned}
v &= \frac{1}{t}[\frac{s_1}{k_1}\times(\frac{k_1t_1v_1v_2}{s_1}+v_2+\frac{k_1r_1v_2}{s_1}-\frac{k_1r_2v_2}{s_1}-v_2) \\
&\quad +\frac{s_2}{k_2}\times(v_2+\frac{k_2r_2v_2}{s_2}-\frac{k_2r_1v_2}{s_2}-v_2)] \\
&= \frac{1}{t}[t_1v_1v_2+r_1v_2-r_2v_2+r_2v_2-r_1v_2] \\
&= \frac{1}{t}[t_1v_1v_2] \\
&= v_1v_2
\end{aligned}$$

A.4 Order Preserving Feature Correctness

In MLS, data ordering is preserved using the data encryption function associated with: (i) the key generation conditions (Sub-section 9.2.1) and (ii) the ω-concept that, although generating random values, retains the data ordering across the cyphertexts. The ω-concept uses a simple mathematical rule to ensure a "gap" between cyphertexts so that adding random offsets (sampled from a particular range) will not cause any overlap, and hence guarantees data ordering. The value for ω can be determined using 10^p, and the random values r_i can then be sampled from range 0 to ω. Random values are generated each time the encryption function is called, a side-effect of this is that data equality is not preserved. This feature facilitates precluding Cyphertext Only Attacks (COAs) by generating different cyphertexts for the same plaintext value, even when the same list of keys are used (the probabilistic feature of the MLS scheme). If we consider the situation where $q = 1$ and $m = 3$, the encryptions of v_1 and v_2 are $E_1 = \{e_{1_1}, e_{1_2}, e_{1_3}\}$ and $E_2 = \{e_{2_1}, e_{2_2}, e_{2_3}\}$ where:

$$\begin{aligned}
e_{1_1} &= \frac{k_1t_1v_1+s_1+k_1(r_{1_1}-r_{1_2})}{s_1} \\
e_{2_1} &= \frac{k_1t_1v_2+s_1+k_1(r_{2_1}-r_{2_2})}{s_1}
\end{aligned}$$

As already noted, the encryption function selects a different value for r_i every time the encryption function is invoked, thus $r_{1_i} \neq r_{2_i}$. If $v_1 > v_2$. Thus, applying the encryption function, and since the k_1, s_1 and t_1 values are all positive, the consistent values (private keys) will be:

$$k_1 t_1 v_1 + s_1 > k_1 t_1 v_2 + s_1$$

Adding random numbers to the above might change the data ordering. The ω-concept is used to create a "gap" between every consecutive plaintext value so as to allow the addition of a random number while preserving the data ordering. The value of ω is embedded in t_1, when $q = 1$, in otherwords $t_1 = (s_1 + k_1) \times \omega$. Recall the value of t_1 is multiplied with the v in MLS encryption function. The value of the random numbers r_{1_i} and r_{2_i} are sampled from 0 to ω. Therefore, the maximum value of $r_{1_1} - r_{1_2}$ is ω and also the maximum value of $r_{2_1} - r_{2_2}$ is ω (less than the gap multiplied with v in encryption function). Recall that r_{i_1} when $q = 1$ is greater than the value of other random values. Therefore, the encrypted values of v_1 and v_2 can be compared:

$$\begin{aligned} k_1 v_1 &> k_1 v_2 \\ k_1 v_1 \omega &\gg k_1 v_2 \omega \qquad\qquad where\ \omega = 10^x \\ k_1 v_1 \omega (s_1 + k_1) &\gg k_1 v_2 \omega (s_1 + k_1) \\ k_1 v_1 t_1 &\gg k_1 v_2 t_1 \\ k_1 v_1 t_1 + s_1 &\gg k_1 v_2 t_1 + s_1 \end{aligned} \tag{A.10}$$

In mathematics if c is an integer number greater than 1, then $c \times num >> c + num$; the ω-concept uses this basic mathematical rule so that if a random value, sampled from the range 0 to ω, is added to the two operand in Equation A.10 the data order will still hold.

$$\begin{aligned} k_1 v_1 t_1 + s_1 + k_1 (r_{1_1} - r_{1_2}) &> k_1 v_2 t_1 + s_1 + k_1 (r_{2_1} - r_{2_2}) \\ \frac{k_1 v_1 t_1 + s_1 + k_1 (r_{1_1} - r_{1_2})}{s_1} &> \frac{k_1 v_2 t_1 + s_1 + k_1 (r_{2_1} - r_{2_2})}{s_1} \\ e_{1_1} &> e_{2_1} \end{aligned} \tag{A.11}$$