

Application of Privacy-Preserving Clustering Methods Using Homomorphic Encryption Algorithms

A thesis submitted to the
Graduate School of Natural and Applied Sciences

by

İsmail AYDIN

in partial fulfillment for the
degree of Master of Science

in

Cybersecurity Engineering



This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Cybersecurity Engineering.

APPROVED BY:

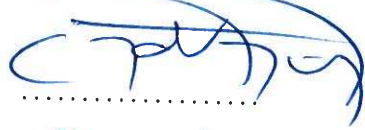
Prof. Dr. Ensar Gül
(Thesis Advisor)



Dr. Ferhat Özgür Çatak
(Thesis Co-advisor)



Assoc. Prof. Müjdat Soytürk



Asst. Prof. Ali Çakmak



This is to confirm that this thesis complies with all the standards set by the Graduate School of Natural and Applied Sciences of İstanbul Şehir University:

DATE OF APPROVAL:

SEAL/SIGNATURE:



Declaration of Authorship

I, İsmail AYDIN, declare that this thesis titled, 'Application of Privacy-Preserving Clustering Methods Using Homomorphic Encryption Algorithms' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 

Date: 19.09.2018

“Kindness is stronger than fear.”

Cicero



Application of Privacy-Preserving Clustering Methods Using Homomorphic Encryption Algorithms

İsmail AYDIN

Abstract

The need of protection and processing of the sensitive data in large scale data systems (for example data derived from financial systems, militaristic systems or social media platforms) is a common problem. Usage of traditional cryptographic methods for data protection mainly needs at least two of the ciphering, deciphering and data processing works to be done on the same side. Because of this, with increase of the data size there will be a need for higher processing power to work on the data.

Using traditional encryption algorithms for protection of the sensitive data on large scale systems, also brings the need of exchanging the needed keys for protection and processing the data. Homomorphic encryption schemes have enough flexibility that, they should be used on data systems that contains data from multiple parts, because of its feature of allowing to process the encrypted data like its non-encrypted form.

With the usage of homomorphic encryption schemes and proper data learning systems on encrypted data, distribution of sensitive data to different parties can be done without violating its privacy. In this thesis, we propose a method to run mathematical computations which needs high processing power on a common platform which offers high processing power of data but not on parties that the sensitive data will be distributed. As a result the partners of this systems will not need to have high processing power to function on the data because the high processing demanding tasks would be done on the common platform.

In this research Paillier Cryptographic system was used to protect data privacy. Paillier Cryptographic algorithm's most prominent features are its asymmetrical and partially homomorphic behavior. We proposed a system that uses privacy preserving distance matrix calculation as input for several clustering algorithms which are commonly used in machine learning systems. Our system is evaluated considering different data lengths and different key lengths. Four different data clustering methods have been tested. By applying clustering algorithms on both encrypted and plain forms of the same data for different key and data lengths, we obtained performance results by using six different metrics.

Keywords: Machine Learning, Cryptography, Homomorphic, Clustering

Homomorfik Şifreleme Algoritmaları Kullanılarak Mahremiyet Korumalı Gruplandırma Yöntemlerinin Uygulanması

İsmail AYDIN

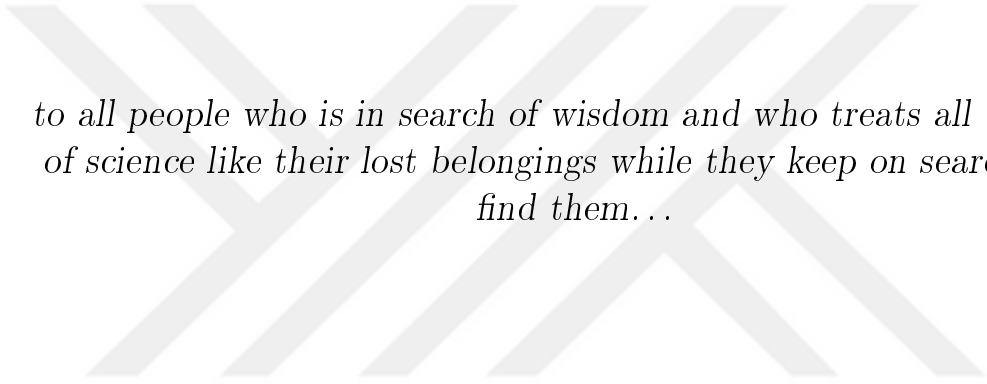
ÖZ

Günümüzde finans, sağlık, askeri sistemler veya sosyal platformlarda elde edilmiş ve mahremiyeti korunması gereken büyük veri topluluklarının işlenmesi/anlamlandırılması ihtiyacı mevcuttur. Mahremiyet koruma amacıyla klasik şifreleme yöntemlerinin kullanımı, verinin kullanılacağı sistemde şifreleme, şifre çözme veya verinin anlamlandırılması işlemlerinin en az ikisinin aynı yerde yapılmasını gerektirir. Veri büyüklüğünün artması ile beraber bu işlemlerin aynı yerde yapılması durumunda büyük miktarlarda bir işlem gücü ihtiyacı doğacaktır.

Klasik anlamda kriptolama yöntemlerinin çok sayıda bağlantı içeren büyük veri sistemlerinde kullanımı durumunda, işlem yüküne ek olarak çok sayıda kullanıcının her birinde uygun anahtar dağıtım mekanizmalarının da çalışması gerekecektir. Çok sayıda kullanıcının bir araya gelmiş olduğu bir büyük veri sisteminde gerek anahtar dağıtım mekanizmalarının koşmasının, gerekse de büyük veri üzerinde yapılacak yüksek işlem gücü gerektiren işlemlerin ortak bir platform üzerinde yapılmasına imkan vermesi sebebiyle bu çalışmada homomorfik şifreleme yöntemlerinin kullanımı önerilmektedir. Homomorfik şifreleme yöntemleri ile beraber şifreli veri üzerinde uygun makine öğrenme yöntemleri kullanılması sayesinde büyük verilerin paydaşlara dağıtımının ve veri işlemenin mahremiyete aykırı bir durum oluşturmadan yapılabilmesi mümkün hale gelmektedir.

Bu sayede sistem paydaşlarının yüksek işlem kapasitesine sahip olmasına gerek kalmadan büyük veri işleme mekanizmalarına dahil olup, işlem yapabilme imkanına sahip olması sağlanacaktır. Tasarlanan sistemin çalışmasına uygun olması sebebiyle asimetric bir şifreleme algoritması olan ve homomorfik özellik göstermesi sebebiyle mahremiyet koruma amacıyla Paillier kriptolama sistemi kullanılmıştır. Makine öğrenme yöntemlerinin uygulanması amacıyla tasarlanan sistem üzerinde farklı veri uzunlukları, farklı anahtar uzunlukları kullanılarak mahremiyeti sağlanan sistemde 4 ayrı makine öğrenme yöntemi koşturulmuştur. Her algoritmanın farklı anahtar ve veri uzunluğu için göstermiş olduğu performans, aynı verinin açık ve kapalı halleri üzerinde koşturulan makine öğrenme algoritmalarının 6 farklı ölçüt üzerinden değerlendirmeye tutulması ile tespit edilmiştir.

Anahtar Sözcükler: Makine Öğrenimi, Kriptografi, Homomorfik, Kümeleme



*to all people who is in search of wisdom and who treats all branches
of science like their lost belongings while they keep on searching to
find them...*

Acknowledgments

I would like to express my sincere gratitude to my Co. advisor Dr. Ferhat Özgür Çatak for the continuous support of my study and related research, for his patience, motivation, and immense knowledge. Whenever I ran into a trouble spot or had a question about my research or writing, he consistently allowed this paper to be my own work, but steered me in the right direction whenever he thought I needed it.

I would also like to thank to Prof. Gül. I am gratefully indebted to him for his very valuable comments and guidance on this thesis.

Finally, I must express my very profound gratitude to my parents and to my spouse for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

Thank you...

Contents

Declaration of Authorship	ii
Abstract	iv
Öz	v
Acknowledgments	vii
List of Figures	x
List of Tables	xiii
1 Introduction	1
1.1 Current Situation	1
1.2 Contribution	3
2 Related Work	4
2.1 Related Work	4
3 Preliminaries	8
3.1 Data Clustering	8
3.1.1 K-Means Clustering	10
3.1.2 Hierarchical Clustering	11
3.1.3 Spectral Clustering	12
3.1.4 Birch Clustering	12
3.1.5 Evaluation Metrics	13
3.1.5.1 Homogeneity	13
3.1.5.2 Completeness	14
3.1.5.3 V-Measure	14
3.1.5.4 Adjusted Rand Index	15
3.1.5.5 Adjusted Mutual Information	15
3.1.5.6 Silhouette Coefficient	16
3.2 Homomorphic Encryption	16
3.2.1 Paillier Cryptosystem	17
3.2.2 Floating Point Numbers	17
4 System Model	18
4.1 Development Environment	18
4.2 Sequence Diagram	18

4.2.1	Client Computaion	18
4.2.2	Data Authority Computation	20
4.2.3	Model Building at Client	21
5	Experiments and Results	24
5.1	Plaintext Results	25
5.1.1	K-Means Algorithm Results	25
5.1.2	Hierarchical Algorithm Results	28
5.1.3	Spectral Algorithm Results	31
5.1.4	Birch Algorithm Results	35
5.2	Encrypted Domain Results	38
5.2.1	K-Means Algorithm Results	38
5.2.2	Hierarchical Algorithm Results	45
5.2.3	Spectral Algorithm Results	52
5.2.4	Birch Algorithm Results	59
5.3	Results	66
6	Conclusions and Future Work	67
	Bibliography	69

List of Figures

4.1	Sequence Diagram for Client Side	19
4.2	Sequence Diagram for Data Authority Side	21
4.3	Sequence Diagram for Model Building at Client Side	22
5.1	Plain domain clustering for dataset 500	25
5.2	Plain domain clustering for dataset 1000	25
5.3	Plain domain clustering for dataset 1500	26
5.4	Plain domain clustering for dataset 2000	26
5.5	Plain domain clustering for dataset 2500	26
5.6	Plain domain clustering for dataset 3000	27
5.7	Plain domain clustering for dataset 3500	27
5.8	Plain domain clustering for dataset 4000	27
5.9	Plain domain clustering for dataset 4500	28
5.10	Plain domain clustering for dataset 5000	28
5.11	Plain domain clustering for dataset 500	28
5.12	Plain domain clustering for dataset 1000	29
5.13	Plain domain clustering for dataset 1500	29
5.14	Plain domain clustering for dataset 2000	29
5.15	Plain domain clustering for dataset 2500	30
5.16	Plain domain clustering for dataset 3000	30
5.17	Plain domain clustering for dataset 3500	30
5.18	Plain domain clustering for dataset 4000	31
5.19	Plain domain clustering for dataset 4500	31
5.20	Plain domain clustering for dataset 5000	31
5.21	Plain domain clustering for dataset 500	32
5.22	Plain domain clustering for dataset 1000	32
5.23	Plain domain clustering for dataset 1500	32
5.24	Plain domain clustering for dataset 2000	33
5.25	Plain domain clustering for dataset 2500	33
5.26	Plain domain clustering for dataset 3000	33
5.27	Plain domain clustering for dataset 3500	34
5.28	Plain domain clustering for dataset 4000	34
5.29	Plain domain clustering for dataset 4500	34
5.30	Plain domain clustering for dataset 5000	35
5.31	Plain domain clustering for dataset 500	35
5.32	Plain domain clustering for dataset 1000	35
5.33	Plain domain clustering for dataset 1500	36
5.34	Plain domain clustering for dataset 2000	36

5.35	Plain domain clustering for dataset 2500	36
5.36	Plain domain clustering for dataset 3000	37
5.37	Plain domain clustering for dataset 3500	37
5.38	Plain domain clustering for dataset 4000	37
5.39	Plain domain clustering for dataset 4500	38
5.40	Plain domain clustering for dataset 5000	38
5.41	Encrypted domain clustering results for dataset 500	39
5.42	Encrypted domain clustering results for dataset 1000	39
5.43	Encrypted domain clustering results for dataset 1500	40
5.44	Encrypted domain clustering results for dataset 2000	40
5.45	Encrypted domain clustering results for dataset 2500	41
5.46	Encrypted domain clustering results for dataset 3000	41
5.47	Encrypted domain clustering results for dataset 3500	42
5.48	Encrypted domain clustering results for dataset 4000	42
5.49	Encrypted domain clustering results for dataset 4500	43
5.50	Encrypted domain clustering results for dataset 5000	43
5.51	Charts that show change of each evaluation metrics score by data length for different key lengths and for K-Means Algorithm	44
5.52	Calculation Time Graph for KMeans Algorithm on Key-Data Length Dimensions	44
5.53	Encrypted domain clustering results for dataset 500	45
5.54	Encrypted domain clustering results for dataset 1000	46
5.55	Encrypted domain clustering results for dataset 1500	46
5.56	Encrypted domain clustering results for dataset 2000	47
5.57	Encrypted domain clustering results for dataset 2500	47
5.58	Encrypted domain clustering results for dataset 3000	48
5.59	Encrypted domain clustering results for dataset 3500	48
5.60	Encrypted domain clustering results for dataset 4000	49
5.61	Encrypted domain clustering results for dataset 4500	49
5.62	Encrypted domain clustering results for dataset 5000	50
5.63	Charts that show change of each evaluation metrics score by data length for different key lengths and for Hierarchical Algorithm	51
5.64	Calculation Time Graph for Hierarchical Algorithm on Key-Data Length Dimensions	51
5.65	Enrypted domain results for dataset 500	52
5.66	Enrypted domain results for dataset 1000	53
5.67	Enrypted domain results for dataset 1500	53
5.68	Enrypted domain results for dataset 2000	54
5.69	Enrypted domain results for dataset 2500	54
5.70	Enrypted domain results for dataset 3000	55
5.71	Enrypted domain results for dataset 3500	55
5.72	Enrypted domain results for dataset 4000	56
5.73	Enrypted domain results for dataset 4500	56
5.74	Enrypted domain results for dataset 5000	57
5.75	Charts that show change of each evaluation metrics score by data length for different key lengths and for Spectral Algorithm	58

5.76 Calculation Time Graph for Spectral Algorithm on Key-Data Length Dimensions	58
5.77 Encrypted domain clustering results for dataset 500	59
5.78 Encrypted domain clustering results for dataset 1000	60
5.79 Encrypted domain clustering results for dataset 1500	60
5.80 Encrypted domain clustering results for dataset 2000	61
5.81 Encrypted domain clustering results for dataset 2500	61
5.82 Encrypted domain clustering results for dataset 3000	62
5.83 Encrypted domain clustering results for dataset 3500	62
5.84 Encrypted domain clustering results for dataset 4000	63
5.85 Encrypted domain clustering results for dataset 4500	63
5.86 Encrypted domain clustering results for dataset 5000	64
5.87 Charts that show change of each evaluation metrics score by data length for different key lengths and for Birch Algorithm	65
5.88 Calculation Time Graph of Birch Algorithm on Key-Data Length Dimensions	65



List of Tables

5.1	Plain domain evaluation metric scores for dataset 500	25
5.2	Plain domain evaluation metric scores for dataset 1000	25
5.3	Plain domain evaluation metric scores for dataset 1500	26
5.4	Plain domain evaluation metric scores for dataset 2000	26
5.5	Plain domain evaluation metric scores for dataset 2500	26
5.6	Plain domain evaluation metric scores for dataset 3000	27
5.7	Plain domain evaluation metric scores for dataset 3500	27
5.8	Plain domain evaluation metric scores for dataset 4000	27
5.9	Plain domain evaluation metric scores for dataset 4500	28
5.10	Plain domain evaluation metric scores for dataset 5000	28
5.11	Plain domain evaluation metric scores for dataset 500	28
5.12	Plain domain evaluation metric scores for dataset 1000	29
5.13	Plain domain evaluation metric scores for dataset 1500	29
5.14	Plain domain evaluation metric scores for dataset 2000	29
5.15	Plain domain evaluation metric scores for dataset 2500	30
5.16	Plain domain evaluation metric scores for dataset 3000	30
5.17	Plain domain evaluation metric scores for dataset 3500	30
5.18	Plain domain evaluation metric scores for dataset 4000	31
5.19	Plain domain evaluation metric scores for dataset 4500	31
5.20	Plain domain evaluation metric scores for dataset 5000	31
5.21	Plain domain evaluation metric scores for dataset 500	32
5.22	Plain domain evaluation metric scores for dataset 1000	32
5.23	Plain domain evaluation metric scores for dataset 1500	32
5.24	Plain domain evaluation metric scores for dataset 2000	33
5.25	Plain domain evaluation metric scores for dataset 2500	33
5.26	Plain domain evaluation metric scores for dataset 3000	33
5.27	Plain domain evaluation metric scores for dataset 3500	34
5.28	Plain domain evaluation metric scores for dataset 4000	34
5.29	Plain domain evaluation metric scores for dataset 4500	34
5.30	Plain domain evaluation metric scores for dataset 5000	35
5.31	Plain domain evaluation metric scores for dataset 500	35
5.32	Plain domain evaluation metric scores for dataset 1000	35
5.33	Plain domain evaluation metric scores for dataset 1500	36
5.34	Plain domain evaluation metric scores for dataset 2000	36
5.35	Plain domain evaluation metric scores for dataset 2500	36
5.36	Plain domain evaluation metric scores for dataset 3000	37
5.37	Plain domain evaluation metric scores for dataset 3500	37

5.38	Plain domain evaluation metric scores for dataset 4000	37
5.39	Plain domain evaluation metric scores for dataset 4500	38
5.40	Plain domain evaluation metric scores for dataset 5000	38
5.41	Encrypted domain evaluation metric scores for dataset 500	39
5.42	Encrypted domain evaluation metric scores for dataset 1000	39
5.43	Encrypted domain evaluation metric scores for dataset 1500	40
5.44	Encrypted domain evaluation metric scores for dataset 2000	40
5.45	Encrypted domain evaluation metric scores for dataset 2500	41
5.46	Encrypted domain evaluation metric scores for dataset 3000	41
5.47	Encrypted domain evaluation metric scores for dataset 3500	42
5.48	Encrypted domain evaluation metric scores for dataset 4000	42
5.49	Encrypted domain evaluation metric scores for dataset 4500	43
5.50	Encrypted domain evaluation metric scores for dataset 5000	43
5.51	Encrypted domain evaluation metric scores for dataset 500	45
5.52	Encrypted domain evaluation metric scores for dataset 1000	45
5.53	Encrypted domain evaluation metric scores for dataset 1500	46
5.54	Encrypted domain evaluation metric scores for dataset 2000	47
5.55	Encrypted domain evaluation metric scores for dataset 2500	47
5.56	Encrypted domain evaluation metric scores for dataset 3000	48
5.57	Encrypted domain evaluation metric scores for dataset 3500	48
5.58	Encrypted domain evaluation metric scores for dataset 4000	49
5.59	Encrypted domain evaluation metric scores for dataset 4500	49
5.60	Encrypted domain evaluation metric scores for dataset 5000	50
5.61	Encrypted domain clustering results with dataset 500	52
5.62	Encrypted domain clustering results with dataset 1000	52
5.63	Encrypted domain clustering results with dataset 1500	53
5.64	Encrypted domain clustering results with dataset 2000	54
5.65	Encrypted domain clustering results with dataset 2500	54
5.66	Encrypted domain clustering results with dataset 3000	55
5.67	Encrypted domain clustering results with dataset 3500	55
5.68	Encrypted domain clustering results with dataset 4000	56
5.69	Encrypted domain clustering results with dataset 4500	56
5.70	Encrypted domain clustering results with dataset 5000	57
5.71	Encrypted domain evaluation metric scores for dataset 500	59
5.72	Encrypted domain evaluation metric scores for dataset 1000	59
5.73	Encrypted domain evaluation metric scores for dataset 1500	60
5.74	Encrypted domain evaluation metric scores for dataset 2000	61
5.75	Encrypted domain evaluation metric scores for dataset 2500	61
5.76	Encrypted domain evaluation metric scores for dataset 3000	62
5.77	Encrypted domain evaluation metric scores for dataset 3500	62
5.78	Encrypted domain evaluation metric scores for dataset 4000	63
5.79	Encrypted domain evaluation metric scores for dataset 4500	63
5.80	Encrypted domain evaluation metric scores for dataset 5000	64

Chapter 1

Introduction

There is a need for big data[1] systems that allows users to quickly handle sensitive data which may be gathered from systems like healthcare systems or financial systems and without violating its privacy.

Machine learning[2] is gaining a high reputation for handling valuable and private data in an efficient way on big data systems. Sometimes big data systems contain data batches related to systems that has different privacy policies from each other but had to be handled in a mutual way. Because of the different privacy policies that different parties have, sensitive data can't be distributed everytime.

1.1 Current Situation

It is a hard question that "How can sensitive data be distributed between multiple parties without making concessions?". Classically to find an answer to this question symmetric and asymmetric (public-private key cryptography) ciphering algorithms[3] are being used. The strength of classical cryptographic tools relies on secrecy of crypto key, strength of the algorithm and randomness that used in algorithm.

When classical ways are appealed, ciphering - deciphering works and processing the open data takes part in the same place. When it is desired to provide privacy using classical cryptographic algorithms, the parties need to have proper key or keys from public-private key pairs and the keys had to be transferred using a safe channel. As it is easy to see using this type of traditional systems, brings the need to compute all processes that need high processing power in the same place. Regrettably, this type of systems are being

incapable for big data systems because classical cryptographic algorithms are mainly designed for small datasets.

Data handling for machine learning algorithms while considering privacy[4] issues, primarily has two main approaches;

Firstly; using the distinctive features of big datasets for the suppression and generalization to sanitize the big data. After that, the sanitized version of data can be distributed[5, 6] or published to data parties to run any machine learning algorithm.

Secondly, using cryptographically secure multi-party computation algorithms[7–9] to construct protocols that can compute the same answer when obtained in private and non-private cases. This approach is applied generally when the relationship between data parties is symmetrical. Symmetrical relationship means that if the database is partitioned and distributed to parties and result of the machine learning algorithms applied to the dataset are same. So the result of the algorithm execution shows that both parties learn the same output based on the joint database.

The difference between these two approaches is in the first approach (sanitization approach), the parties don't execute machine learning algorithms on the data which belong to themselves and the database owner doesn't get an output of the execution. Depending on the content and the quiddity of the data there might be a need to develop a classification model that allows to work just on a specific part of the data. To develop a classification model, compute-intensive processes would be used for sanitization without violating privacy of the data. Big data classifiers need an efficient for distributed learning and privacy preserving protocol. The method we will suggest aims to allow a user to create needed classifiers without reaching any extra information about the data. Therefore database owner also wouldn't know anything about the data classifier. Creating a data classifier by means of this method would be examined through a prototype application and the performance will be observed. In this thesis, a framework will be proposed and used for applying machine learning algorithms to datasets when it is distributed and shared between parties. We will also use Paillier Cryptographic system for handling big data.

1.2 Contribution

In today's world the data which needs to be preserved privately can reach up to large scales and may differ in a wide range of varieties. There are several methods in literature to provide privacy for these data. Applying classical cryptographic algorithms can't be enough every time for handling privacy issues of large scale data. It is foreseen that when a process needs to run on a sensitive dataset, it is not suitable every time to send the encrypted data to different parties. As classical cryptographic algorithms are used for encryption, the needed processes can be executed on data only after decryption of it. This way, although data privacy is highly violated for the side where the data would be executed in, there are some mechanisms to solve this problem. Considering their competence level, it is clear that this procedure can't be used in every condition. In respect to this data privacy violation problem, there is always a necessity for a system which can both allow to preserve privacy for data and doesn't violate privacy as exposing the real data to irrelevant parties when needed process execution is performed. In this thesis, a system that uses the Paillier Cryptography for classifying big data systems and allows to handle/process the data without violating the privacy has been proposed/designed. This proposed work can be implemented to any system that gathers critical, sensitive or private information to run several processes on it. As mentioned before, health care systems, militaristic systems, financial-commercial systems or instant private image/video processing systems. By changing used algorithms to process the data or running different algorithms rather than clustering algorithms, this model can be modified to make the system suitable to handle different needs. The proposed system would allow to use the data properly while preserving the privacy of data.

The main contributions of this research are as follows:

- Overcoming the need to use actual sensitive data for data handling is achieved by building a model that allows to use the distance matrix of the same data instead.
- The Paillier cryptosystem encryption-based clustering model building is proposed for preserving privacy and thus private clustering model training is achieved.
- With usage of the distance matrix of sensitive data, clustering performance of four different clustering algorithms have been evaluated in respect of 6 different evaluation metrics and computational time.
- A system model has been offered, which allows handling high processing power demanding tasks to be done on a powerful platform. So that the overall computational time aimed to be reduced thus it can be handled more effectively.

Chapter 2

Related Work

2.1 Related Work

In this section we will examine works related to using machine learning algorithms which depends on privacy preserving on big data systems.

Lindell Y. and Pinkas B., suggests a system which uses ID3 algorithm for data processing safety. They have stated that their system needs relatively less communication rounds and bandwidth. In this system, ID3 algorithm is used with decision tree learning and while privacy of data is preserved different users work on the data and then the results are merged by using cryptographic protocols [10]. The main difference between our proposed work and this work is the used cryptographic algorithm, ID3. Lindell Y. and Pinkas B. also focused on the problem of secure multi-party computation on a joint database but, their solution for privacy is using ID3 algorithm while the projected computation on the database is decision tree learning.

Chaudhuri K. and Monteleon C., consider the balance between secrecy and learnability while designing a privacy preserving algorithm for a database. They focus on privacy preserving logistic regression algorithm. Bounding the sensitivity due to distortion is measured when a noise is applied on the system while the regularized logical regression algorithm is using a classifier. A privacy-preserving regularized logistic regression algorithm is provided which is based on solving a perturbed optimization problem [11]. In their work they tried to construct a new learning method based on logistic regression to create privacy preserving linear classifiers, which differs from our work that we didn't preserve privacy by our classifiers, but with an homomorphic encryption algorithm.

Agrawal R. and Srikant R. state that in the future, data processing technics are going to aim on merging different security requirements on different platforms. They evaluated mathematical value of the distributed data to its original form and tried to accurately estimate the true values of the original data from distributed ones [12]. In this work the privacy is tried to be preserved by perturbing the original data by some randomization techniques, different from our proposed model. Also decision tree algorithms are used for classification of both original and reconstructed mutual data and these algorithms are ByClass and Local. In our work the mutual data is not perturbed for privacy and with this we also didn't need to reconstruct the original data.

Xu K. , Yue H. and their friends consider the traditional methods of cryptography for parties that doesn't share open data in respect of adequation. Due to this problem, they tried to minimize the data that needs to be processed with using the data locality feature of Apache Hadoop architecture's Map Reduce for protecting the data privacy[13], which is a big difference from our work that we offer usage of a cryptographic protocol. While the main focus of this work is similar to our study, the approach to find a solution for privacy preserving while handling big data is the main difference from our work because we didn't consider a commercial platform's instruments to find a solution but we tried to offer a general system without using any commercial platform. Also this work involves Hadoop's another feature for getting local training results, Mapper. The data locality is also a big difference from our proposed system because this work obliges the participants of the system to handle their data locally, not on a common powerful platform. Also because this system mainly runs on the Hadoop platform, participants must use their data in HDFS (Hadoop Distributed File System) format.

Merugu S. and Ghosh J. examined the costs of security and communication of distributed data in supervised and non-supervised scenarios. The suggestion they made is to transmit the parameters of suitable generative models which built at local data sites to a central database, instead of sharing the original data. The work showed that generating artificial samples from the original data distributions with using Markov Chain Monte Carlo techniques, it is mathematically possible to represent all the data with a mean "model" [14]. In this work, privacy is preserved by the hardness of reconstruction of the original data from the distributed model which is derived locally, no encryption algorithm is used. This work also includes distributed model clustering between different parties with different security concerns, so each party can choose a suitable expectation-maximization algorithm to use for clustering and the central "model merger" tries to find the best solution for merging these clustering results. On the contrary of this work, in our work we offer a single data authority and the calculations are made on encrypted data because

of its homomorphic characteristic and clustering is also not done in locals.

Shokri R. and Shmatikov V. tried to use the ability of gathering information and model building of artificial neural networks from complex datasets. They tried to design a practical system that allows different parties to jointly learn an accurate neural network model without sharing their input data. The researchers think that their system has a strong privacy compared to any existing approach due to minimal data sharing which is actually a small fraction of network parameters [15]. The usage of artificial intelligence and actuating neural network algorithms are done on the client side and this is one of the differences from our proposed model. Another difference is, this study offers a model that uses artificial intelligence which aims to work independent from the specific algorithm which is used on training data. Also different from proposed system, participants of this system share their models with each other, so participants may also learn from other participants' models.

Yang et al. offer a system that allows users to use a model for calculating data frequencies which also preserves privacy. The main logic of the system is to calculate the frequencies of specific values or a group of values of client side's data at the data mining side and while doing it data privacy is still protected by ElGamal cryptographic algorithm. It has been stated that there is no information shared except frequency of data values [16]. This work is focusing mainly on the scenario that participants of the system doesn't want to use the result of the data mining procedure which is the most prominent difference from our proposed system that in our system client "wants to use" the clustering result. Another difference is that this work also aims on calculating the frequencies of specific values of the data of client side but, our model clusters data using certain clustering algorithms.

Sahin O.D. , Agrawal A. and El Abbadi A. offer a system that guarantees the data of a party which doesn't pertain to another data source won't be revealed. To provide the privacy and build a distributed decision tree learning algorithm, ID3 algorithm and Shamir's secret sharing are used [17]. Their work differs from our model by the algorithm used for privacy which is Shamir's secret sharing and the algorithm runs on the data aiming to create decision trees which is ID3 algorithm. Different from our work, this work is proposing a non-homomorphic algorithm for data privacy and constrains the client to use it three times successively on different calculation phases, but we propose a system model that uses homomorphic algorithm and the clustering algorithms run on the encrypted data so encryption and decryption costs are significantly lower.

Li et al. suggest usage of multi key-fully homomorphic encryption as well as a hybrid structure which combines double decryption with fully homomorphic encryption. They tried to prove that these two privacy-preserving algorithms are proper to use with deep learning algorithms over encrypted data. Different users choose their keys and encrypt their data. Encrypted data is sent on a cloud and the execution on the data is made by these two suggested systems [18]. Different from our model, this work mainly focuses on the issue of collaborative deep learning. Also we use a classical encryption-decryption routine in our model but in this work, to preserve privacy double decryption is offered not only to protect the data, but also to protect the model that every participant of the system created from their data.

Yi X. and Zhang Y. considered a privacy preserving Bayes classifier method for horizontally partitioned data and proposed two protocols. One of these protocols are two-party protocol and the other one is a multi-party protocol. Multi-party protocol is used between owners of sensitive data and a semi trusted server while two party protocol just broadcasts the classification result. In this work it is assumed that these two protocols are trusted and can preserve privacy [19]. This study differs from our proposed model by the used classification protocol which is naive Bayes classification. While we propose a model that preserves privacy by using a homomorphic encryption algorithm, Paillier cryptographic system, this study aims for the same objective by enhancing the Bayes classification model offered by Kantarcioglu and Vaidya.

Secretan J. , Georgiopoulos M. , Koufakou A. and Cardona K. approached the hardness issue of developing a privacy preserving data mining (PPDM) algorithm. PPDM algorithms are computationally intensive to execute and there is a need in the data mining that developers need convenient abstraction algorithms for simplification of the system. Different from our work, this study focuses on using parallel computing between different organizations and it is advised in their study because of its ability to bring high performance and that can bear on the computationally intensive works of data mining. Their study mainly considers a system built on the idea of in one tier a simplified use of cluster and grid resources would exist and at another tier the system would just abstract the communication for algorithm development [20]. This study mainly differs from our work by two reasons. Firstly, its main focus is trying to integrate a high performance and parallel computing environment between different organizations and secondly it suggests usage of APHID (Architecture for Private and High-performance Integrated Data mining) because of the lack of middleware frameworks that organizations would need to support PPDM.

Chapter 3

Preliminaries

In this chapter, we will briefly examine on data clustering, clustering methods, homomorphic encryption and Paillier Cryptosystem.

3.1 Data Clustering

Clustering can be described as dividing accumulated elements[21] into different groups depending on special features they have. Elements that are similar with each other should be in the same group as much as possible. Same logic is a subject on clustering algorithms aiming to work together with data mining algorithms.

Clustering analysis has been originally used in anthropology by Driver and Kroeber[22, 23] and then introduced to psychology by Zubin. So, clustering may be done using different methods due to different needs and different logical reasons with compliance to several flexibilities. Some of these methods can be described as,

Centroid-Based Clustering: Centralized clustering or centroid-based clustering[24, 25] represents a group by a central vector which doesn't have to be a member of the dataset it belongs. This method comes up with a problem: How many clusters there should be? This question is the main drawback of data mining algorithms that depends on this method (such as K-Means algorithm) and common approach to this problem is to find approximate solutions. After deciding cluster numbers, then central vectors for each cluster is calculated by squared distances from the clusters to find nearest elements to form clusters. Due to this logic the squared distances from center should have their

minimal value.

Distribution-Based Clustering: In this method elements of a big dataset is clustered due to the statistical model they create[26]. Clusters can be defined as elements belonging most likely to the same distribution. This method can be considered as an excellent method theoretically but it suffers a main problem known as over fitting. One prominent mixture model that is in use with this method is *Gaussian mixture*[27]. Dataset is initially modeled with a *Gaussian distribution* randomly then the parameters are optimized to fit the dataset better. This will converge into an optimum model, so iteration is needed to find the best model. Distribution-based clustering models are good for capturing correlation and dependencies between samples although it brings an extra burden on the user in terms of iterative computing.

Density-based clustering: When clusters are defined considering the areas of high density of elements in a big dataset, that method is density based clustering[28, 29]. Methods that use density-based clustering use different criterion for defining the density. One of the most popular criterion is called as "*density reachability*" [30]. This logic works in the way of searching for elements in a dataset which are within a certain threshold value and adding those elements into a same cluster. There are different algorithms that can forms clusters according to same-density data and because of the working logic these algorithms have they can form arbitrarily shaped clusters on the contrary of many other clustering algorithms.

Connectivity-based clustering: This clustering method relies on the idea of clustering logic which collects elements of a big dataset that are more related to nearby elements than elements farther away and form a cluster[31–33]. This method forms clusters according to their distance, so a cluster can be described by the maximum distance needed to collect elements. Different clusters will form at different distances, so this model can be represented with a "*dendrogram*" [34] because these algorithms provide a hierarchical model that within a certain model clusters also merge with each other. Distance values that are in use for clustering can be calculated or selected due to different needs. For example distance that will be used for clustering can be minimum or maximum distances between elements or average distances between them.

Different clustering algorithms can be used according to the chosen clustering logic. In our work, we analyzed four different clustering algorithms in respect to different logical approaches which are described above and working procedure of these algorithms will be

explained below together with the method they use while clustering a dataset.

3.1.1 K-Means Clustering

K-Means clustering method[35] creates clusters from a dataset according to previously determined cluster number (n clusters) and while doing that, clustering is done in according to have nearest "*inertia*" values for every cluster. Inertia value is calculated as sum of squared distances between the central point of a cluster and every other point inside the same cluster. This algorithm is generally useful for big datasets and is used in many different applications[36]. K-Means algorithm divides a set of N samples into K disjoint clusters while cluster centroids are and the other points inside a cluster are X . K-Means algorithm aims to choose centroids that minimize the inertia in accordance with the equation of:

$$\sum_{i=0}^n \min(\|x - \mu_i\|^2) \quad (3.1)$$

Inertia, or the within-cluster sum of squares, can be used to measure how internally coherent clusters are. This criterion also suffers from several conditions. For example it is usually assumed that the dataset is "*complex*" and "*isotropic*" but unfortunately it isn't always the case, so it makes inertia inefficient to elongated clusters or irregular shaped datasets. Furthermore, while smaller values are better and the best case is when the value is 0, on high-dimensional datasets Euclidean distances[37] tend to become inflated. This situation is named as "*curse of dimensionality*". To speed up K-Means algorithm and alleviate this problem, dimensionality reduction algorithms can be used before running K-Means algorithm on a dataset.

To get the best clustering results, K-Means algorithm should run on the same dataset multiple times[38]. Because of the need for high processing power and speed, we suggest that computations such as these algorithms should run on a powerful cloud environment to eliminate the case that users shall provide that much of processing power to the system. After enough time, K-Means algorithm will always converge to a local minimum value.

K-Means algorithm will initially create clusters by grouping elements around chosen central points and according to inertia values of these clusters. As the iteration goes on, algorithm shifts central points and re-group elements into clusters and calculate new inertia to converge into a minimum value. As it easy to see, it is important to choose accurate central points at the initialization of computing because more accurate cluster centroids will significantly reduce time or number of iteration to get better clustering.

3.1.2 Hierarchical Clustering

Hierarchical clustering is a general name for certain clustering methods that builds nested clusters by merging or splitting their elements. These clustering methods create clusters with a logic similar to root-tree structure. The hierarchy of clusters can be represented as a tree shape (*dendrogram*) while the roots of the tree represent clusters and leaves represent only one sample that collect some clusters under itself.

Hierarchical algorithms can be expressed under two main titles, Agglomerative clustering and Divisive clustering algorithms. In our work we used Agglomerative Clustering method and this method works with a bottom-up approach like root to tree logic, unlike Divisive clustering method. The way that Agglomerative clustering method will follow for clustering depends on the clustering number which is predetermined and the method the algorithm will use[39]. These methods are:

- Ward: For every group, calculation of inertia is an issue like in the K-Means Clustering but usage of this value is different from K-Means algorithm due to structure of Agglomerative clustering. It aims to minimize the inertia differences within all clusters.

$$d_{ij} = d(\{X_i\}, \{X_j\}) = \|X_i - X_j\|^2 \quad (3.2)$$

- Maximum or Complete Linkage: Clustering is done with consideration of minimizing the maximum distances between different clusters. While the distance between clusters is d , the logic of this method can be described as:

$$\max\{d(x, y) : x \in A, y \in B\} \quad (3.3)$$

- Average Linkage: Minimizing the mean distance between pairs of clusters are used for clustering. For example, while x and y represent points belonging to different clusters, the equation to calculate the mean distance between cluster A and cluster B is as:

$$\frac{1}{|A| \cdot |B|} \sum_{x \in A} \sum_{y \in B} d(x, y) \quad (3.4)$$

3.1.3 Spectral Clustering

Spectral clustering mainly work on to embed the affinity matrix between samples[40], followed by a clustering algorithm. This method is especially efficient on relatively small datasets or if the affinity matrix is sparse and the dataset is convex[41]. This algorithm needs cluster number to be specified before working on the dataset.

3.1.4 Birch Clustering

Birch algorithm[42] builds a tree called "*the characteristic feature tree*" (*CFT*). *CFT* structure consist of "*characteristic feature nodes*" (*CFN*) and these *CFNs* are made of "*characteristic feature sub-clusters*" (*CFS*). With the information gathered from characteristic feature sub-clusters, which is the subsidiary of *CFT* tree, there is no need to save the entire data on the memory to create clusters from the dataset. Birch algorithm also brings effectiveness to memory use on the platform it runs and it is done by holding some information about the dataset. Some of these informations are:

- Number of samples in a sub-cluster.
- Linear Sum: A *n-dimensional* vector holding the sum of all samples
- Centroids: This avoids recalculation of linear sum for *n* samples
- Squared Sum: Squared norm of the centroids.

Birch algorithm primarily needs two information to cluster the dataset. These are threshold value which will be the radiant of clusters which puts a limit for clusters and branching factor which defines maximum number of elements that every cluster can have. Birch algorithm can only work on dataset after gathering these information. After the clustering is done, if a new sample is inserted into the dataset, it is then merged with most proper sub-cluster constrained by the threshold and branching factor conditions. If the radius of the sub-cluster obtained after the merging the new sample and the branching factor is exceeded, then a new space shall be allocated for this new sample. In this condition the easiest solution to this can be splitting the most suitable cluster into two (if it doesn't result in exceeding the cluster number) [43].

3.1.5 Evaluation Metrics

3.1.5.1 Homogeneity

Homogeneity criteria can only be satisfied if members of each single class are placed into a distinct cluster in terms of homogeneity[44]. So that each cluster significantly contains only members of a single class. The class distribution within each cluster should be done from a single class. Homogeneity gets a value between 0 and 1. For a perfect clustering, the homogeneity value gets the value 1.

As Y represents the data which belongs to the same class and T represents the clusters that the data would be clustered into, homogeneity value can be expressed as $\mathcal{H}(Y|T)$. The value of $\mathcal{H}(Y|T)$ is dependent on the size of the dataset. We use homogeneity value by its normalized form by $\mathcal{H}(Y)$ instead of its raw entropy value. While $\mathcal{H}(Y)$ could provide the maximum homogeneity value, this form can be expressed as;

$$\frac{(\mathcal{H}(Y|T))}{(\mathcal{H}(Y))} \quad (3.5)$$

In a perfect homogeneous situation, this normalization, $\frac{(\mathcal{H}(Y|T))}{(\mathcal{H}(Y))}$ equals to 0. Thus, as we know that 1 is desirable and 0 is undesirable condition, the homogeneity can be defined as:

$$h = \begin{cases} 1 & \text{if } \mathcal{H}(Y, T) = 0 \\ 1 - \frac{\mathcal{H}(Y|T)}{\mathcal{H}(Y)} & \text{else} \end{cases} \quad (3.6)$$

Where a dataset that consists of N data points and these data points belongs to Y number of classes which varies between $c = 1, \dots, Y$ and placed into T number of clusters which varies between $k = 1, \dots, T$. x_{ck} shows the number of data points belongs to the class c and is also an element of cluster k . n shows the quantity of number of classes.

$$\begin{aligned} \mathcal{H}(Y|T) &= - \sum_{k=1}^{|T|} \sum_{c=1}^{|Y|} \frac{x_{ck}}{N} \log \frac{x_{ck}}{\sum_{c=1}^{|Y|} x_{ck}} \\ \mathcal{H}(Y) &= - \sum_{c=1}^{|Y|} \frac{\sum_{k=1}^{|T|} x_{ck}}{n} \log \frac{\sum_{k=1}^{|T|} x_{ck}}{n} \end{aligned} \quad (3.7)$$

3.1.5.2 Completeness

This metric, which is symmetrical to the homogeneity metric, expresses the proportion of data belonging to the same class within the same dataset[44]. If the data belonging to the same data class is included in the same group as the result of the clustering of the dataset, this metric which takes the ideal value in this situation regarded as 1, if the grouping is farthest from ideal, the value of this metric will be 0.

In order to satisfy this criterion each of the clusters should comprise of elements which belongs only one class. Distribution of cluster assignments within each class is used to evaluate completeness. As Y represents the data which belongs to the same class and T represents the clusters that the data would be clustered into, completeness value can be expressed as $\mathcal{H}(T|Y)$. In ideal condition $\mathcal{H}(T|Y) = 0$. The worst case scenario is when each class is represented by each cluster and in this case $\mathcal{H}(T|Y) = \mathcal{H}(T) = 1$. Completeness can be defined as:

$$c = \begin{cases} 1 & \text{if } \mathcal{H}(T, Y) = 0 \\ 1 - \frac{\mathcal{H}(T|Y)}{\mathcal{H}(T)} & \text{else} \end{cases} \quad (3.8)$$

Where a dataset that consists of N data points and these data points belongs to Y number of classes which varies between $c = 1, \dots, Y$ and placed into T number of clusters which varies between $k = 1, \dots, T$, a_{ck} shows the number of data points which is an element of cluster k and is also belongs to the class c . n shows the quantity of number of classes.

$$\begin{aligned} \mathcal{H}(T|Y) &= - \sum_{c=1}^{|Y|} \sum_{k=1}^{|T|} \frac{a_{ck}}{N} \log \frac{a_{ck}}{\sum_{k=1}^{|T|} a_{ck}} \\ \mathcal{H}(T) &= - \sum_{k=1}^{|T|} \frac{\sum_{c=1}^{|Y|} a_{ck}}{n} \log \frac{\sum_{c=1}^{|Y|} a_{ck}}{n} \end{aligned} \quad (3.9)$$

3.1.5.3 V-Measure

V-Measure is a criterion which measures how successfully did homogeneity and completeness criteria have been satisfied[44]. V-Measure is calculated by taking the harmonic mean of the homogeneity and completeness metrics. This criterion takes values between 1 and 0. As described above in homogeneity and completeness sections, these two metric have working logic that are opposite to each other. Increase in homogeneity results in decrease in completeness, and vice versa. V-Measure can be calculated as:

$$v - Measure = \frac{(2 \times homogeneity \times completeness)}{(homogeneity + completeness)} \quad (3.10)$$

3.1.5.4 Adjusted Rand Index

A metric called "*Rand Index*" which is a measure of similarities between two data clusterings, should be calculated in order to get "*Adjusted RandIndex (ARI)*"[45]. While *Rand Index* may vary between 0 and 1, *Adjusted Rand Index* can also yield negative values. Rand index which is calculated separately for both the clustering which is expected to be ideal and the clustering that is currently made, and then these index values are used in the formula below to "*adjust*" the *Rand Index*:

$$ARI = \frac{(RI - \mathbb{E}(RI))}{(max(RI) - \mathbb{E}(RI))} \quad (3.11)$$

$\mathbb{E}(RI)$ shows the expected value which is a result of a set of calculations obtained from a contingency table which is formed by amount of the objects of the dataset which had been put in the same or different clusters with the compared clusters.

Adjusted Rand Index gets its perfect score when the clustering is random and independent of number of clusters, than the score would be 0. On the contrary if the clusters are identical and/or similar to each other, then the index becomes 1. This metric is symmetrical. So:

$$ARI(x, y) == ARI(y, x) \quad (3.12)$$

3.1.5.5 Adjusted Mutual Information

This metric is also an "adjusted" metric like *ARI*. Mutual information tells us how much information is shared between different clusters and this metric measures this information. So, *adjustedmutualinformation* [46] can be considered as a similarity measure. In our work this metric measures the number of mutual elements between different clusters. This metric is equal to 1 when the clusters are completely identical, and when clusters are independent from each other this metric becomes equal to 0. That means there is no information shared. Adjusted Mutual Information is the adjusted form of mutual information. The mutual information value is adjusted as below where U and K are the clusterings which will be under the scope:

$$AMI(U, T) = \frac{I(U, K) - E(MI(U, K))}{max(H(U), H(K)) - E(I(U, K))} \quad (3.13)$$

$E(MI(U, K))$ shows the expected value which is a result of a set of calculations obtained from a contingency table which is formed by amount of the objects of the dataset which had been put in the same or different clusters with the compared clusters.

This metric is also symmetrical like adjusted random information.

3.1.5.6 Silhouette Coefficient

Silhouette coefficient[47] is calculated by using both intra-cluster distance and mean nearest-cluster distance which is the distance between a sample and the nearest cluster that the element is not a part of for each of the elements in a dataset. The formula is:

$$\text{Silhouette Coeff.} = \frac{(y - x)}{\max(x, y)} \quad (3.14)$$

where y is the distance between an input instance and the nearest cluster which the instance doesn't belong and x is the mean value of distances within the cluster which the instance is a part of. To calculate the Silhouette coefficient, the dataset should have at least two clusters. This metric returns the mean value over all calculated Silhouette coefficient values for instances in dataset.

Silhouette coefficient varies between -1 and 1. When this metric is considered for an instance the more Silhouette coefficient is closer to -1, the more likely the instance is in wrong cluster. If this metric is considered for all dataset the more the value gets closer to -1, the more clustering isn't accurate and instances are more likely misplaced and clustering had put instances in clusters which they should not belong. On the contrary, the more this metric gets closer to 1, the more likely the clustering is accurate. When the value of this metric near 0, then probably clusters are overlapped.

3.2 Homomorphic Encryption

Homomorphic encryption is a form of encryption method that allows computation on encrypted data and generates result which would have been same result if the same computation would be performed on the plain data. As it can be seen from this main property of the method, the purpose of the method is to preserve privacy[48–51]. Homomorphic encryption can also be used in connecting different services without exposing their sensitive data. Homomorphical encryption algorithms can be expressed in two groups as partially homomorphic algorithms and fully homomorphic algorithms[52]. In our work, we used Paillier Cryptosystem and this is a additive partially homomorphic algorithm[53].

3.2.1 Paillier Cryptosystem

Paillier cryptosystem[54] is an asymmetric , probabilistic and public key cryptosystem. It preserves privacy[55] depending on difficulty of the problem of computing the n -th residue classes. Paillier cryptosystem is a additive partially homomorphic system, that means encryption of M_1 and M_2 plain datasets with a K public key gives the same result as the encryption of addition of same two dataset ($M_1 + M_2$) with using the same K public key. This encryption algorithm works by doing two main jobs in an order, first one is key generation and the second one is encryption/decryption of dataset.

As explained before Paillier Cryptographic system has homomorphic properties which makes this algorithm more convenient to be used in several fields. These properties are:

- Addition of encrypted data: Result of adding of two encrypted datasets matches with the result of enciphering and adding two datasets.
- Multiplication of encrypted data with a non-encrypted value: Multiplying an encrypted data with a number N is same with multiplying the plain form of that data with the same number N and encrypting it.

3.2.2 Floating Point Numbers

In this work, floating point numbers are used to express data which has been encrypted using Paillier Cryptosystem in the python environment and to express the values in the datasets we used in this work. As a natural consequence, the operations on the data are also based on the values defined in this type.

The number of digits in the fraction part of the data that is defined as floating point can be very large and if these numbers would be used than it would definitely cost more processing power and processing time, so in this work only the first 5 fractional digits have been used. Because of the 5 digit limit has been put on the fractional digits of input data and more than these digits are not used, which has been possibly generated as a result of computational work, are rounded into 5 digits and this situation possibly creates minimal data losses or deviation of computation [56]. It is seen that these effects on the calculation result depends on the grouping algorithm used, but overall it is low in effectiveness [57].

Chapter 4

System Model

4.1 Development Environment

In this thesis, Python 2017.3.3 community edition is used for algorithm programming which encrypts data using Paillier cryptography and makes clustering using 4 different algorithms which have been described in section 3. Each algorithm run on a computer which has Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz octa core processor (4 real +4 pseudo) along with 16GB of RAM. Parallel computing has not been used everytime but while some algorithms allow to use all processor cores while running, that property has been used.

4.2 Sequence Diagram

In this section, the system which is designed to preserve privacy while handling private or sensitive data is explained by sequence diagrams. This system doesn't only aim to preserve privacy, but also aims to handle the data efficiently in terms of time and processing power.

4.2.1 Client Computaion

Client side doesn't hold the data which needs to be handled, because management and maintenance of a big data storage brings unnecessary extra cost. In this work client of this system is considered as an ordinary PC user, so there is no data storage for big data systems on client side and the main task of client side is just using/handling sensitive data when it is needed. The client generates public and private key pair by establishing a

key exchange session with data authority, then sends its public key and asks for the data needed. As the data needs to stay encrypted and not revealed to client, at this point, data computing cloud starts to handle the data.

Data Computing cloud also doesn't store the sensitive data but preserves enough processing power for handling the encrypted data. On computing cloud servers, the mathematical computations run to compute an encrypted distance matrix from the encrypted data in a form that can be used by the client. Cloud servers perform needed calculations for client side without violating the data privacy and sends the results, which is in our case the encrypted distance matrix. Client side then uses the encrypted matrix in order to build a model and evaluate the data. As seen in Figure 4.1, in this system client side doesn't need the actual data to use because enough information can be derived from the distance matrix.

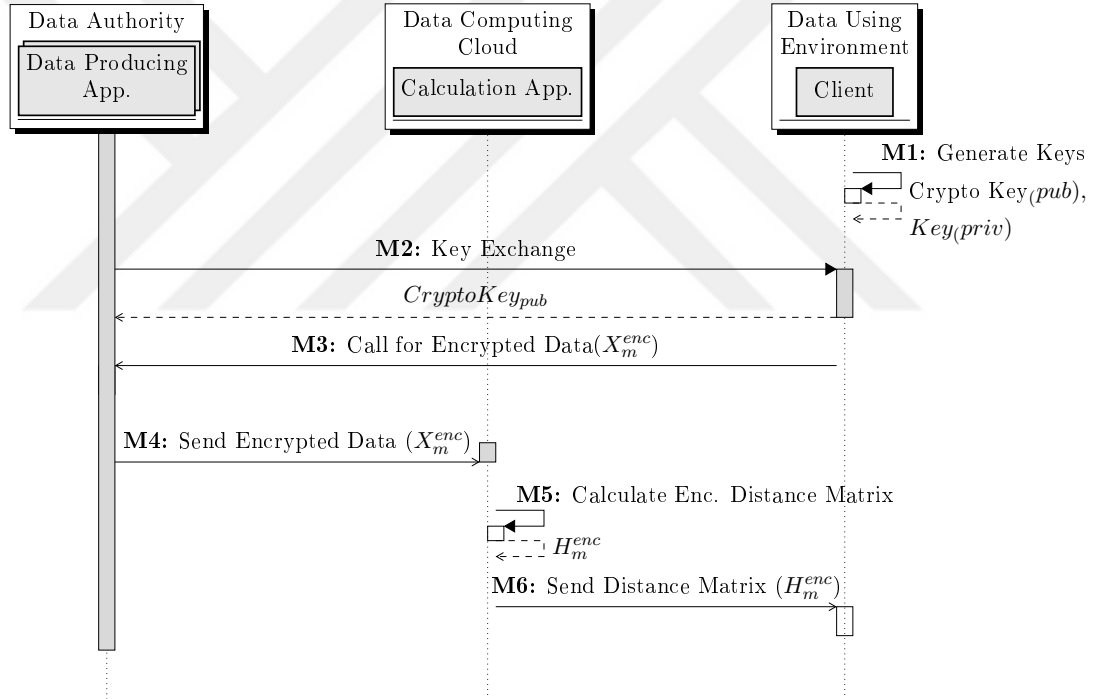


FIGURE 4.1: Sequence Diagram for Client Side

The pseudo code for *Client Computation* part of the system is shown in Algorithm 1. In key generation step, pseudo code doesn't contain explanation of step-by-step key generation. As the Paillier key generation is a generic model, we didn't need to explain those steps in details (choosing two prime numbers or choosing exponents etc.).

Algorithm 1: Client Computation

```

1 begin
2   begin Key Generation & Key Exchange
3      $(CryptoKey_{pub}, Key_{priv}) \leftarrow$  Key Generation
4     Send  $CryptoKey_{pub}$  to Data Authority
5     Data Auth.  $\leftarrow CryptoKey_{pub}$ 
6   for  $m \in \mathbb{N}$  do
7     begin Asking for Enc. Data
8       ask for  $(X_m^{enc})$  from Data Auth.
9       Data Auth.  $\leftarrow$  Client asks for  $(X_m^{enc})$ 
10      Data Comp. Cloud  $\leftarrow$  Data Auth. sends  $(X_m^{enc})$ 
11    begin Calculation of Enc. Dist. Matrix
12      send  $(X_m^{enc})$  to Data Auth.
13       $H_m^{enc}$  is calculated  $\leftarrow$  from  $(X_m^{enc})$ 
14    Client  $\leftarrow$  Data Comp. Cloud sends  $(H_m^{enc})$ 
15

```

4.2.2 Data Authority Computation

In our work client doesn't has to maintain a storage big enough for handling big data, as this condition is described in client computation section. Providing data storage for the system is a responsibility for data authority (this was also described in client computation section), but this is not the main duty. In this system data authority isn't only the storage location, but also the sensitive data producer which encrypts and stores the data as its main duty. In respect to our system design, there is no regulation that forces the system to work with a single data authority. Instead, in reality, there should be a large number of data authorities.

Data authority creates, stores and most importantly encrypts the data, with using the keys gathered from the key exchange session conducted between itself and client side, and sends the encrypted form of the data to data computing cloud as the client needs to evaluate. In data authority side Paillier Cryptosystem is used as encryption scheme and the key is the public key of client side. On this side, data privacy is preserved and pure data isn't revealed to any part of the system. Encryption of pure data and its transmission is done as explained in Algorithm 2 and it can be seen that pure data is not revealed to any party, but just its encrypted form is transmitted to data computing cloud as seen in Figure 4.2.

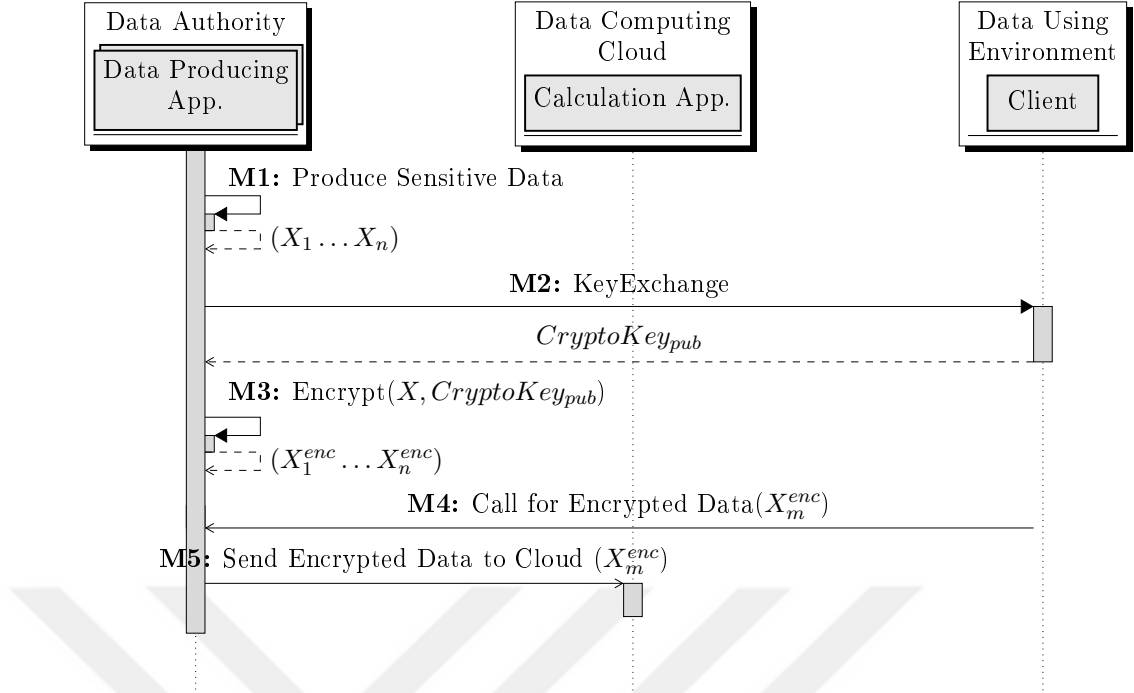


FIGURE 4.2: Sequence Diagram for Data Authority Side

Algorithm 2: Data Authority Computation

```

1 begin
2   begin Producing the Sensitive Data
3     Data Auth.  $\leftarrow (X_1 \dots X_n)$  as  $n \in \mathbb{N}$ 
4   begin Initiate Key Exchange Session with Client
5     Data Auth.  $\leftarrow CryptoKey_{pub}$  of Client
6   for  $n \in \mathbb{N}$  do
7     begin Encryption of Sensitive Data
8        $(X_1^{enc} \dots X_n^{enc})$  is calculated  $\leftarrow$  from  $((X_1 \dots X_n), CryptoKey_{pub})$ 
9     begin Sending Encrypted Data to Data Comp. Cloud
10      Data Comp. Cloud  $\leftarrow (X_1^{enc} \dots X_n^{enc})$ 
11    end
12  end
13 end
  
```

4.2.3 Model Building at Client

In this work, the client side is just an ordinary PC user (as explained before). After the encrypted distance matrix has been computed on Data Computing Servers, it is sent to the client side. The distance matrix has enough knowledge to create a model because of the Paillier Cryptosystem's specialty of homomorphic behavior. Once the calculated distance matrix reaches to client side, it is decrypted by using the *private key* of client. After considering which algorithm will be used in order to build the model, that algorithm uses the decrypted distance matrix as input and the result is evaluated as client side's will.

As seen in Algorithm 3, the clustering algorithm should be determined every time at the client side because predetermining the algorithm would be meaningless as every data wouldn't possess same property and different algorithms would satisfy different needs.

As seen in Figure 4.3, client side receives just the encrypted form of distance matrix and decrypts it using its private key. Evaluation of data starts just after deciding which clustering algorithm will be used. Decrypted data enters into the clustering algorithm and the result of the clustering process is a raw data on client side.

Output of the clustering process possesses valuable information that needs to be modeled but at this point, client side will need extra info about contents of clusters to explain them accurately and build up a meaningful model from them. This info can be gathered by creating an extra session between client and data authority and asking to get it in a form of a metric to interpret the clusterings.

In this system, this info can be gathered by client while the client calls for the encrypted data from data authority so the info can be gathered with the needed data. As a second method, this information can also be gathered by establishing an extra session with data authority after the clustering process. This information must be asked from data authority, not from computing cloud because it is not a trusted party and its only job is to overcome the difficulty of making computations on encrypted data. This info is also as sensitive as the data itself and must be delivered in encrypted form.

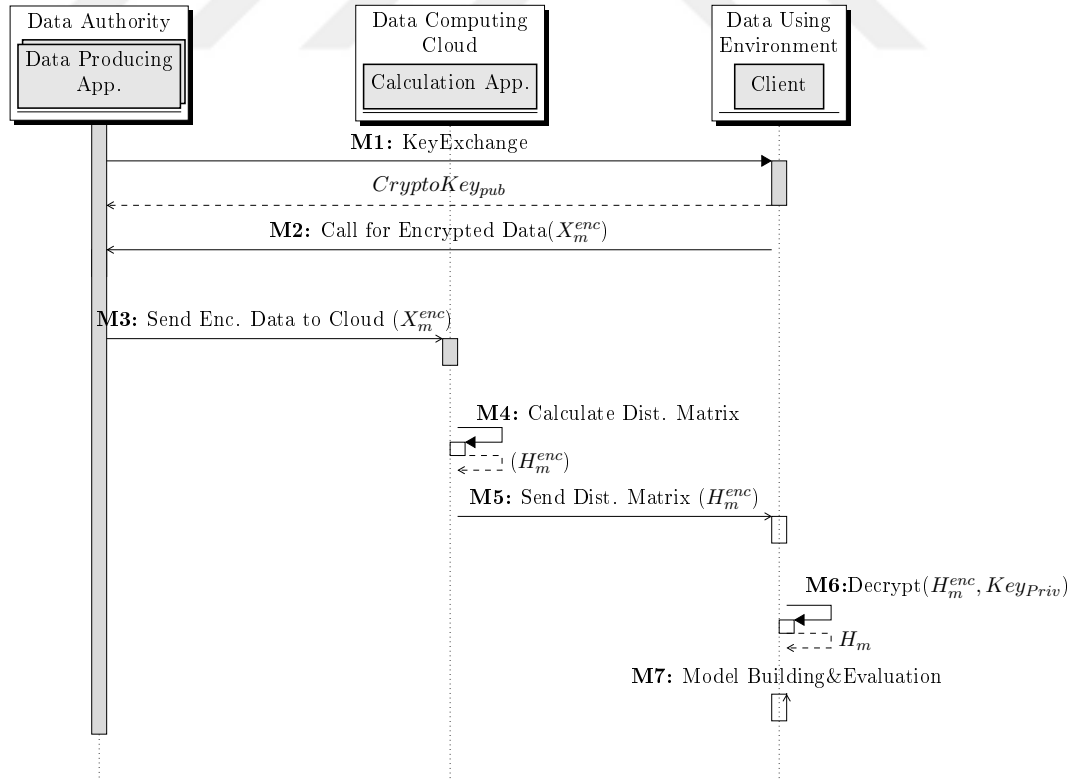


FIGURE 4.3: Sequence Diagram for Model Building at Client Side

Algorithm 3: Model Building at Client

```

1 begin
2   begin
3     Client  $\leftarrow$  (CryptoKeypub, Keypriv)
4     Data Auth.  $\leftarrow$  Client asks for ( $X_m^{enc}$ )
5     for  $m \in \mathbb{N}$  do
6       begin Calculation of Enc. Dist. Matrix
7         Data Comp. Cloud  $\leftarrow$  Data Auth. sends ( $X_m^{enc}$ )
8         ( $H_m^{enc}$ ) is calculated  $\leftarrow$  from ( $X_m^{enc}$ )
9       Client  $\leftarrow$  ( $H_m^{enc}$ ) from Data Comp. Cloud
10      for  $m \in \mathbb{N}$  do
11        begin Decryption of Distance Matrix
12          ( $H_m$ ) is calculated at Client  $\leftarrow$  from ( $H_m^{enc}$ , Keypriv)
13        begin Model Building at Client
14          Clustering Algorithm  $\leftarrow$  ( $H_m$ )
15          Client  $\leftarrow$  Clustered Pure Data

```

Chapter 5

Experiments and Results

In this work, clustering methods which have been described in section 3 have been studied by running each of them on 10 different datasets (from 500 to 5000 rows) with using 5 different bit lengths of keys. Datasets are produced/chosen different from each other by their data and data lengths. The computer used in our experiments has limited computational capacity to calculate distance matrix especially when 512 and 1024 bit long keys are used on data which has more than 5000 rows. The data length of each dataset is also the name of the dataset (dataset 500, dataset 1000 etc.) and used key lengths vary between 64 bit and 1024 bits.

Experimental results have tables which include evaluation metric scores under the name of "*Plain/Encrypted domain clustering results*" based on the chosen dataset and domain. Plain data results have same time result for each key length, because for plain domain encryption there is no calculation using a key. Evaluation metrics have (except *silhouette coefficient*) maximum scores because of the used data is *artificial* and not *real*.

All algorithms have been run on python environment (as explained in section 4.1.) and all algorithm codes have been modified to create same number of clusters (*20 clusters*) from given data. Just in the case of *Birch Algorithm* on figures that show distribution of clusters, number of clusters is as 19, because the cluster numbers are varying between 0 and 19.

5.1 Plaintext Results

5.1.1 K-Means Algorithm Results

From Table 5.1 to 5.10 and Figure 5.1 to 5.10 we will see the distribution of *plain* data which its length varies between 500 and 5000 due to K-Means algorithm. *K – Means* algorithm forms clusters considering *distances* between points. This algorithm is used generally when the data that needs to be clustered has flat geometry, creating too many clusters is not necessary and created clusters are even sized.

TABLE 5.1: Plain domain evaluation metric scores for dataset 500

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.362	9.844827

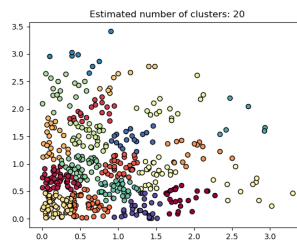


FIGURE 5.1: Plain domain clustering for dataset 500

TABLE 5.2: Plain domain evaluation metric scores for dataset 1000

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.350	36.957112

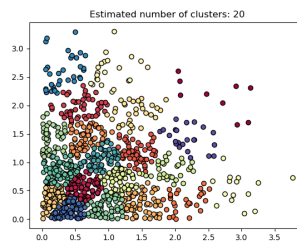


FIGURE 5.2: Plain domain clustering for dataset 1000

TABLE 5.3: Plain domain evaluation metric scores for dataset 1500

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.359	86.728137

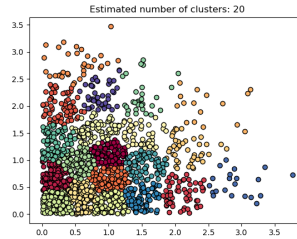


FIGURE 5.3: Plain domain clustering for dataset 1500

TABLE 5.4: Plain domain evaluation metric scores for dataset 2000

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.352	163.580145

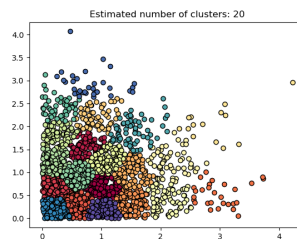


FIGURE 5.4: Plain domain clustering for dataset 2000

TABLE 5.5: Plain domain evaluation metric scores for dataset 2500

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.346	271.935549

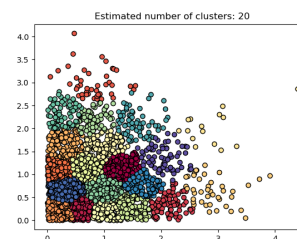


FIGURE 5.5: Plain domain clustering for dataset 2500

TABLE 5.6: Plain domain evaluation metric scores for dataset 3000

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.338	413.855393

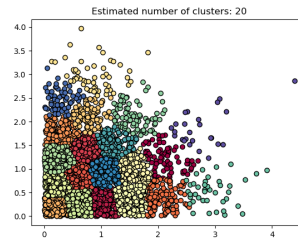


FIGURE 5.6: Plain domain clustering for dataset 3000

TABLE 5.7: Plain domain evaluation metric scores for dataset 3500

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.338	623.065649

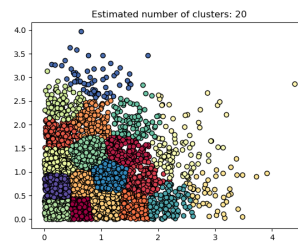


FIGURE 5.7: Plain domain clustering for dataset 3500

TABLE 5.8: Plain domain evaluation metric scores for dataset 4000

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.345	1048.278932

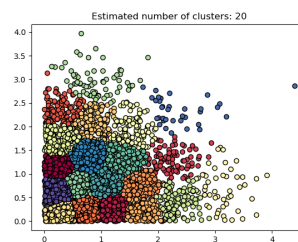


FIGURE 5.8: Plain domain clustering for dataset 4000

TABLE 5.9: Plain domain evaluation metric scores for dataset 4500

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.349	1292.195140

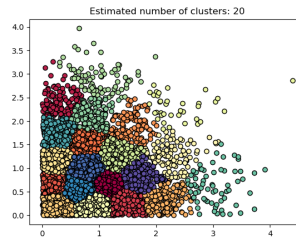


FIGURE 5.9: Plain domain clustering for dataset 4500

TABLE 5.10: Plain domain evaluation metric scores for dataset 5000

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.340	1292.195140

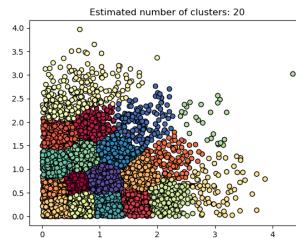


FIGURE 5.10: Plain domain clustering for dataset 5000

5.1.2 Hierarchical Algorithm Results

From Table 5.11 to 5.20 and Figure 5.11 to 5.20 we will see the distribution of *plain* data which its length varies between 500 and 5000 due to Hierarchical algorithm. *Hierarchical* algorithm forms clusters considering *pairwise distances* between points. This algorithm is used generally when creating too many clusters is necessary and there are possible connectivity constraints to form clusters.

TABLE 5.11: Plain domain evaluation metric scores for dataset 500

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.295	2.297151

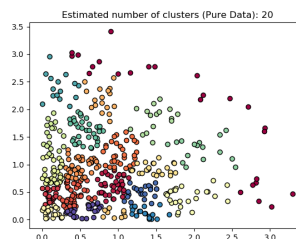


FIGURE 5.11: Plain domain clustering for dataset 500

TABLE 5.12: Plain domain evaluation metric scores for dataset 1000

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.287	14.892210

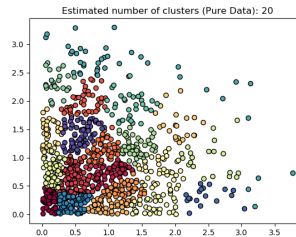


FIGURE 5.12: Plain domain clustering for dataset 1000

TABLE 5.13: Plain domain evaluation metric scores for dataset 1500

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.276	53.347093

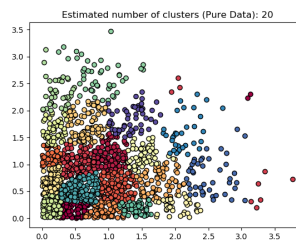


FIGURE 5.13: Plain domain clustering for dataset 1500

TABLE 5.14: Plain domain evaluation metric scores for dataset 2000

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.285	123.057987

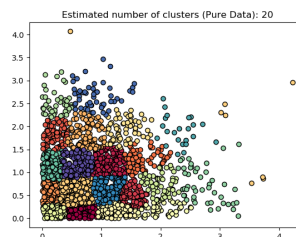


FIGURE 5.14: Plain domain clustering for dataset 2000

TABLE 5.15: Plain domain evaluation metric scores for dataset 2500

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.305	223.921661

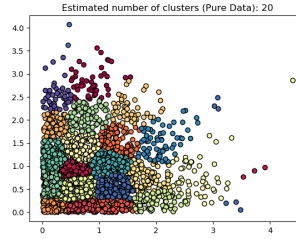


FIGURE 5.15: Plain domain clustering for dataset 2500

TABLE 5.16: Plain domain evaluation metric scores for dataset 3000

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.297	410.769117

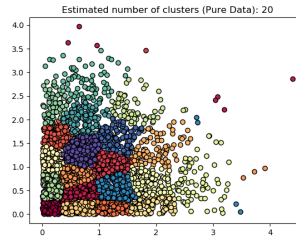


FIGURE 5.16: Plain domain clustering for dataset 3000

TABLE 5.17: Plain domain evaluation metric scores for dataset 3500

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.298	606.139748

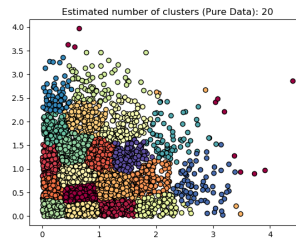


FIGURE 5.17: Plain domain clustering for dataset 3500

TABLE 5.18: Plain domain evaluation metric scores for dataset 4000

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.296	964.552434

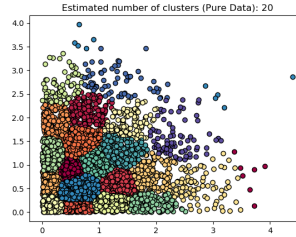


FIGURE 5.18: Plain domain clustering for dataset 4000

TABLE 5.19: Plain domain evaluation metric scores for dataset 4500

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.294	1206.877016

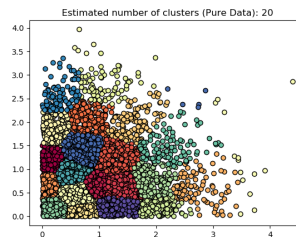


FIGURE 5.19: Plain domain clustering for dataset 4500

TABLE 5.20: Plain domain evaluation metric scores for dataset 5000

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.304	1592.067019

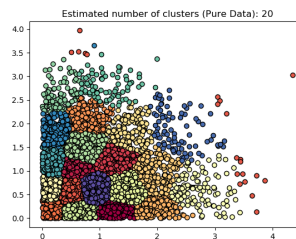


FIGURE 5.20: Plain domain clustering for dataset 5000

5.1.3 Spectral Algorithm Results

From Table 5.21 to 5.30 and Figure 5.21 to 5.30 we will see the distribution of *plain* data which its length varies between 500 and 5000 due to Spectral algorithm. *Spectral*

algorithm forms clusters considering *nearest – neighbors*. This algorithm is used generally when the data that needs to be clustered has non-flat geometry, creating many clusters is not necessary and created clusters are even sized.

TABLE 5.21: Plain domain evaluation metric scores for dataset 500

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.309	4.687996

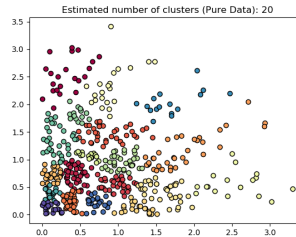


FIGURE 5.21: Plain domain clustering for dataset 500

TABLE 5.22: Plain domain evaluation metric scores for dataset 1000

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.320	8.625928

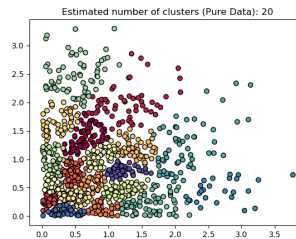


FIGURE 5.22: Plain domain clustering for dataset 1000

TABLE 5.23: Plain domain evaluation metric scores for dataset 1500

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.288	16.470525

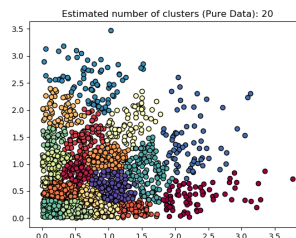


FIGURE 5.23: Plain domain clustering for dataset 1500

TABLE 5.24: Plain domain evaluation metric scores for dataset 2000

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.300	26.471610

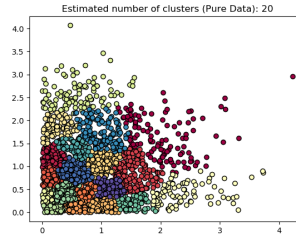


FIGURE 5.24: Plain domain clustering for dataset 2000

TABLE 5.25: Plain domain evaluation metric scores for dataset 2500

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.304	39.410522

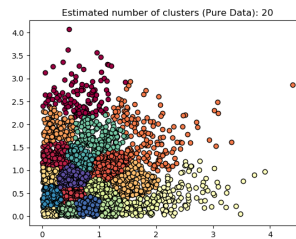


FIGURE 5.25: Plain domain clustering for dataset 2500

TABLE 5.26: Plain domain evaluation metric scores for dataset 3000

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.289	53.833924

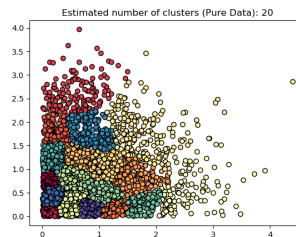


FIGURE 5.26: Plain domain clustering for dataset 3000

TABLE 5.27: Plain domain evaluation metric scores for dataset 3500

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.301	71.788992

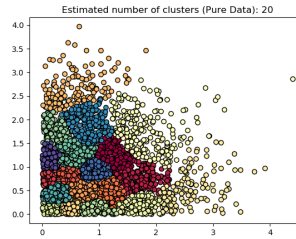


FIGURE 5.27: Plain domain clustering for dataset 3500

TABLE 5.28: Plain domain evaluation metric scores for dataset 4000

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.296	98.604372

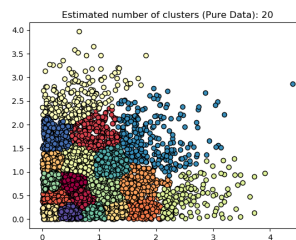


FIGURE 5.28: Plain domain clustering for dataset 4000

TABLE 5.29: Plain domain evaluation metric scores for dataset 4500

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.294	121.950676

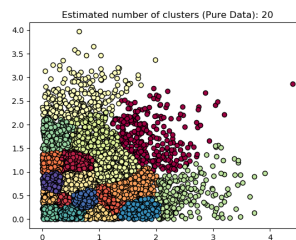


FIGURE 5.29: Plain domain clustering for dataset 4500

TABLE 5.30: Plain domain evaluation metric scores for dataset 5000

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.310	148.259339

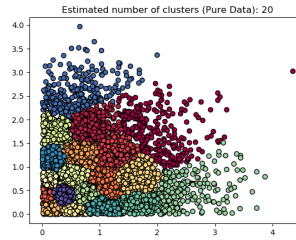


FIGURE 5.30: Plain domain clustering for dataset 5000

5.1.4 Birch Algorithm Results

From Table 5.31 to 5.40 and Figure 5.31 to 5.40 we will see the distribution of *plain* data which its length varies between 500 and 5000 due to Spectral algorithm. *Birch* algorithm forms clusters considering *Euclidean distance* between points. This algorithm is used generally when the data that needs to be clustered is large and data reduction in respect to outlier removal is needed.

TABLE 5.31: Plain domain evaluation metric scores for dataset 500

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.334	2.984673

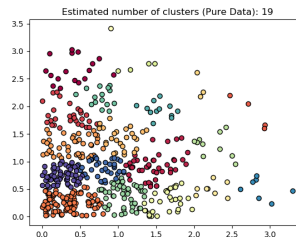


FIGURE 5.31: Plain domain clustering for dataset 500

TABLE 5.32: Plain domain evaluation metric scores for dataset 1000

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.299	15.079737

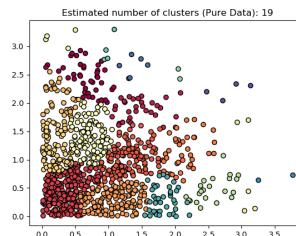


FIGURE 5.32: Plain domain clustering for dataset 1000

TABLE 5.33: Plain domain evaluation metric scores for dataset 1500

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.299	42.676480

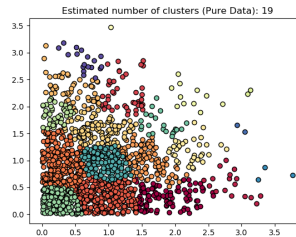


FIGURE 5.33: Plain domain clustering for dataset 1500

TABLE 5.34: Plain domain evaluation metric scores for dataset 2000

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.289	96.952344

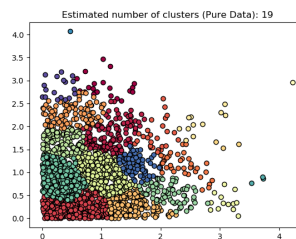


FIGURE 5.34: Plain domain clustering for dataset 2000

TABLE 5.35: Plain domain evaluation metric scores for dataset 2500

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.294	213.186763

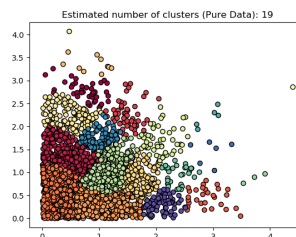


FIGURE 5.35: Plain domain clustering for dataset 2500

TABLE 5.36: Plain domain evaluation metric scores for dataset 3000

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.294	347.996374

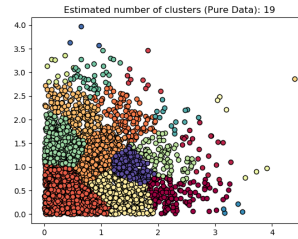


FIGURE 5.36: Plain domain clustering for dataset 3000

TABLE 5.37: Plain domain evaluation metric scores for dataset 3500

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.282	525.583719

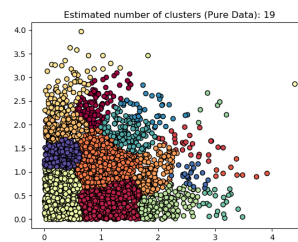


FIGURE 5.37: Plain domain clustering for dataset 3500

TABLE 5.38: Plain domain evaluation metric scores for dataset 4000

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.261	795.867486

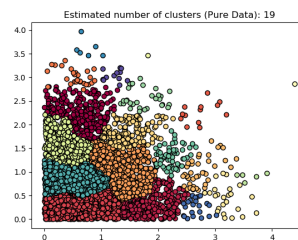


FIGURE 5.38: Plain domain clustering for dataset 4000

TABLE 5.39: Plain domain evaluation metric scores for dataset 4500

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.287	1106.477862

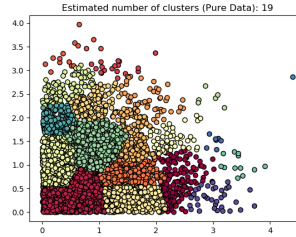


FIGURE 5.39: Plain domain clustering for dataset 4500

TABLE 5.40: Plain domain evaluation metric scores for dataset 5000

Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
1000	1000	1000	1000	1000	0.276	1509.153672

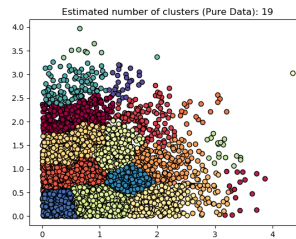


FIGURE 5.40: Plain domain clustering for dataset 5000

5.2 Encrypted Domain Results

5.2.1 K-Means Algorithm Results

From Table 5.41 to 5.50 and Figure 5.41 to 5.50 we will see the distribution of *encrypted* data which its length varies between 500 and 5000 due to K-Means algorithm. *K-Means* algorithm forms clusters considering *distances* between points. This algorithm is used generally when the data that needs to be clustered has flat geometry, creating too many clusters is not necessary and created clusters are even sized.

TABLE 5.41: Encrypted domain evaluation metric scores for dataset 500

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.787	0.778	0.783	0.615	0.745	0.362	31.094117
128	0.855	0.838	0.846	0.738	0.814	0.362	36.25088
256	0.855	0.838	0.846	0.738	0.814	0.362	56.539257
512	0.855	0.838	0.846	0.738	0.814	0.362	201.91177
1024	0.741	0.724	0.732	0.488	0.705	0.350	1124.906108

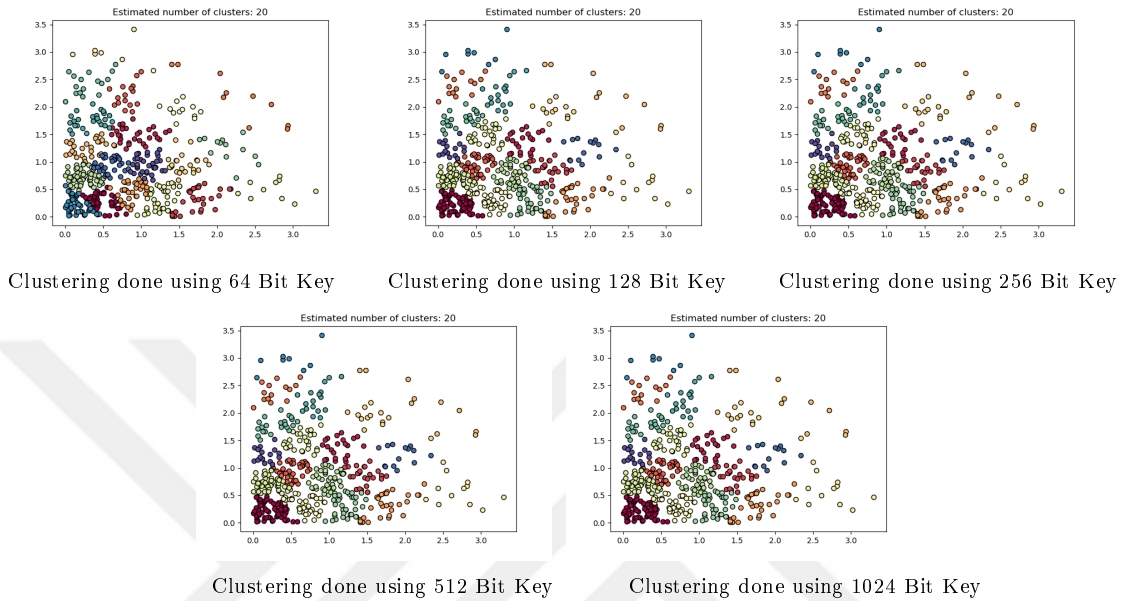


FIGURE 5.41: Encrypted domain clustering results for dataset 500

TABLE 5.42: Encrypted domain evaluation metric scores for dataset 1000

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.768	0.742	0.755	0.542	0.724	0.350	125.628153
128	0.741	0.724	0.732	0.488	0.705	0.350	144.886491
256	0.741	0.724	0.732	0.488	0.705	0.350	228.416989
512	0.741	0.724	0.732	0.488	0.705	0.350	725.917815
1024	0.741	0.724	0.732	0.488	0.705	0.350	4186.542025

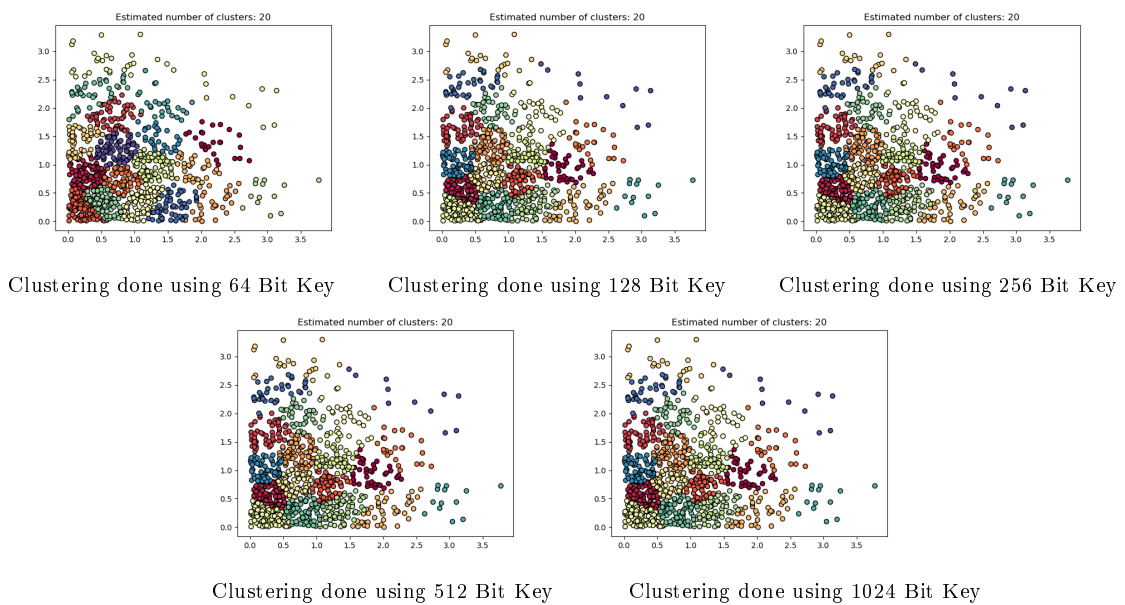


FIGURE 5.42: Encrypted domain clustering results for dataset 1000

TABLE 5.43: Encrypted domain evaluation metric scores for dataset 1500

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.811	0.802	0.807	0.691	0.794	0.359	293.46373
128	0.905	0.897	0.901	0.863	0.892	0.359	334.0554
256	0.905	0.897	0.901	0.863	0.892	0.359	523.392246
512	0.905	0.897	0.901	0.863	0.892	0.359	1682.56815
1024	0.905	0.897	0.901	0.863	0.892	0.359	8325.129793

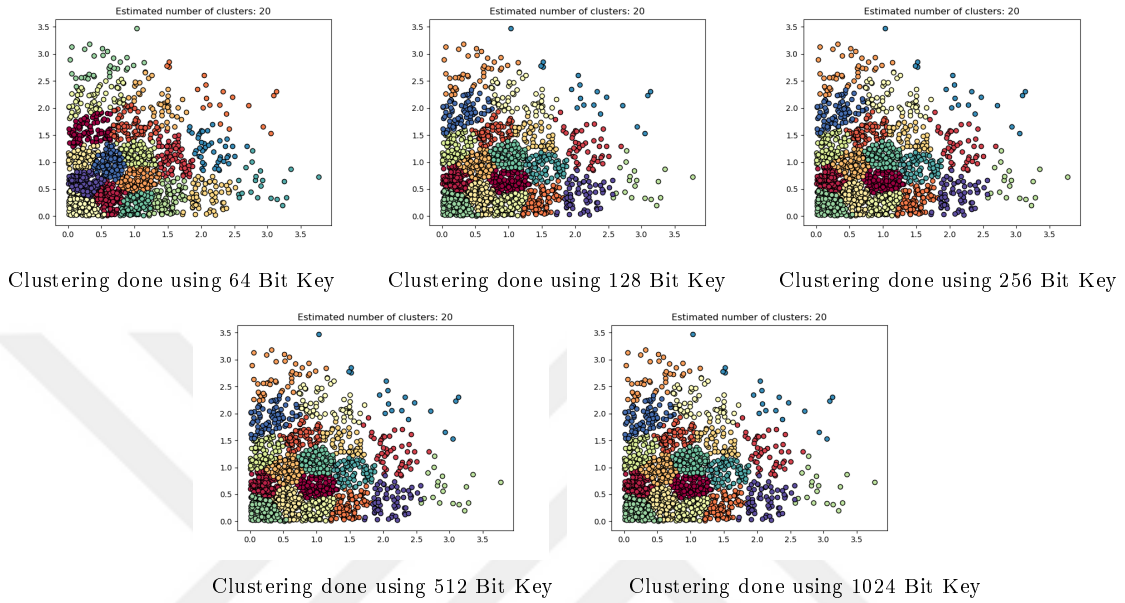


FIGURE 5.43: Encrypted domain clustering results for dataset 1500

TABLE 5.44: Encrypted domain evaluation metric scores for dataset 2000

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.834	0.815	0.824	0.753	0.809	0.352	488.452087
128	0.811	0.793	0.802	0.720	0.786	0.352	535.880552
256	0.811	0.793	0.802	0.720	0.786	0.352	849.303891
512	0.811	0.793	0.802	0.720	0.786	0.352	2994.485032
1024	0.811	0.793	0.802	0.720	0.786	0.352	15613.316685

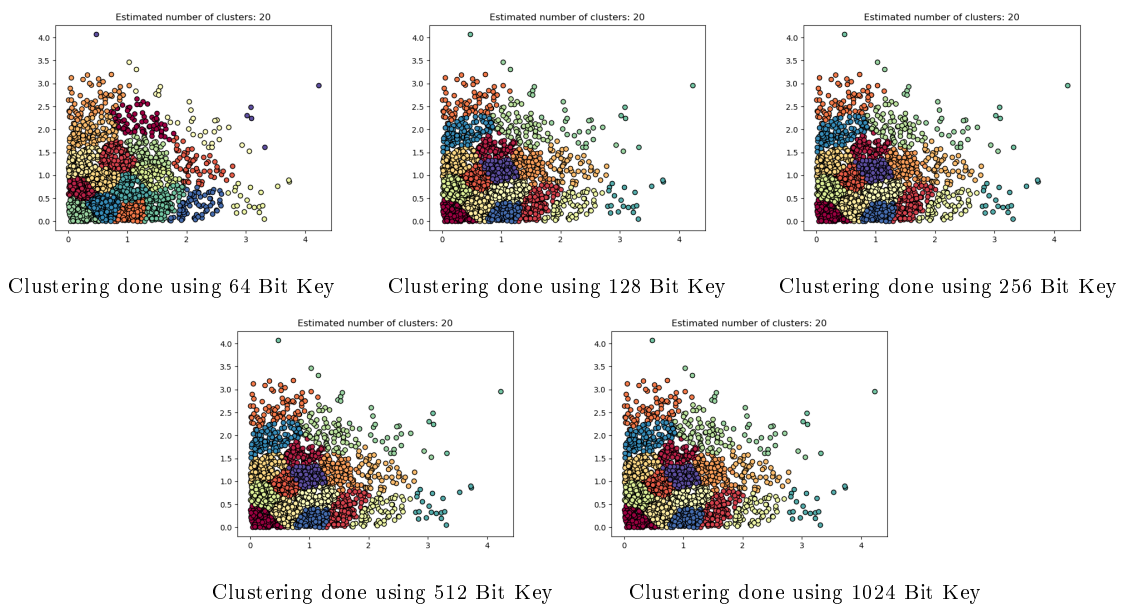


FIGURE 5.44: Encrypted domain clustering results for dataset 2000

TABLE 5.45: Encrypted domain evaluation metric scores for dataset 2500

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.807	0.778	0.793	0.662	0.773	0.346	688.117421
128	0.773	0.766	0.769	0.607	0.759	0.346	785.466996
256	0.773	0.766	0.769	0.607	0.759	0.346	1226.975066
512	0.773	0.766	0.769	0.607	0.759	0.346	4152.607761
1024	0.773	0.766	0.769	0.607	0.759	0.346	23101.712638

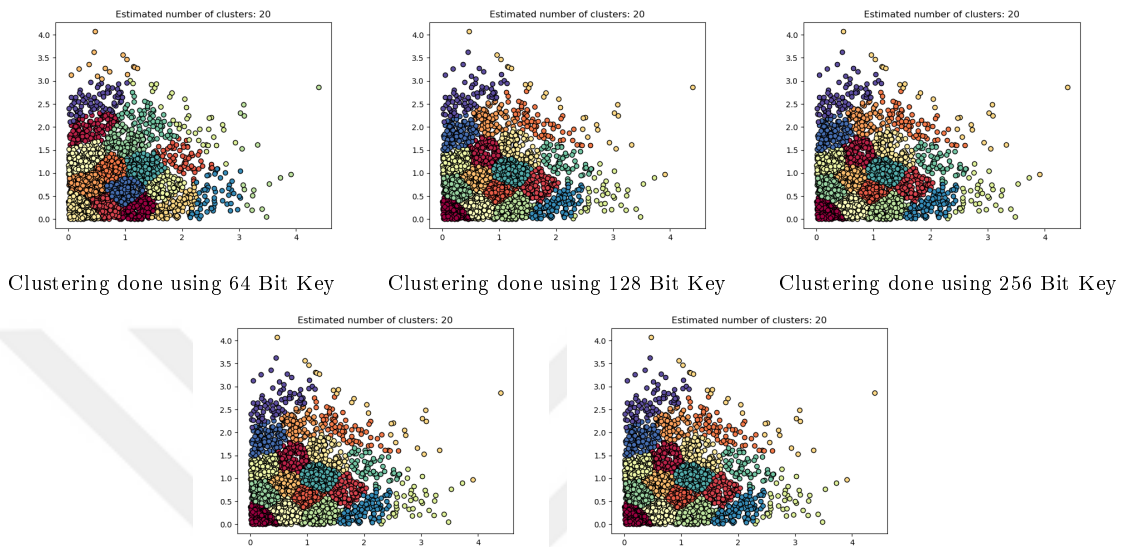


FIGURE 5.45: Encrypted domain clustering results for dataset 2500

TABLE 5.46: Encrypted domain evaluation metric scores for dataset 3000

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.736	0.720	0.728	0.528	0.714	0.338	1014.156519
128	0.743	0.727	0.735	0.543	0.721	0.338	1157.243526
256	0.743	0.727	0.735	0.543	0.721	0.338	2056.959974
512	0.743	0.727	0.735	0.543	0.721	0.338	5541.251946
1024	0.743	0.727	0.735	0.543	0.721	0.338	34827.048195

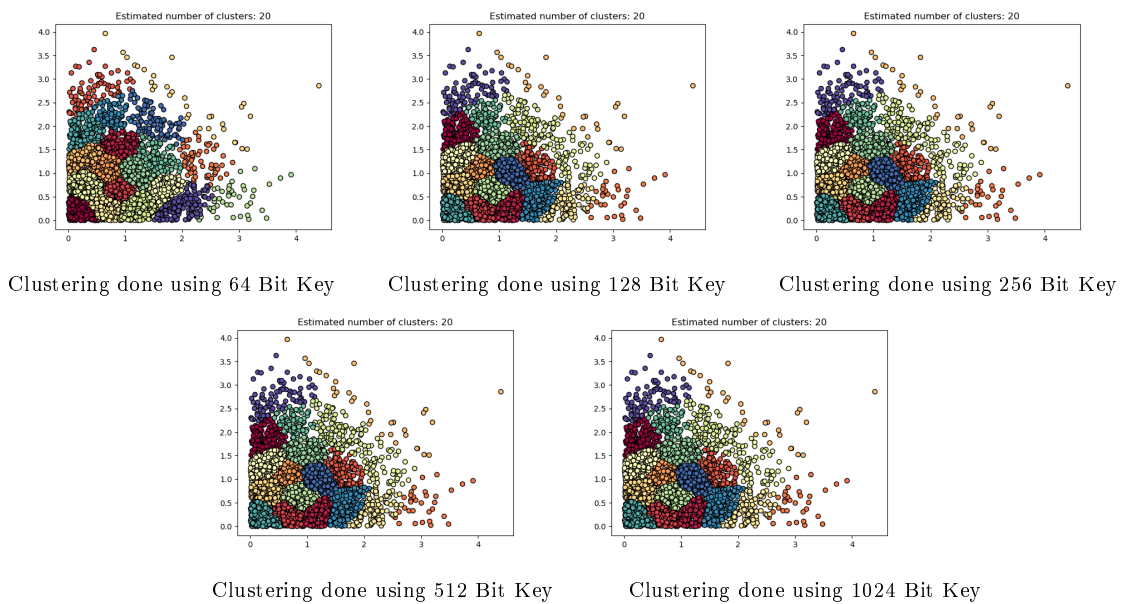


FIGURE 5.46: Encrypted domain clustering results for dataset 3000

TABLE 5.47: Encrypted domain evaluation metric scores for dataset 3500

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.810	0.791	0.800	0.685	0.787	0.338	1457.896283
128	0.835	0.822	0.828	0.745	0.819	0.338	1586.997932
256	0.835	0.822	0.828	0.745	0.819	0.338	2471.958513
512	0.835	0.822	0.828	0.745	0.819	0.338	7559.3366
1024	0.835	0.822	0.828	0.745	0.819	0.338	42589.211932

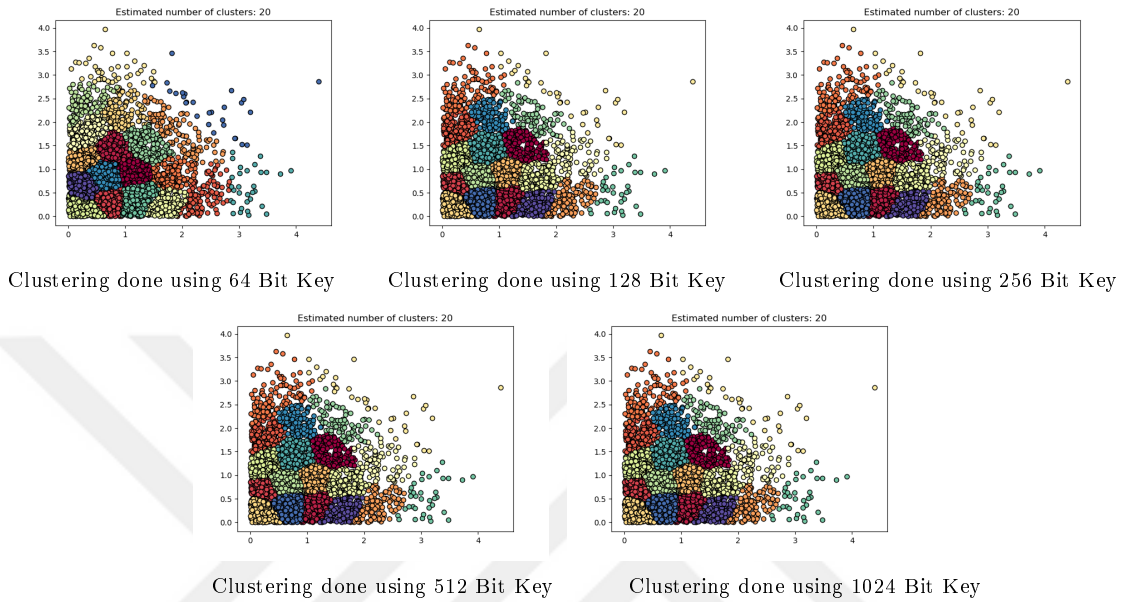


FIGURE 5.47: Encrypted domain clustering results for dataset 3500

TABLE 5.48: Encrypted domain evaluation metric scores for dataset 4000

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.776	0.762	0.769	0.623	0.758	0.345	1823.396682
128	0.795	0.781	0.788	0.671	0.788	0.345	2109.998674
256	0.795	0.781	0.788	0.671	0.788	0.345	3194.871339
512	0.795	0.781	0.788	0.671	0.788	0.345	10375.714473
1024	0.795	0.781	0.788	0.671	0.788	0.345	56810.999143

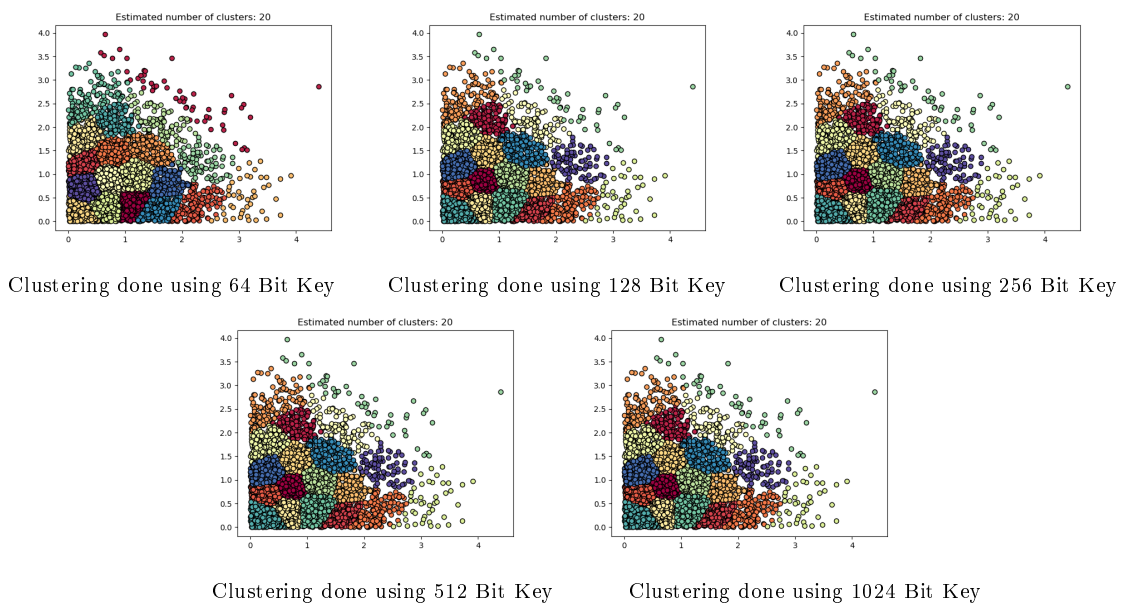


FIGURE 5.48: Encrypted domain clustering results for dataset 4000

TABLE 5.49: Encrypted domain evaluation metric scores for dataset 4500

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.696	0.686	0.691	0.465	0.682	0.349	2872.369434
128	0.763	0.747	0.755	0.585	0.744	0.349	3251.606473
256	0.763	0.747	0.755	0.585	0.744	0.349	4154.388187
512	0.763	0.747	0.755	0.585	0.744	0.349	14540.128381
1024	0.763	0.747	0.755	0.585	0.744	0.349	74140.224168

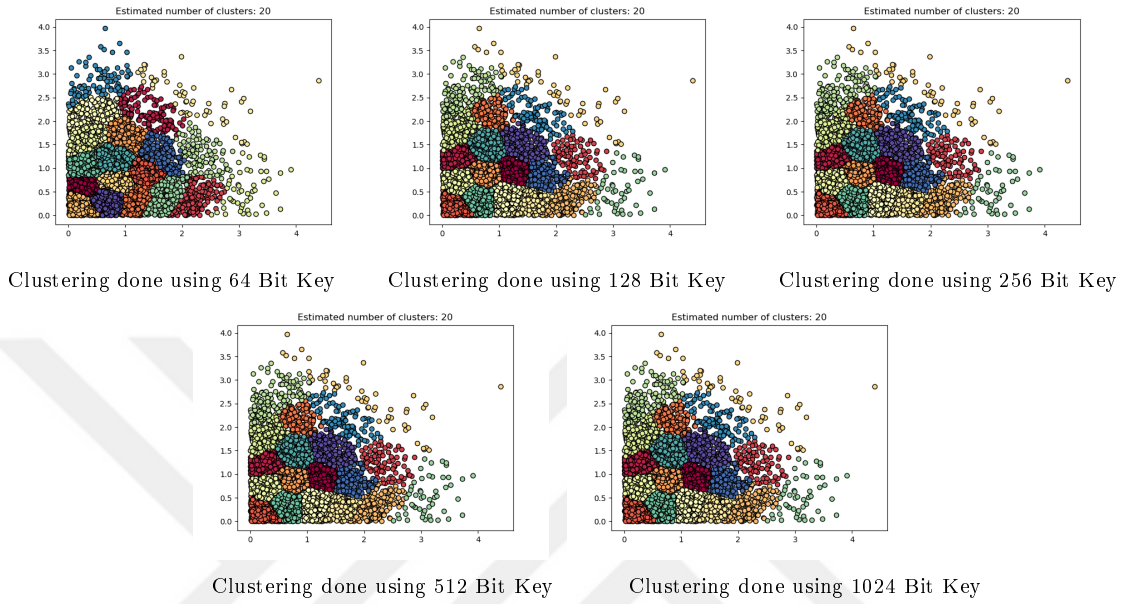


FIGURE 5.49: Encrypted domain clustering results for dataset 4500

TABLE 5.50: Encrypted domain evaluation metric scores for dataset 5000

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.746	0.733	0.740	0.574	0.730	0.340	3306.081361
128	0.766	0.751	0.759	0.604	0.748	0.340	4061.713102
256	0.766	0.751	0.759	0.604	0.748	0.340	6257.986052
512	0.766	0.751	0.759	0.604	0.748	0.340	15636.902162
1024	0.766	0.751	0.759	0.604	0.748	0.340	93672.799487

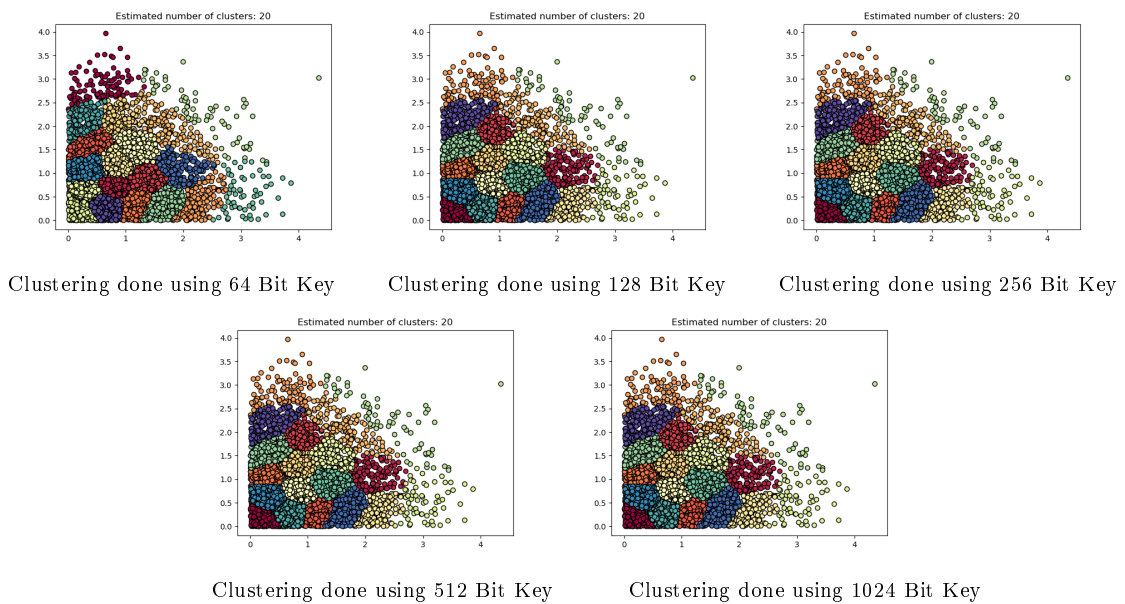


FIGURE 5.50: Encrypted domain clustering results for dataset 5000



FIGURE 5.51: Charts that show change of each evaluation metrics score by data length for different key lengths and for K-Means Algorithm

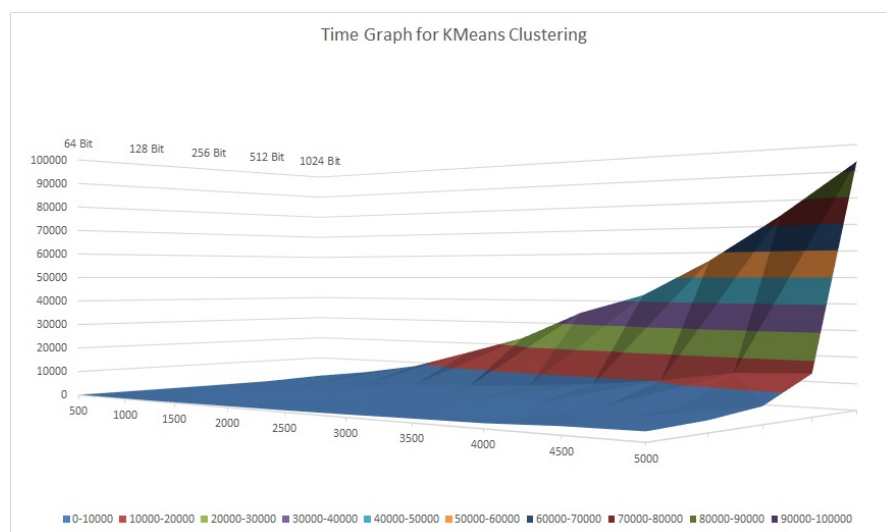


FIGURE 5.52: Calculation Time Graph for KMeans Algorithm on Key-Data Length Dimensions

5.2.2 Hierarchical Algorithm Results

From Table 5.51 to 5.60 and Figure 5.52 to 5.61 we will see the distribution of *encrypted* data which its length varies between 500 and 5000 due to Hierarchical algorithm. *Hierarchical* algorithm forms clusters considering *pairwise distances* between points. This algorithm is used generally when creating too many clusters is necessary and there are possible connectivity constraints to form clusters.

TABLE 5.51: Encrypted domain evaluation metric scores for dataset 500

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.788	0.732	0.759	0.475	0.693	0.295	18.986376
128	0.769	0.723	0.745	0.449	0.682	0.291	22.642983
256	0.786	0.742	0.764	0.482	0.704	0.296	40.160459
512	0.768	0.724	0.745	0.448	0.683	0.289	145.019023
1024	0.796	0.748	0.771	0.493	0.711	0.312	853.861069

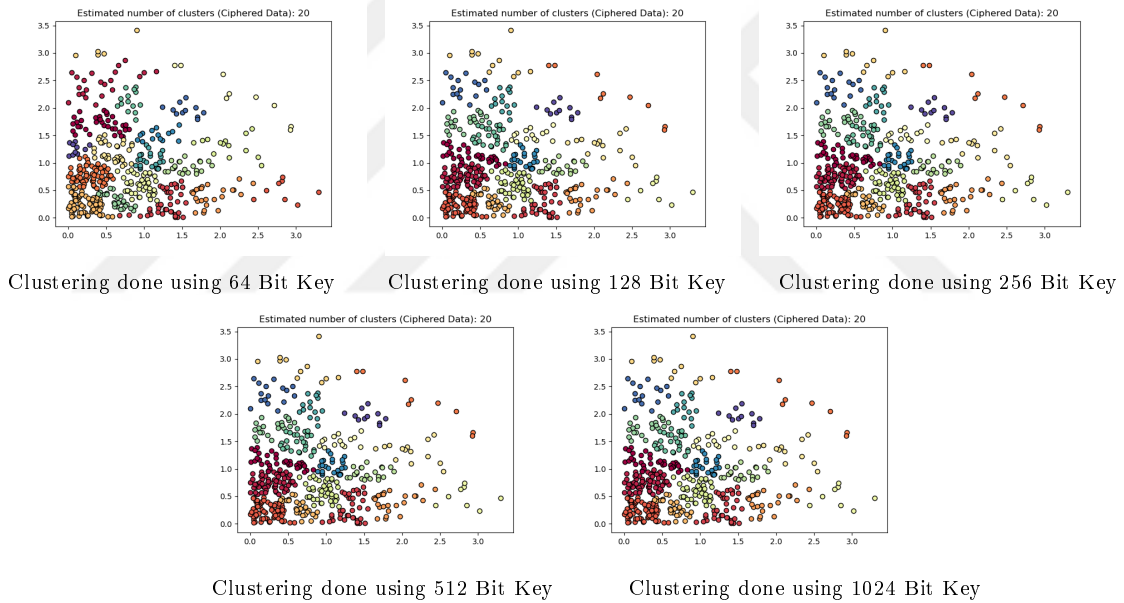


FIGURE 5.53: Encrypted domain clustering results for dataset 500

TABLE 5.52: Encrypted domain evaluation metric scores for dataset 1000

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.714	0.662	0.687	0.419	0.639	0.287	83.323777
128	0.705	0.656	0.680	0.422	0.632	0.272	97.465241
256	0.708	0.661	0.684	0.422	0.638	0.272	167.564546
512	0.739	0.679	0.708	0.453	0.657	0.296	579.170437
1024	0.727	0.671	0.698	0.429	0.649	0.277	3534.103834

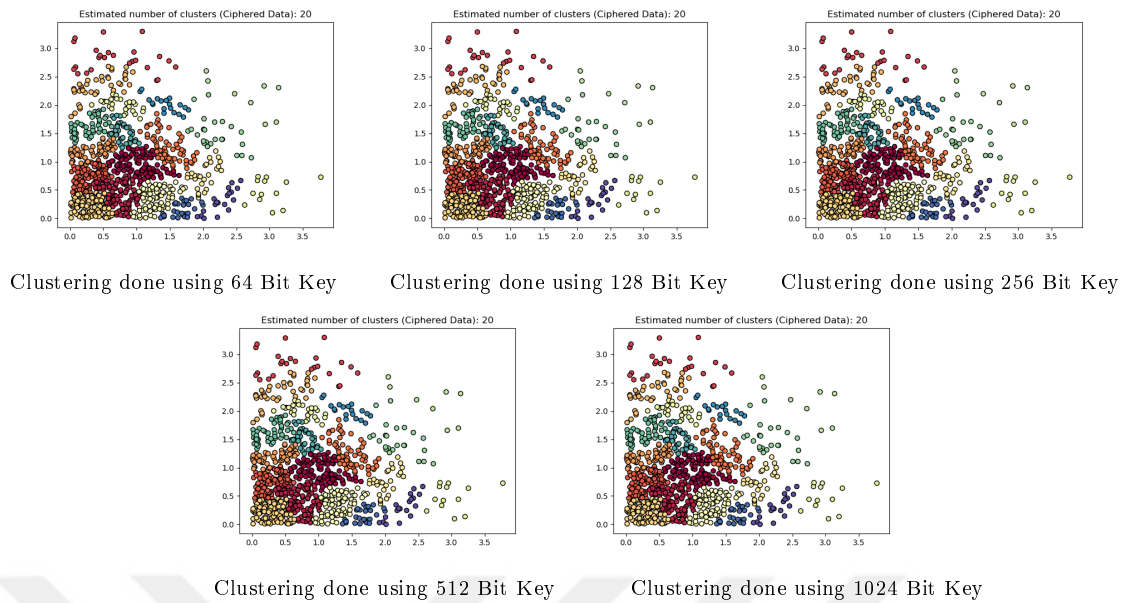


FIGURE 5.54: Encrypted domain clustering results for dataset 1000

TABLE 5.53: Encrypted domain evaluation metric scores for dataset 1500

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.706	0.661	0.683	0.431	0.646	0.276	196.051964
128	0.741	0.712	0.726	0.507	0.699	0.303	228.930361
256	0.723	0.689	0.705	0.468	0.675	0.265	390.931864
512	0.748	0.717	0.732	0.512	0.704	0.280	1317.826500
1024	0.706	0.731	0.719	0.496	0.693	0.286	10427.998650

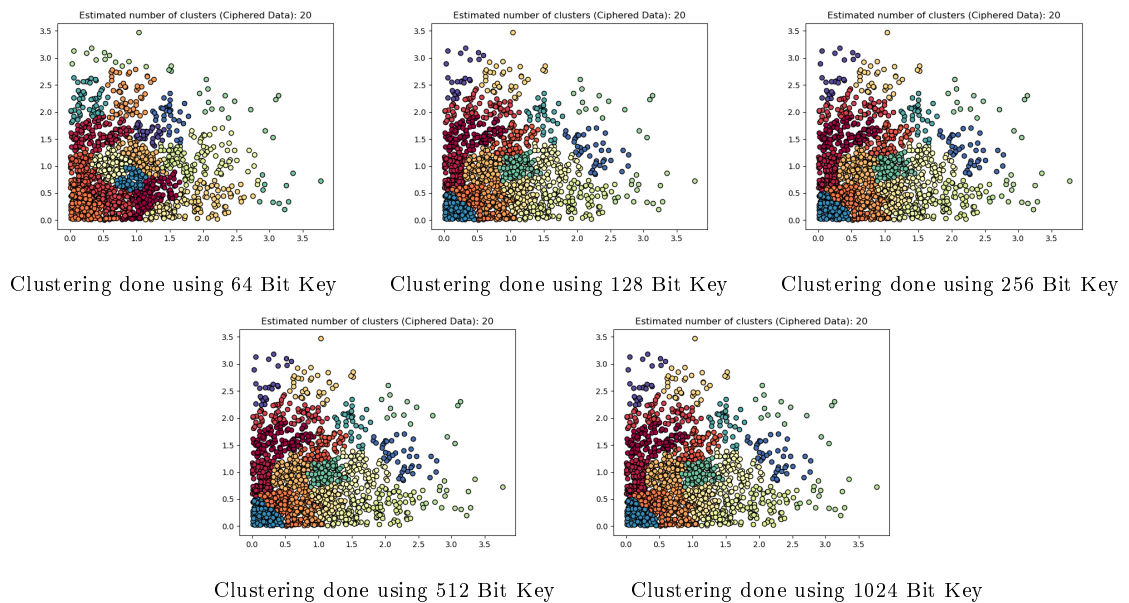


FIGURE 5.55: Encrypted domain clustering results for dataset 1500

TABLE 5.54: Encrypted domain evaluation metric scores for dataset 2000

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.680	0.661	0.670	0.390	0.650	0.285	367.038711
128	0.735	0.715	0.725	0.528	0.706	0.294	428.576449
256	0.722	0.706	0.714	0.504	0.696	0.306	725.576526
512	0.747	0.728	0.737	0.540	0.719	0.298	2329.031204
1024	0.706	0.717	0.712	0.490	0.696	0.299	14633.003870

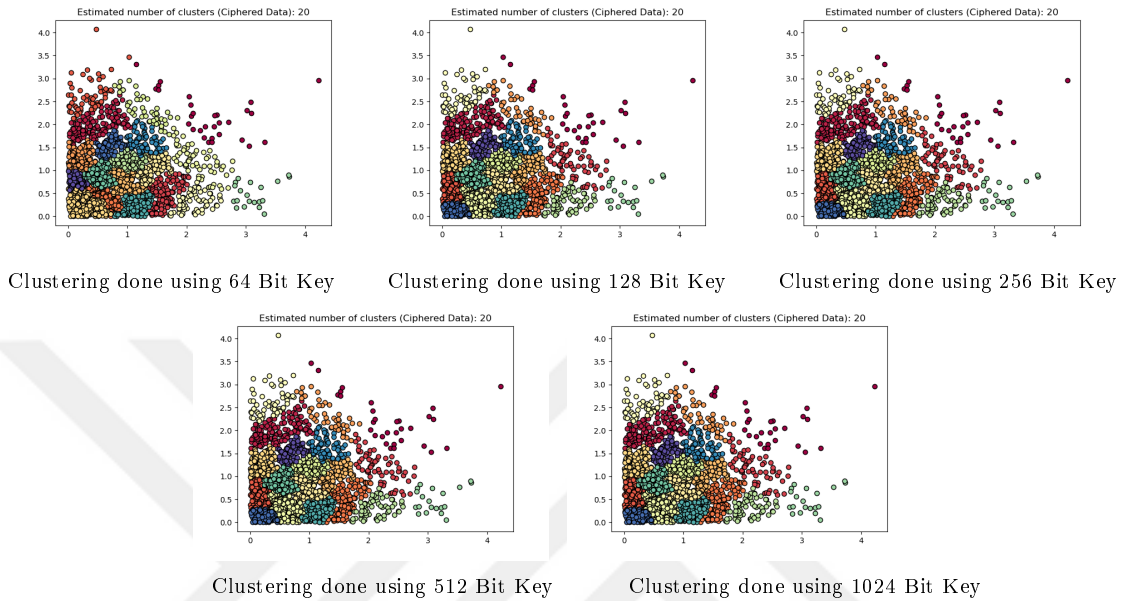


FIGURE 5.56: Encrypted domain clustering results for dataset 2000

TABLE 5.55: Encrypted domain evaluation metric scores for dataset 2500

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.686	0.648	0.667	0.389	0.639	0.305	613.736580
128	0.727	0.713	0.720	0.505	0.706	0.302	710.651168
256	0.716	0.692	0.704	0.470	0.685	0.288	1398.371611
512	0.711	0.699	0.705	0.479	0.691	0.301	4549.038970
1024	0.711	0.724	0.717	0.494	0.703	0.274	21740.579705

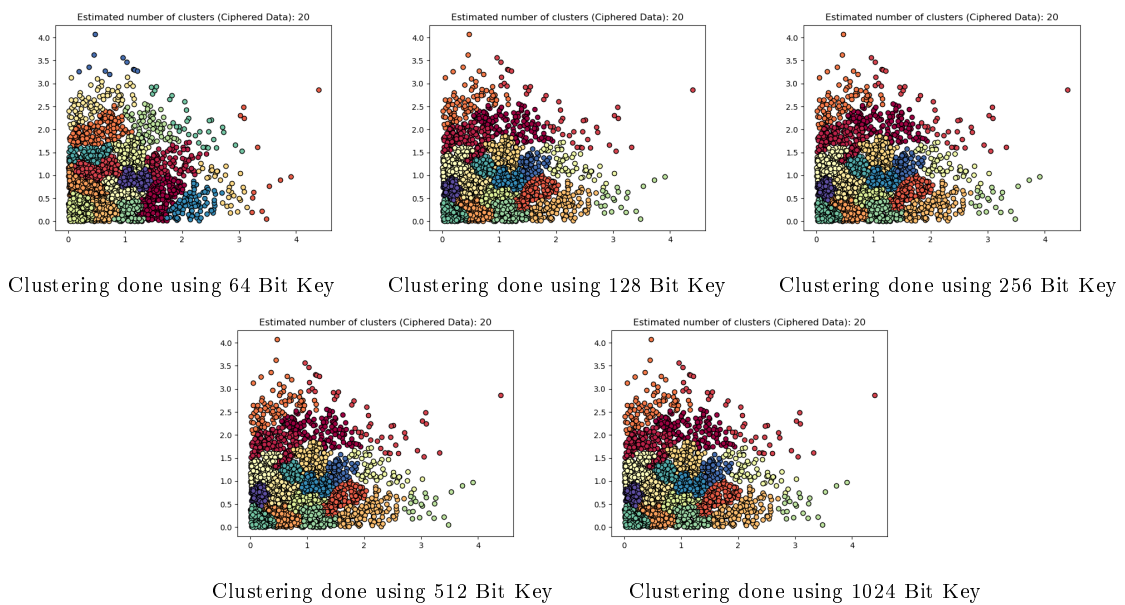


FIGURE 5.57: Encrypted domain clustering results for dataset 2500

TABLE 5.56: Encrypted domain evaluation metric scores for dataset 3000

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.695	0.664	0.679	0.442	0.657	0.297	1165.375747
128	0.703	0.675	0.689	0.452	0.668	0.301	1270.462384
256	0.703	0.675	0.689	0.428	0.669	0.295	2163.898116
512	0.699	0.670	0.684	0.454	0.663	0.292	6722.317208
1024	0.673	0.702	0.687	0.455	0.666	0.271	34352.033333

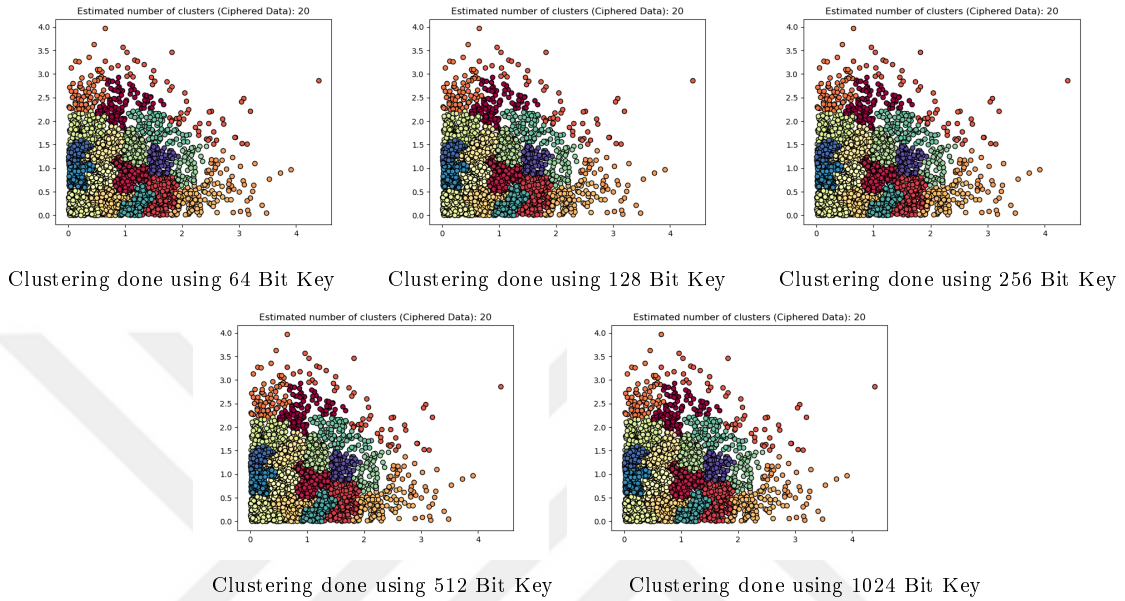


FIGURE 5.58: Encrypted domain clustering results for dataset 3000

TABLE 5.57: Encrypted domain evaluation metric scores for dataset 3500

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.715	0.682	0.698	0.476	0.676	0.298	1334.015664
128	0.712	0.690	0.701	0.474	0.684	0.300	1544.466207
256	0.713	0.688	0.700	0.462	0.682	0.296	2373.623768
512	0.717	0.689	0.703	0.470	0.683	0.307	7435.351474
1024	0.703	0.680	0.691	0.456	0.674	0.289	43162.416905

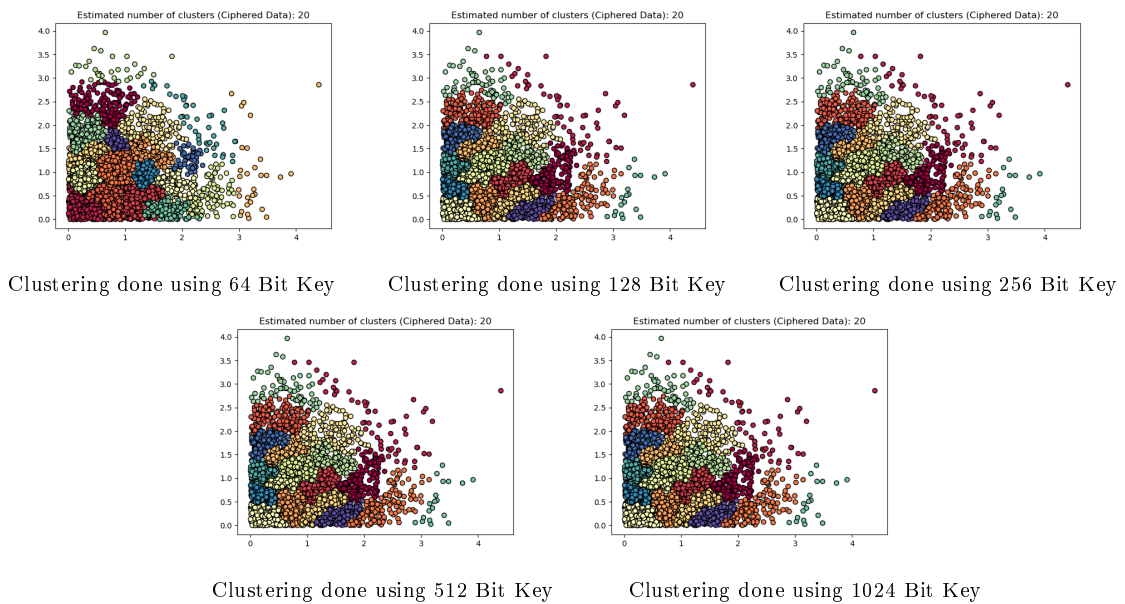


FIGURE 5.59: Encrypted domain clustering results for dataset 3500

TABLE 5.58: Encrypted domain evaluation metric scores for dataset 4000

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.713	0.684	0.698	0.454	0.679	0.296	1856.406278
128	0.644	0.688	0.665	0.392	0.638	0.264	2839.123717
256	0.641	0.688	0.664	0.384	0.636	0.264	3254.513282
512	0.659	0.703	0.680	0.407	0.654	0.264	9928.278455
1024	0.646	0.689	0.667	0.389	0.640	0.264	58259.589612

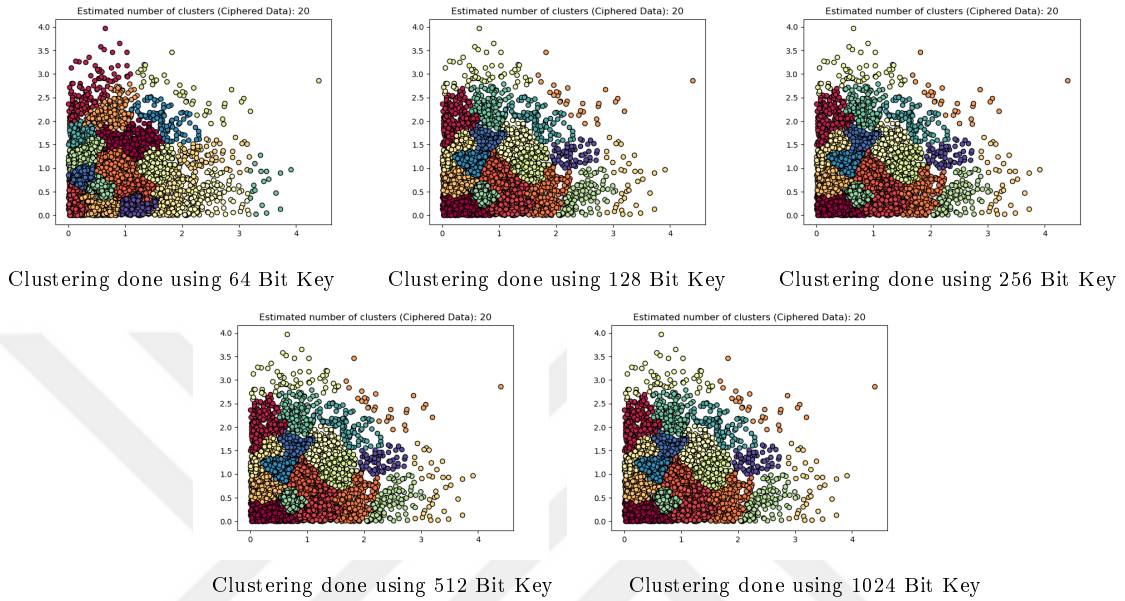


FIGURE 5.60: Encrypted domain clustering results for dataset 4000

TABLE 5.59: Encrypted domain evaluation metric scores for dataset 4500

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.615	0.659	0.636	0.348	0.609	0.253	4582.444243
128	0.691	0.710	0.700	0.486	0.686	0.274	2785.228748
256	0.674	0.696	0.684	0.442	0.669	0.274	5738.454688
512	0.684	0.708	0.696	0.461	0.680	0.274	12611.520444
1024	0.675	0.701	0.688	0.457	0.671	0.274	75562.476217

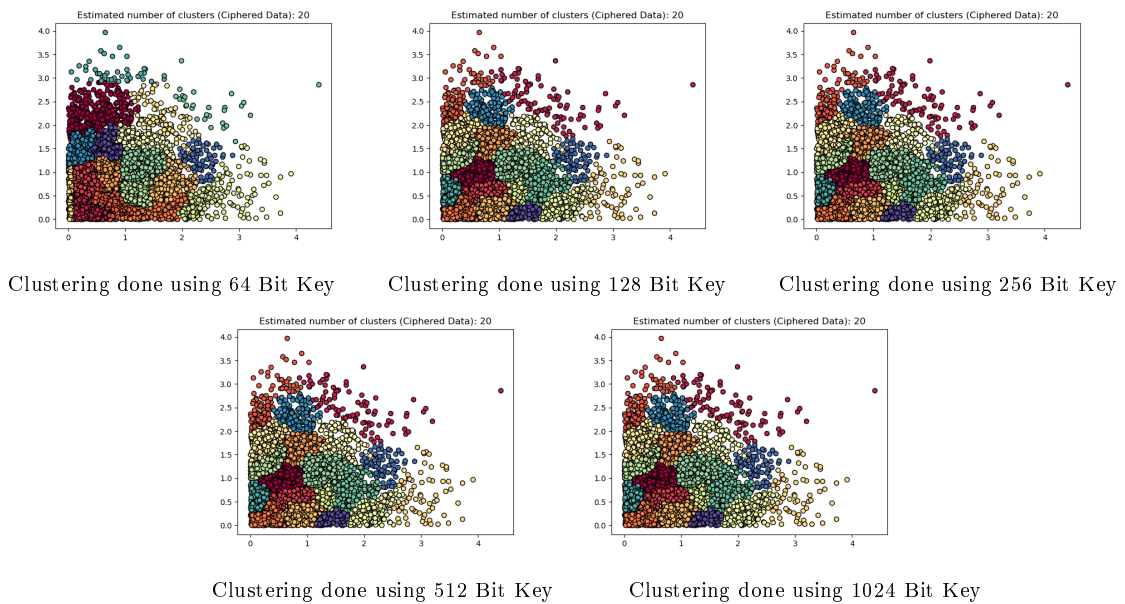


FIGURE 5.61: Encrypted domain clustering results for dataset 4500

TABLE 5.60: Encrypted domain evaluation metric scores for dataset 5000

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.627	0.665	0.645	0.361	0.622	0.250	4082.753650
128	0.639	0.673	0.655	0.390	0.635	0.245	4321.325598
256	0.654	0.686	0.670	0.405	0.649	0.245	5737.903436
512	0.634	0.659	0.646	0.360	0.629	0.245	15691.378081
1024	0.651	0.680	0.665	0.401	0.646	0.245	90506.861951

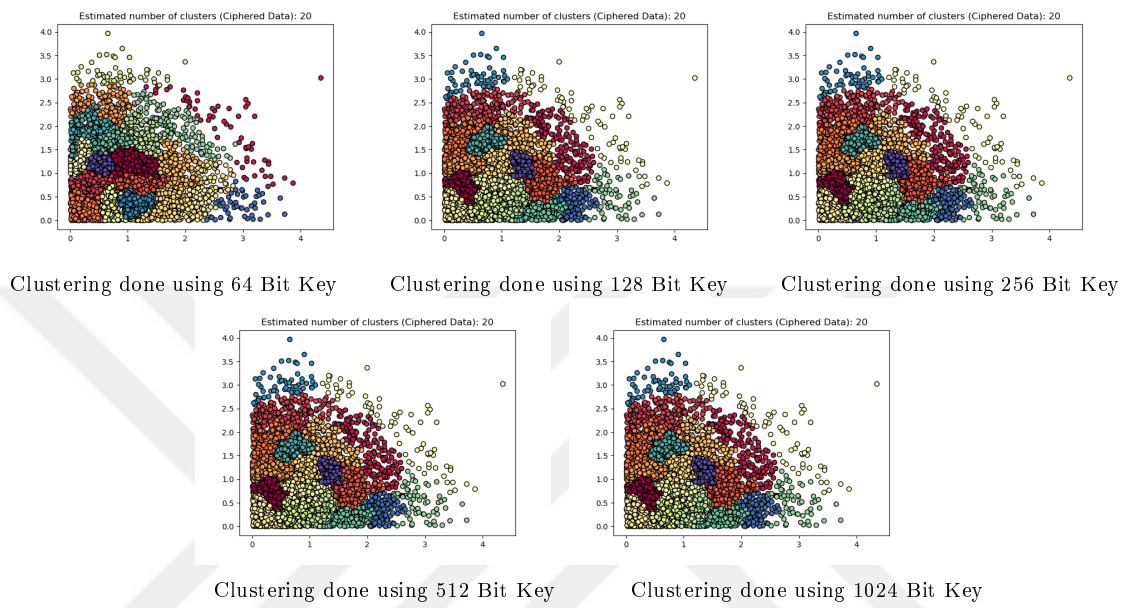


FIGURE 5.62: Encrypted domain clustering results for dataset 5000



FIGURE 5.63: Charts that show change of each evaluation metrics score by data length for different key lengths and for Hierarchical Algorithm

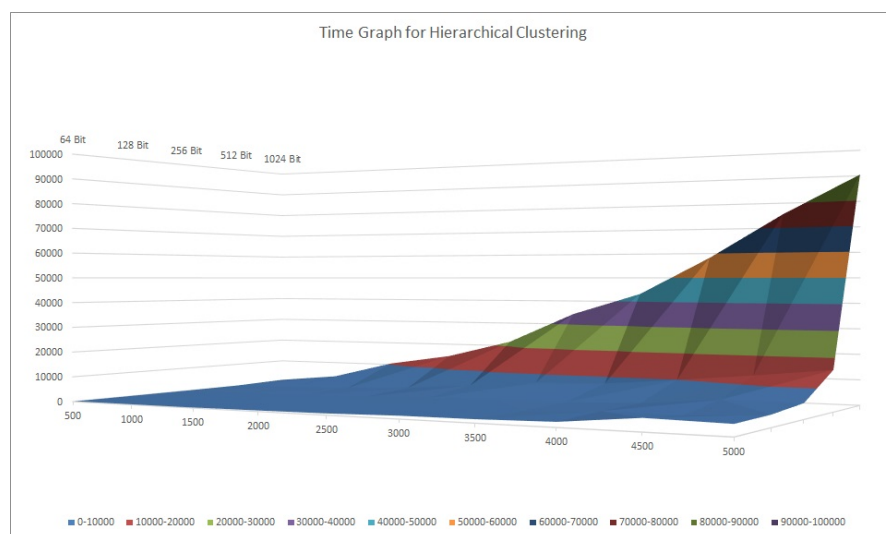


FIGURE 5.64: Calculation Time Graph for Hierarchical Algorithm on Key-Data Length Dimensions

5.2.3 Spectral Algorithm Results

From Table 5.61 to 5.70 and Figure 5.63 to 5.72 we will see the distribution of *encrypted* data which its length varies between 500 and 5000 due to Spectral algorithm. *Spectral* algorithm forms clusters considering *nearest – neighbors*. This algorithm is used generally when the data that needs to be clustered has non-flat geometry, creating many clusters is not necessary and created clusters are even sized.

TABLE 5.61: Encrypted domain clustering results with dataset 500

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.880	0.876	0.878	0.756	0.857	0.312	27.473035
128	0.874	0.871	0.872	0.714	0.852	0.301	32.294984
256	0.886	0.880	0.883	0.762	0.862	0.317	55.222464
512	0.897	0.892	0.894	0.793	0.876	0.310	148.256260
1024	0.874	0.868	0.871	0.730	0.849	0.318	1907.164346

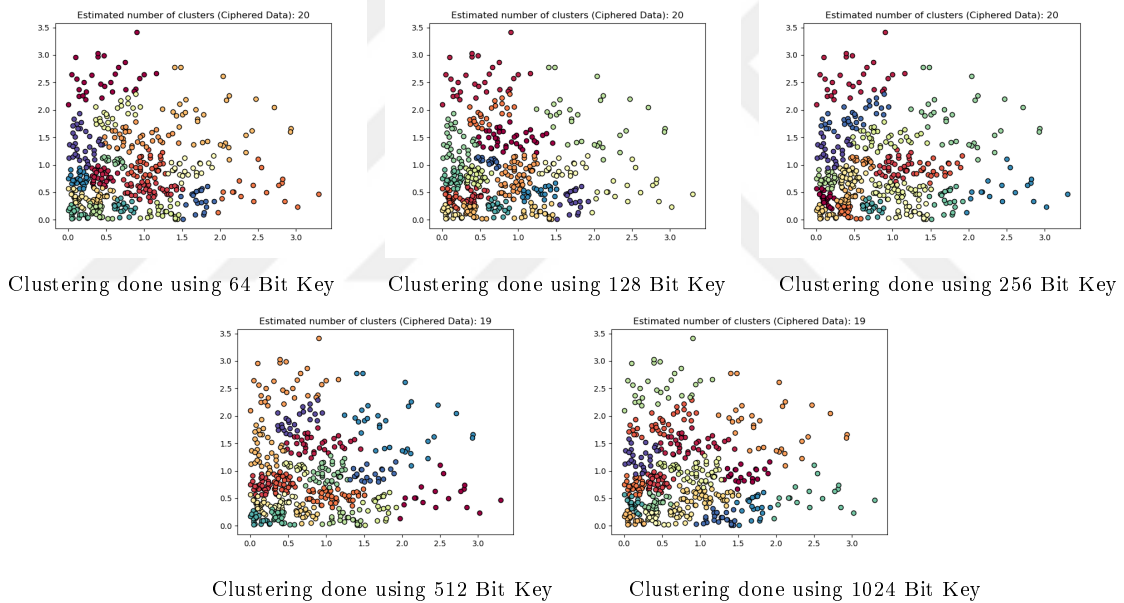


FIGURE 5.65: Encrypted domain results for dataset 500

TABLE 5.62: Encrypted domain clustering results with dataset 1000

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.843	0.858	0.851	0.713	0.833	0.262	97.614100
128	0.816	0.818	0.817	0.640	0.804	0.285	117.100256
256	0.842	0.848	0.845	0.694	0.832	0.291	219.221150
512	0.815	0.819	0.817	0.646	0.803	0.284	770.700724
1024	0.823	0.822	0.822	0.644	0.810	0.295	3458.167745

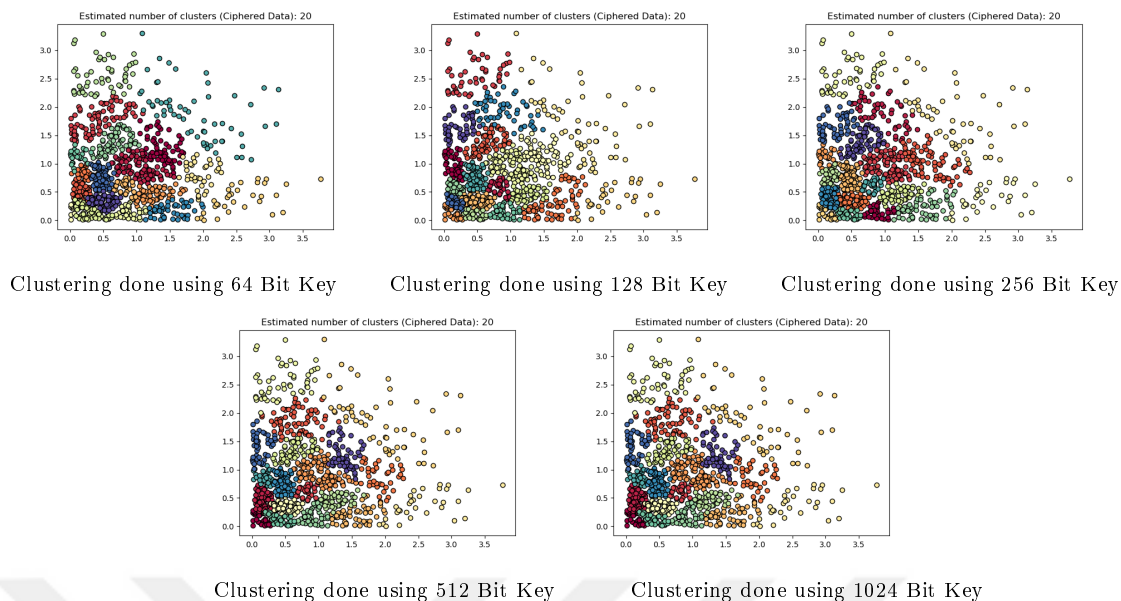


FIGURE 5.66: Encrypted domain results for dataset 1000

TABLE 5.63: Encrypted domain clustering results with dataset 1500

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.768	0.763	0.765	0.575	0.753	0.259	167.661065
128	0.806	0.803	0.805	0.641	0.795	0.275	204.724721
256	0.834	0.826	0.830	0.691	0.819	0.286	368.695161
512	0.811	0.811	0.811	0.674	0.803	0.269	1318.623197
1024	0.788	0.783	0.785	0.591	0.773	0.275	7844.362055

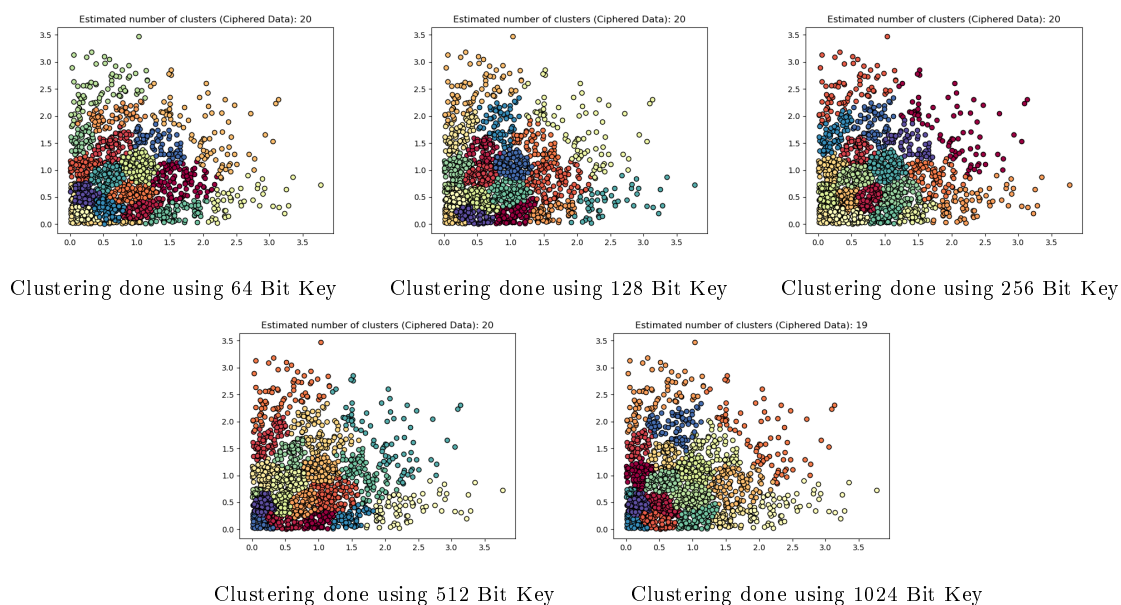


FIGURE 5.67: Encrypted domain results for dataset 1500

TABLE 5.64: Encrypted domain clustering results with dataset 2000

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.785	0.791	0.788	0.630	0.778	0.247	292.515220
128	0.820	0.827	0.823	0.668	0.814	0.271	358.194028
256	0.822	0.829	0.825	0.676	0.816	0.273	646.240033
512	0.833	0.840	0.836	0.707	0.828	0.270	2298.568044
1024	0.841	0.848	0.845	0.730	0.836	0.271	13878.686489

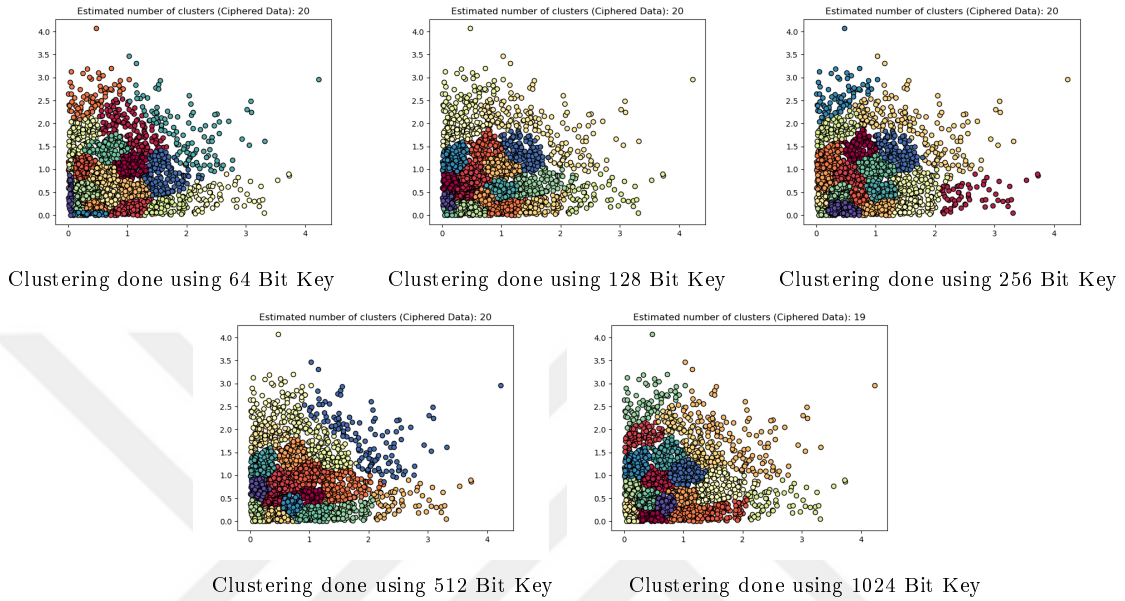


FIGURE 5.68: Encrypted domain results for dataset 2000

TABLE 5.65: Encrypted domain clustering results with dataset 2500

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.777	0.784	0.780	0.616	0.772	0.261	591.294237
128	0.816	0.820	0.818	0.685	0.812	0.289	714.742061
256	0.799	0.803	0.801	0.636	0.795	0.286	1289.402599
512	0.829	0.832	0.831	0.710	0.825	0.289	4808.001379
1024	0.806	0.806	0.806	0.665	0.801	0.291	23225.882568

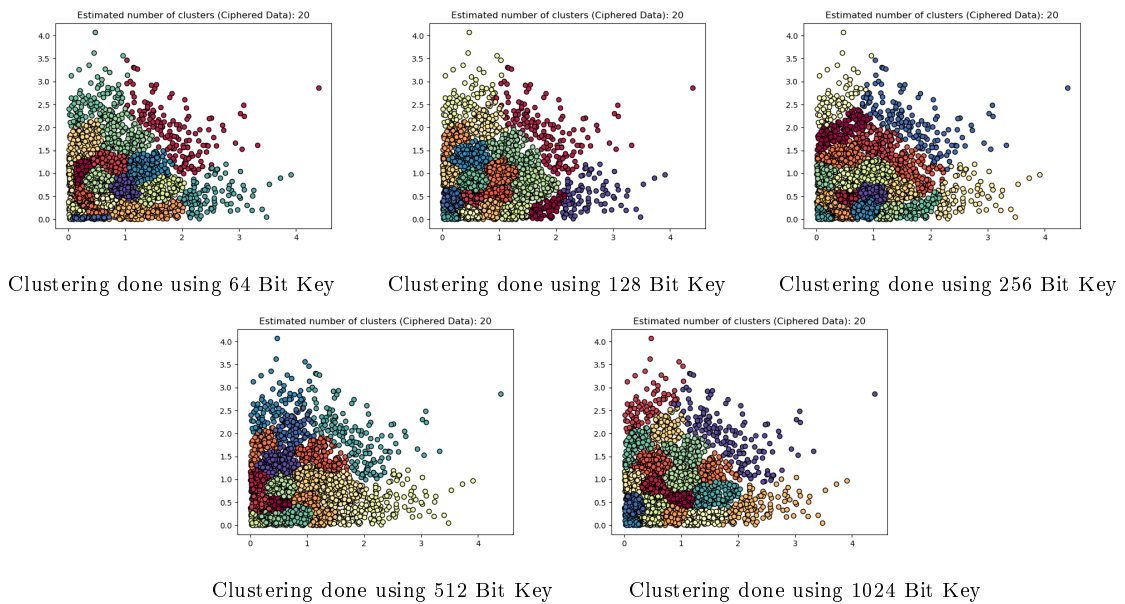


FIGURE 5.69: Encrypted domain results for dataset 2500

TABLE 5.66: Encrypted domain clustering results with dataset 3000

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.699	0.725	0.712	0.502	0.693	0.217	698.578957
128	0.792	0.800	0.796	0.622	0.788	0.273	807.522692
256	0.837	0.837	0.835	0.710	0.830	0.277	1427.103635
512	0.817	0.821	0.819	0.678	0.813	0.269	5333.425660
1024	0.814	0.820	0.817	0.688	0.810	0.266	31492.537981

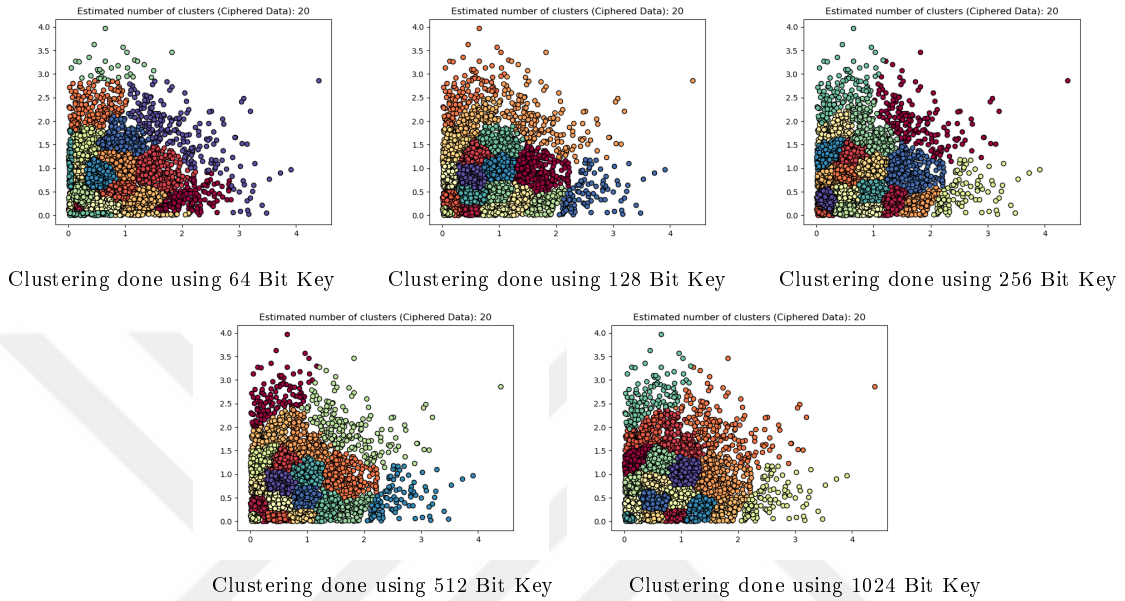


FIGURE 5.70: Encrypted domain results for dataset 3000

TABLE 5.67: Encrypted domain clustering results with dataset 3500

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.704	0.739	0.721	0.503	0.699	0.221	943.595023
128	0.755	0.763	0.759	0.553	0.751	0.270	1104.710917
256	0.760	0.767	0.764	0.563	0.756	0.270	1963.394550
512	0.795	0.798	0.797	0.633	0.792	0.270	6978.438284
1024	0.766	0.772	0.769	0.576	0.763	0.270	45285.358207

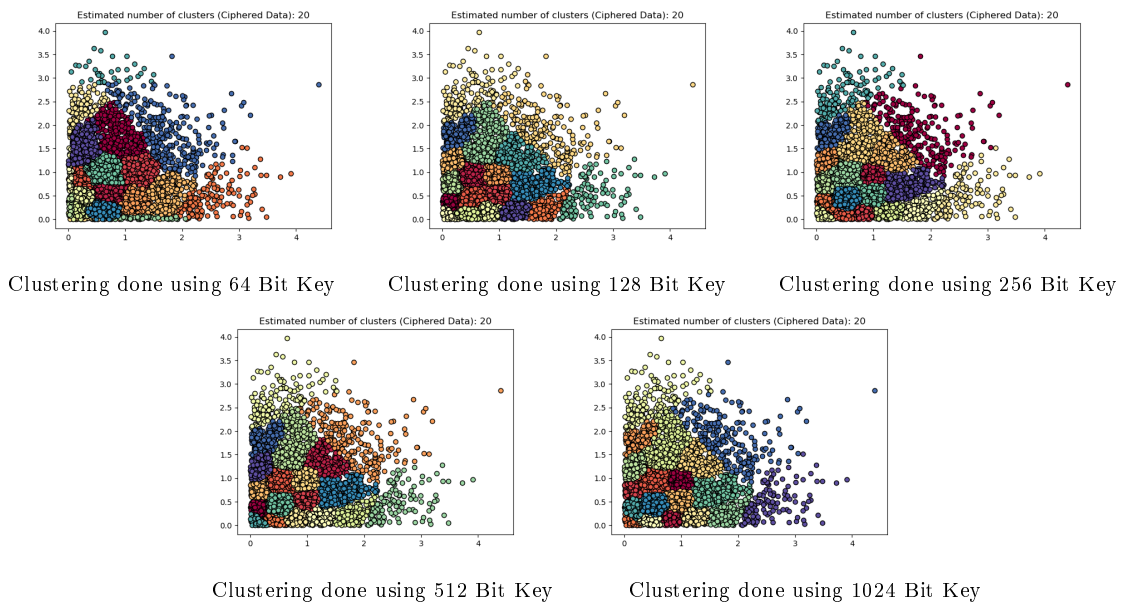


FIGURE 5.71: Encrypted domain results for dataset 3500

TABLE 5.68: Encrypted domain clustering results with dataset 4000

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.783	0.802	0.792	0.626	0.779	0.245	1192.235111
128	0.780	0.782	0.781	0.591	0.776	0.284	1464.420077
256	0.777	0.780	0.779	0.579	0.774	0.281	3012.274721
512	0.766	0.770	0.768	0.556	0.762	0.283	11371.570717
1024	0.777	0.780	0.778	0.576	0.773	0.283	56022.390935

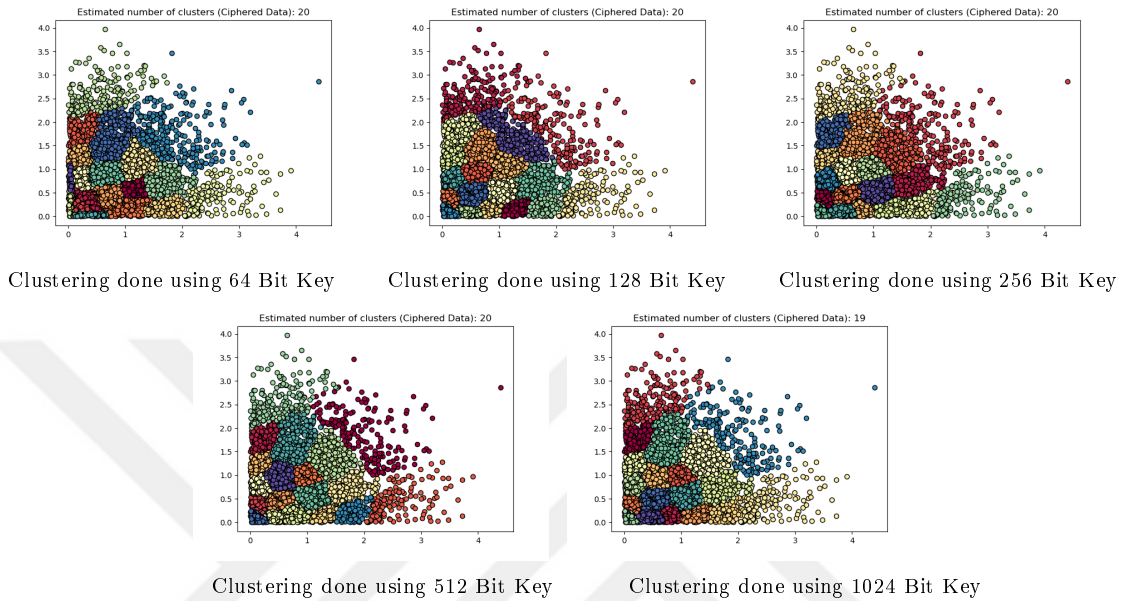


FIGURE 5.72: Encrypted domain results for dataset 4000

TABLE 5.69: Encrypted domain clustering results with dataset 4500

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.783	0.788	0.786	0.609	0.780	0.275	1470.769980
128	0.766	0.770	0.768	0.580	0.763	0.276	2437.502503
256	0.820	0.824	0.822	0.702	0.818	0.291	3236.176361
512	0.812	0.819	0.815	0.666	0.810	0.284	11463.184758
1024	0.792	0.794	0.793	0.632	0.789	0.289	81526.500912

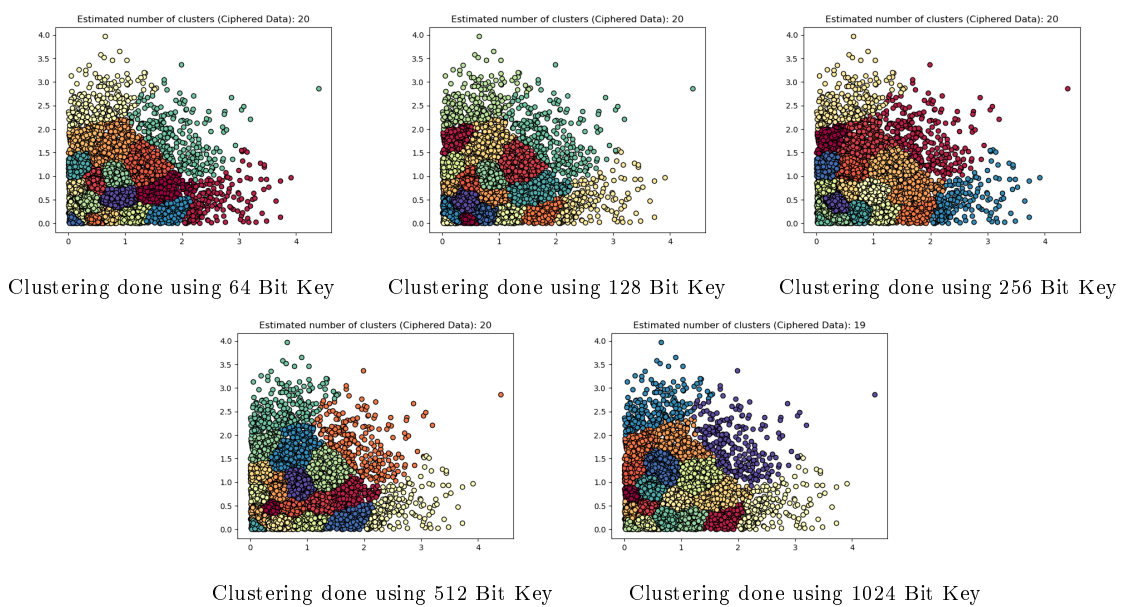


FIGURE 5.73: Encrypted domain results for dataset 4500

TABLE 5.70: Encrypted domain clustering results with dataset 5000

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.695	0.728	0.711	0.494	0.691	0.216	1816.478657
128	0.788	0.794	0.791	0.611	0.786	0.289	2205.573661
256	0.793	0.799	0.796	0.625	0.791	0.288	4031.744384
512	0.806	0.813	0.809	0.648	0.804	0.280	14343.408337
1024	0.792	0.798	0.795	0.621	0.790	0.288	96517.342535

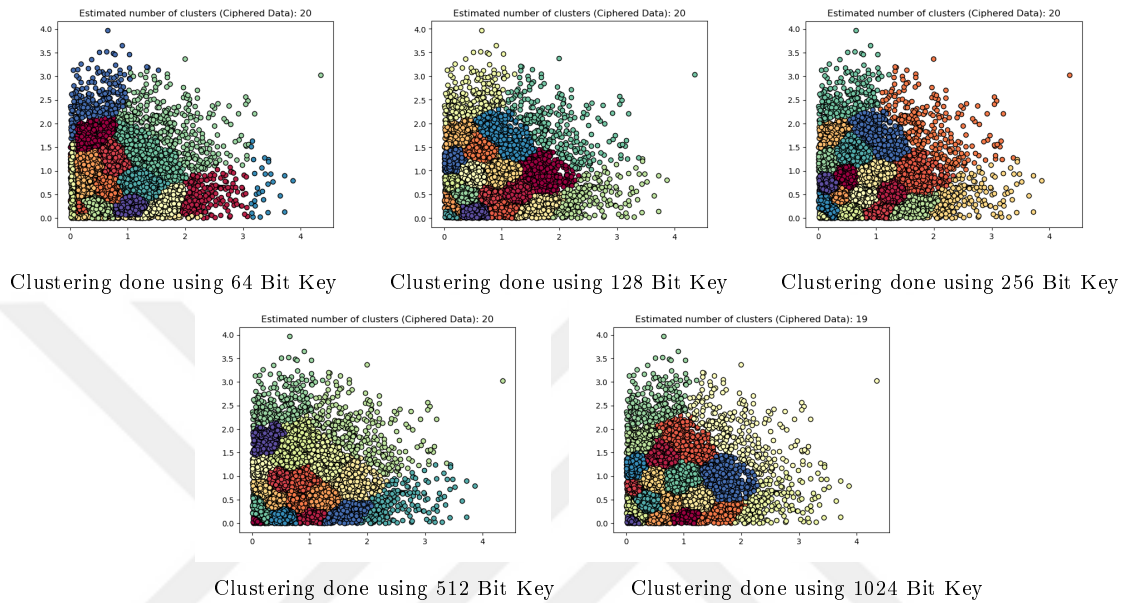


FIGURE 5.74: Encrypted domain results for dataset 5000



FIGURE 5.75: Charts that show change of each evaluation metrics score by data length for different key lengths and for Spectral Algorithm

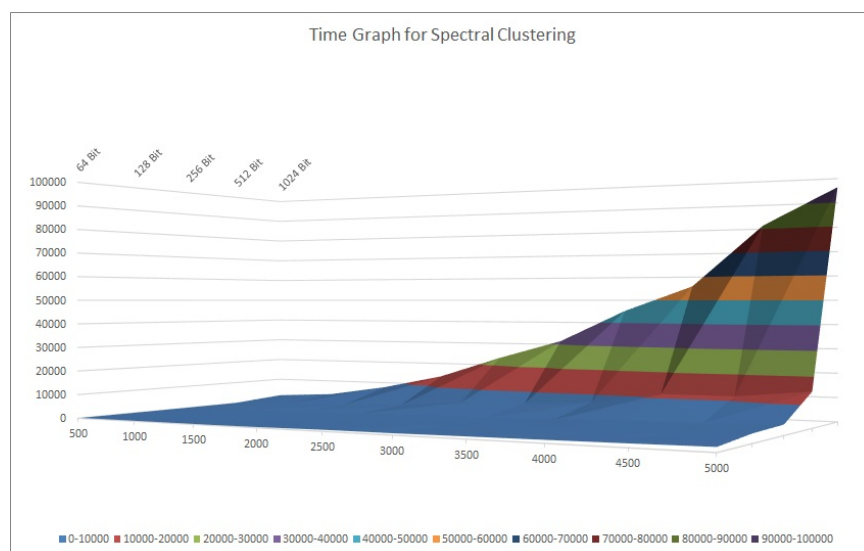


FIGURE 5.76: Calculation Time Graph for Spectral Algorithm on Key-Data Length Dimensions

5.2.4 Birch Algorithm Results

From Table 5.71 to 5.80 and Figure 5.74 to 5.83 we will see the distribution of *encrypted* data which its length varies between 500 and 5000 due to Spectral algorithm. *Birch* algorithm forms clusters considering *Euclidean distance* between points. This algorithm is used generally when the data that needs to be clustered is large and data reduction in respect to outlier removal is needed.

TABLE 5.71: Encrypted domain evaluation metric scores for dataset 500

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.754	0.686	0.718	0.474	0.647	0.325	19.439627
128	0.754	0.686	0.718	0.474	0.647	0.325	23.486909
256	0.754	0.686	0.718	0.474	0.647	0.325	41.035640
512	0.754	0.686	0.718	0.474	0.647	0.325	143.734250
1024	0.754	0.686	0.718	0.474	0.647	0.325	1048.355556

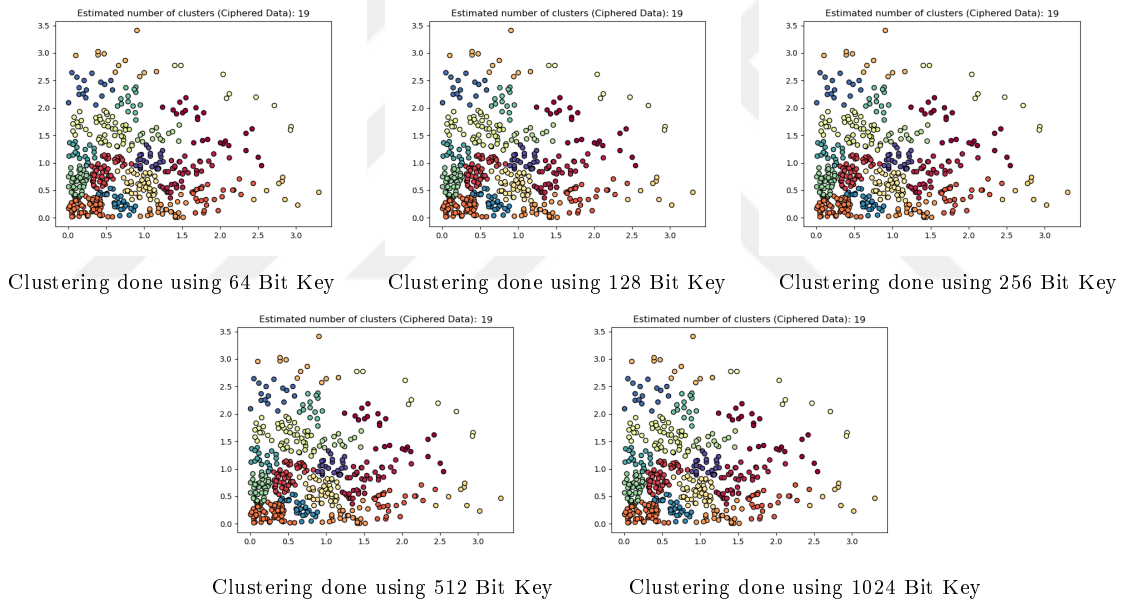


FIGURE 5.77: Encrypted domain clustering results for dataset 500

TABLE 5.72: Encrypted domain evaluation metric scores for dataset 1000

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.735	0.610	0.667	0.427	0.588	0.292	83.829280
128	0.735	0.610	0.667	0.427	0.588	0.292	128.729702
256	0.735	0.610	0.667	0.427	0.588	0.292	175.842091
512	0.735	0.610	0.667	0.427	0.588	0.292	580.245359
1024	0.735	0.610	0.667	0.427	0.588	0.292	3977.079074

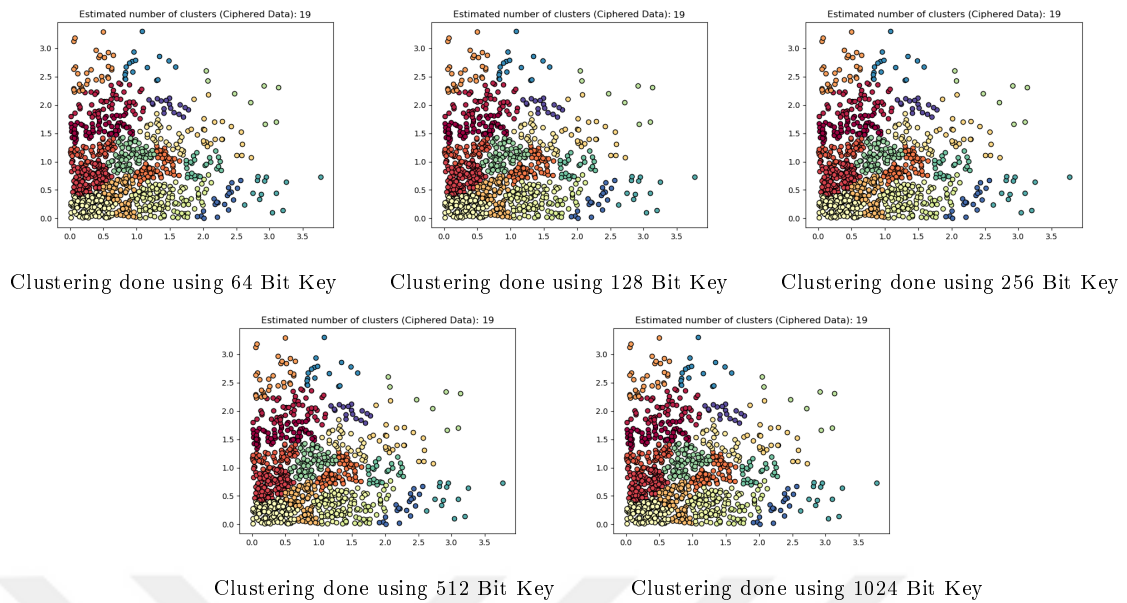


FIGURE 5.78: Encrypted domain clustering results for dataset 1000

TABLE 5.73: Encrypted domain evaluation metric scores for dataset 1500

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut.	Inf	Silh. Cff.	Time (s)
64	0.698	0.595	0.642	0.367	0.580	0.271	211.437032	
128	0.698	0.595	0.642	0.367	0.580	0.271	238.995674	
256	0.698	0.595	0.642	0.367	0.580	0.271	424.972714	
512	0.698	0.595	0.642	0.367	0.580	0.271	2658.128383	
1024	0.698	0.595	0.642	0.367	0.580	0.271	8180.718817	

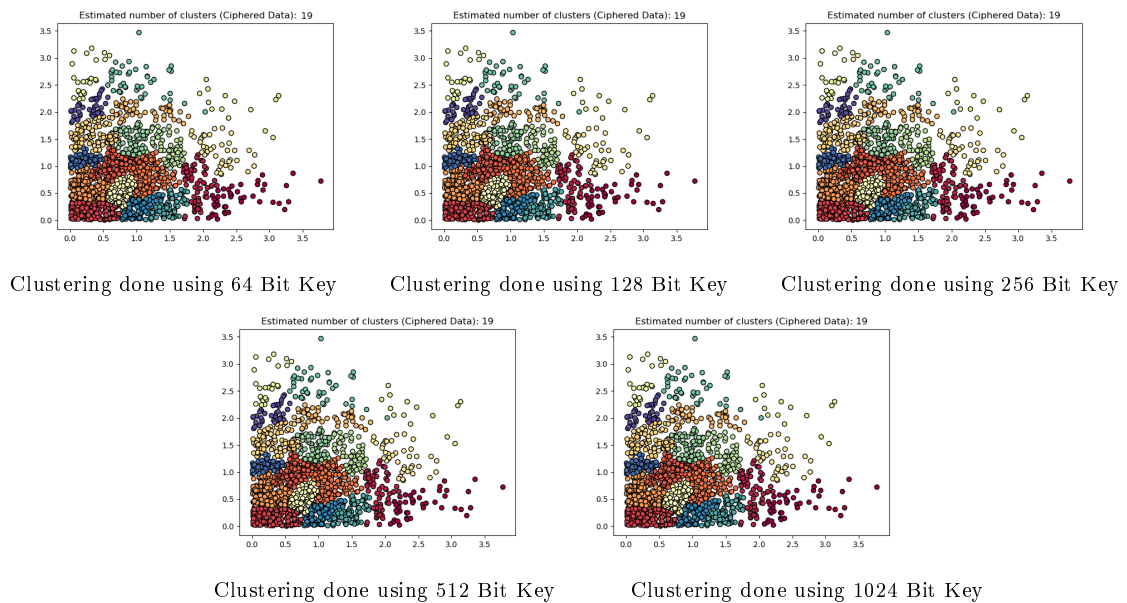


FIGURE 5.79: Encrypted domain clustering results for dataset 1500

TABLE 5.74: Encrypted domain evaluation metric scores for dataset 2000

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.706	0.569	0.630	0.364	0.557	0.282	372.510162
128	0.706	0.569	0.630	0.364	0.557	0.282	547.822752
256	0.706	0.569	0.630	0.364	0.557	0.282	1275.947787
512	0.706	0.569	0.630	0.364	0.557	0.282	4679.906635
1024	0.706	0.569	0.630	0.364	0.557	0.282	14160.692468

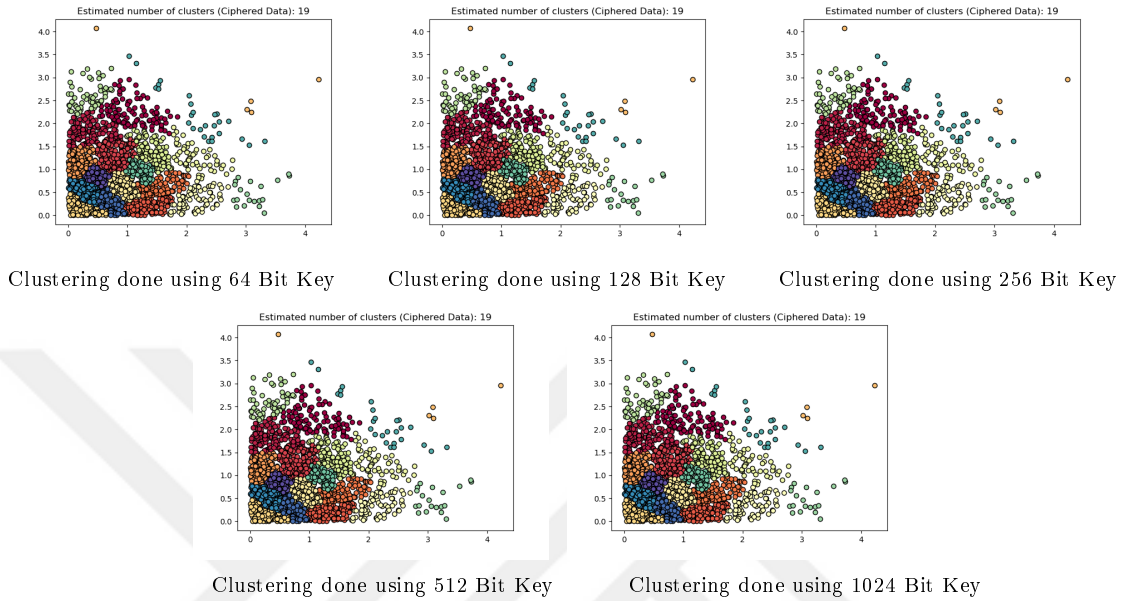


FIGURE 5.80: Encrypted domain clustering results for dataset 2000

TABLE 5.75: Encrypted domain evaluation metric scores for dataset 2500

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.723	0.562	0.632	0.363	0.553	0.297	663.725682
128	0.723	0.562	0.632	0.363	0.553	0.297	876.060204
256	0.723	0.562	0.632	0.363	0.553	0.297	1158.569366
512	0.723	0.562	0.632	0.363	0.553	0.297	3739.838171
1024	0.723	0.562	0.632	0.363	0.553	0.297	23466.670509

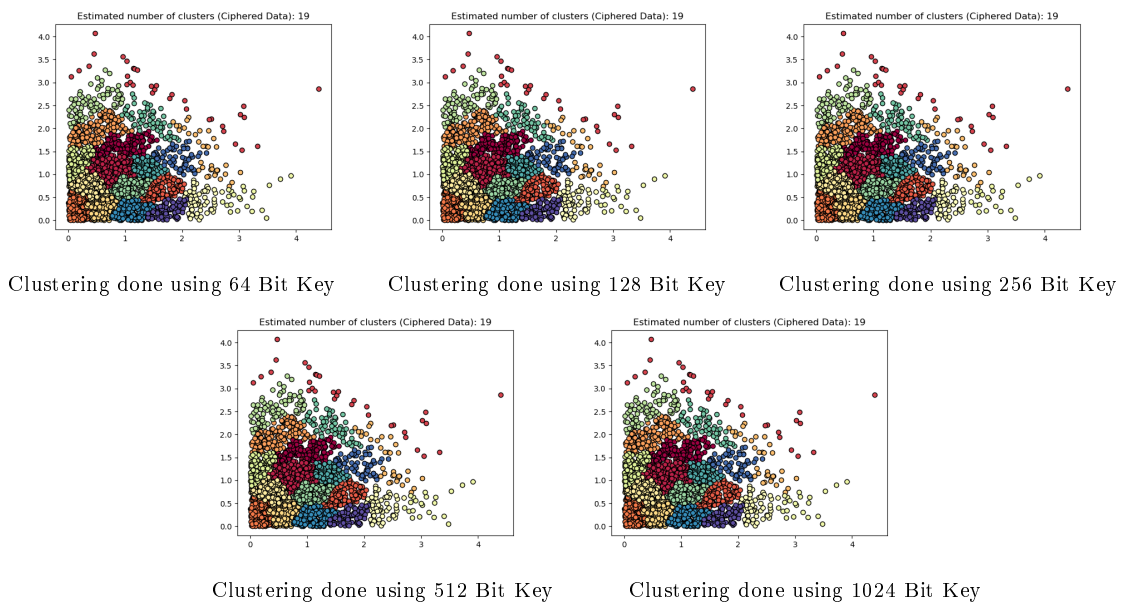


FIGURE 5.81: Encrypted domain clustering results for dataset 2500

TABLE 5.76: Encrypted domain evaluation metric scores for dataset 3000

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.735	0.543	0.625	0.353	0.535	0.272	1091.686250
128	0.719	0.520	0.603	0.296	0.511	0.267	1415.069280
256	0.719	0.520	0.603	0.296	0.511	0.267	3687.206187
512	0.719	0.520	0.603	0.296	0.511	0.267	5497.333181
1024	0.719	0.520	0.603	0.296	0.511	0.267	39601.765869

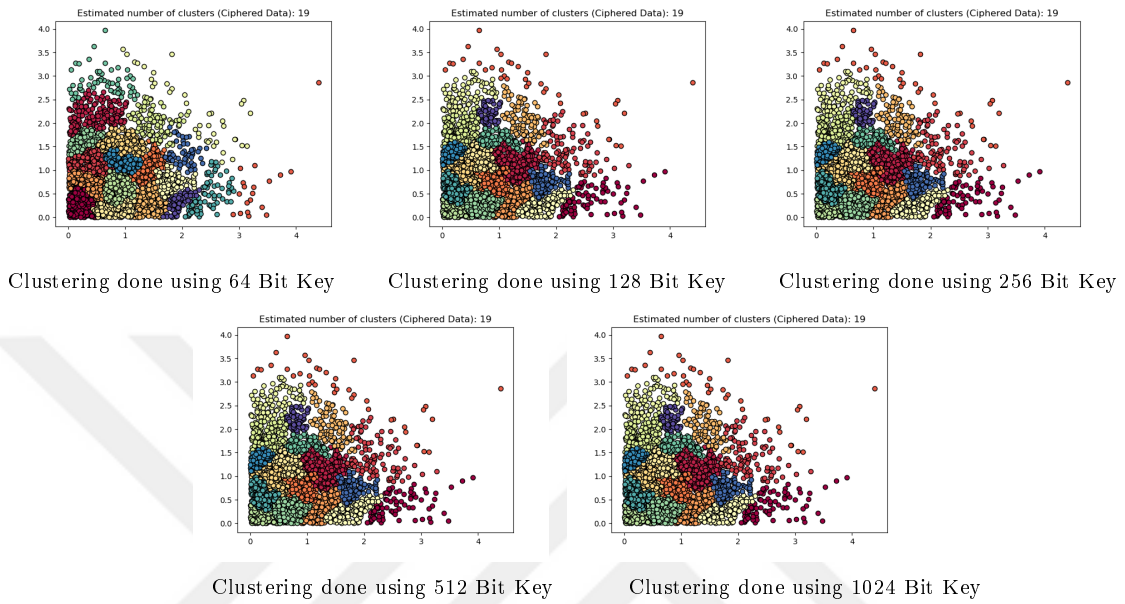


FIGURE 5.82: Encrypted domain clustering results for dataset 3000

TABLE 5.77: Encrypted domain evaluation metric scores for dataset 3500

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.722	0.547	0.622	0.334	0.540	0.260	1574.051378
128	0.722	0.547	0.622	0.334	0.540	0.260	1822.918705
256	0.722	0.547	0.622	0.334	0.540	0.260	2367.135811
512	0.722	0.547	0.622	0.334	0.540	0.260	10418.645935
1024	0.722	0.547	0.622	0.334	0.540	0.260	42655.888462

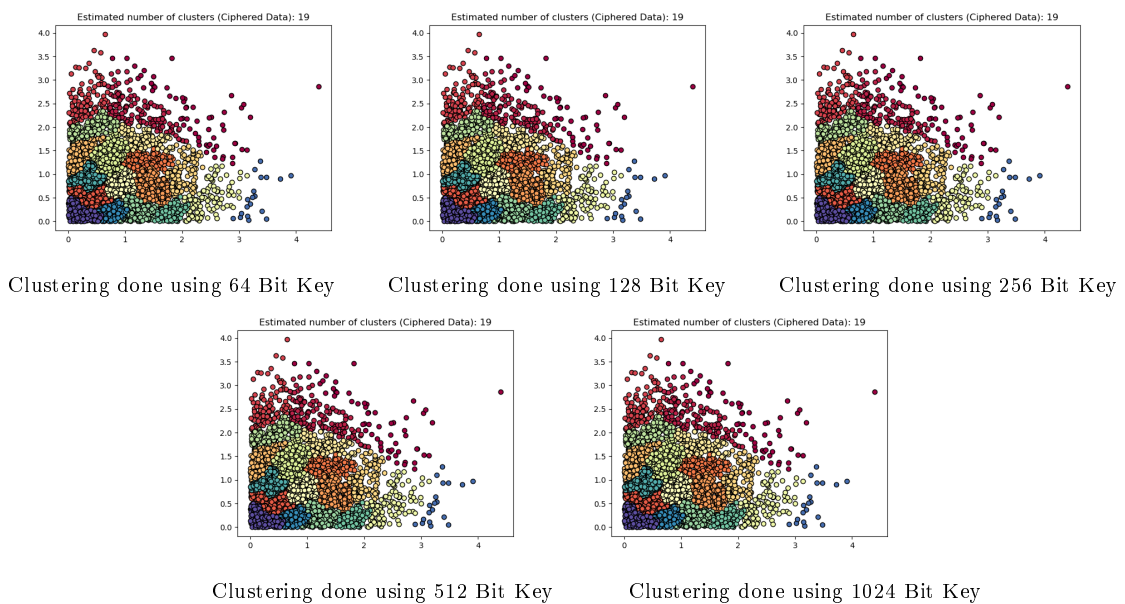


FIGURE 5.83: Encrypted domain clustering results for dataset 3500

TABLE 5.78: Encrypted domain evaluation metric scores for dataset 4000

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.716	0.539	0.615	0.381	0.532	0.273	1876.723837
128	0.721	0.533	0.613	0.363	0.527	0.271	2154.585296
256	0.721	0.533	0.613	0.363	0.527	0.271	3222.278868
512	0.721	0.533	0.613	0.363	0.527	0.271	9963.323332
1024	0.721	0.533	0.613	0.363	0.527	0.271	56957.615504

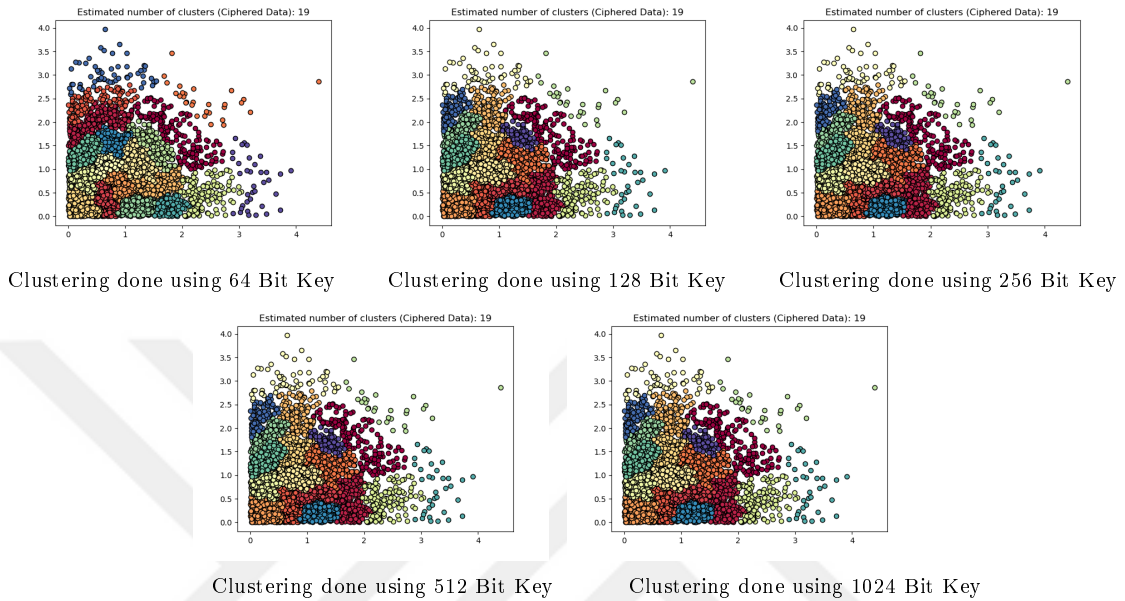


FIGURE 5.84: Encrypted domain clustering results for dataset 4000

TABLE 5.79: Encrypted domain evaluation metric scores for dataset 4500

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.712	0.559	0.626	0.391	0.553	0.251	2452.598178
128	0.680	0.529	0.595	0.297	0.523	0.256	2818.579733
256	0.680	0.529	0.595	0.297	0.523	0.256	4270.994490
512	0.680	0.529	0.595	0.297	0.523	0.256	13707.474558
1024	0.680	0.529	0.595	0.297	0.523	0.256	81720.847180

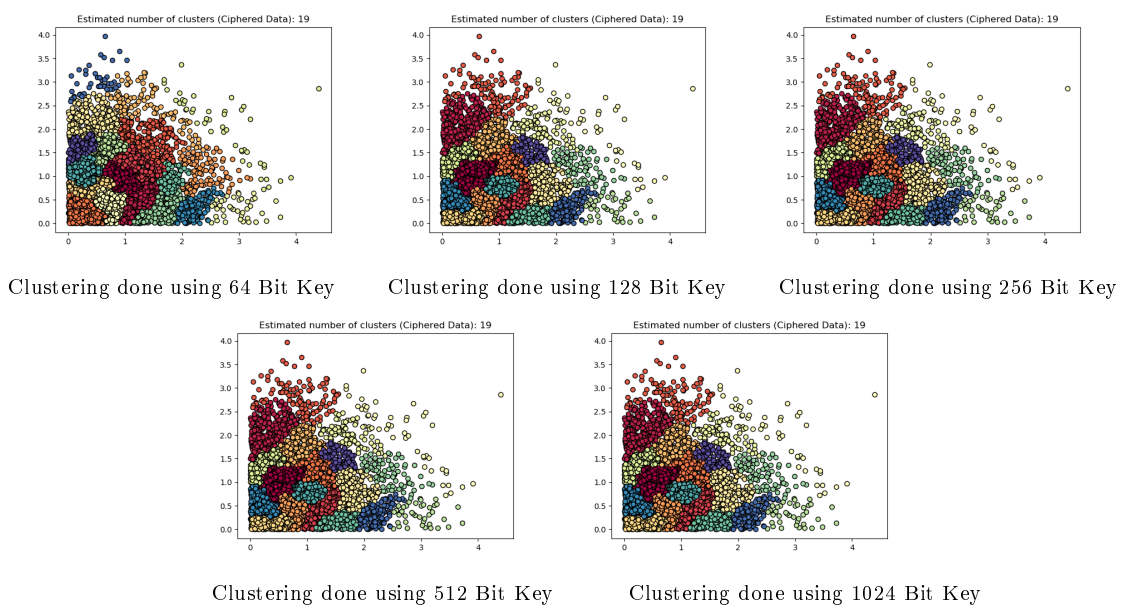


FIGURE 5.85: Encrypted domain clustering results for dataset 4500

TABLE 5.80: Encrypted domain evaluation metric scores for dataset 5000

Key	Hom.	Comp.	V-Measure	Adj.Rand.	Adj. Mut. Inf	Silh. Cff.	Time (s)
64	0.714	0.601	0.652	0.422	0.596	0.250	3156.384398
128	0.723	0.581	0.644	0.373	0.577	0.245	3498.406328
256	0.723	0.581	0.644	0.373	0.577	0.245	7191.104841
512	0.723	0.581	0.644	0.373	0.577	0.245	16862.456699
1024	0.723	0.581	0.644	0.373	0.577	0.245	90333.516413

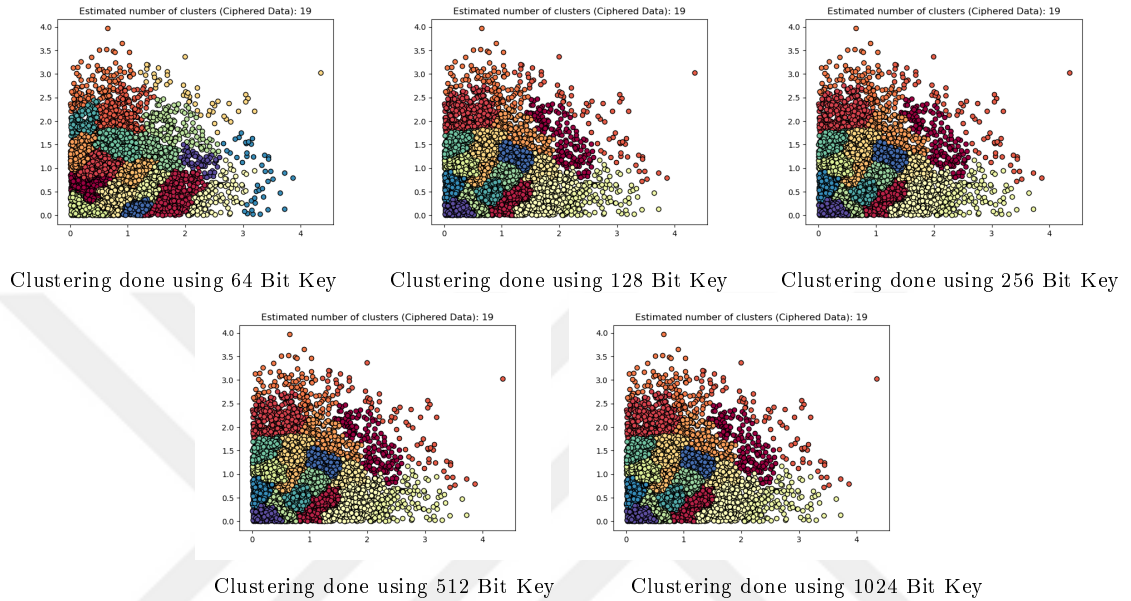


FIGURE 5.86: Encrypted domain clustering results for dataset 5000



FIGURE 5.87: Charts that show change of each evaluation metrics score by data length for different key lengths and for Birch Algorithm

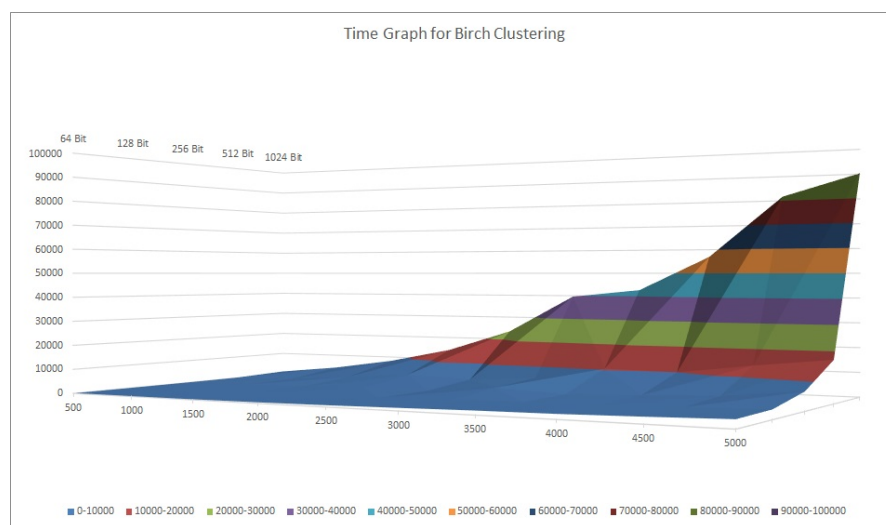


FIGURE 5.88: Calculation Time Graph of Birch Algorithm on Key-Data Length Dimensions

5.3 Results

Computations made on encrypted data are done in more time than plain data as expected. As the key size changes, computational time also changes proportionally in case of encrypted data computation, but in case of plain data computation time only changes by the change of data size.

Considering all six evaluation metrics and time, we offer usage suggestions for each clustering algorithm to run them in the most efficient way, while we define the efficiency in this work as better scores on evaluation metrics and stronger key usage.

For all algorithms that we examined, it is possible to suggest usage of 1024 bit key for all of them as evaluation metrics have high enough scores, but when it comes to computational time it is clearly not efficient enough. Using 1024 bit key takes almost six times longer computational time than using 512 bit key. As almost all evaluation scores are close to each other in case of using 1024 bit and 512 bit keys, computational time for using 512 bit key is clearly more manageable than using 1024 bit key.

Only Birch and Hierarchical algorithms need slightly less computational time than others while using 1024 bit key, so it is possible to suggest 1024 bit key for these algorithms, just in case of running these algorithms on a more powerful processor than we used in this work or, on the same processor but in the way of parallel computation.

Tables that show evaluation scores of plain data computations indicates that working on the plain data would get perfect evaluation scores except silhouette coefficient. Although the silhouette coefficient doesn't get the perfect score, it is already so close that it can be considered as 0,5. In this work, this case is resulted because we used artificial data which produced on python environment by the function of "*make blobs*", instead of a *real* data. In terms of real data, evaluation metrics can't get perfect scores for all cases and probability of getting perfect scores for real data is very low.

Chapter 6

Conclusions and Future Work

Data privacy holds an important place on on-line systems and different kinds of data brings more sensitivity that needs to be evaluated. As the need for handling sensitive data and increase of data sizes bring an important need to build efficient systems for both privacy preserving and efficiency on computation. In this work we examined four different clustering algorithms in terms of six different evaluation metrics and computational time. Each algorithm brings out relatively similar results, but in detail they have differences. Using our proposed model evaluation scores of clustering models of plain domain and encrypted domains are quite similar which means that conversion from floating point numbers to integer creates negligible differences between each clustering algorithm.

In this work plain data is artificial data, not real. So the data has already been produced due to a mathematical logic. This situation caused the clustering of plain domain to get perfect scores. In real world with using real data, this case won't be occurred because the real data won't be a result of a certain mathematical logic but instead it will be random data. So studying with real data would give more similar results to evaluation scores of encrypted domain. Therefore if the real data and its encrypted forms can be used in a future study, observation results for clustering algorithms will be more accurate.

Our proposed model preserves privacy for only training phase of clustering algorithms using encrypted and distributed datasets. The client builds final clustering model with aggregation of each encrypted distance matrix calculated at each party. As a result, the final model is in plain domain and some information such as cluster centroids are plain. If the client wants to share the model then some information leakage may occur.

As a future work, in order to prevent data leakage from the model, also the model should be encrypted using homomorphic encryption algorithms. In order to encrypt the model, the cluster centroids should be encrypted. The system that we offer with this work, doesn't possess the ability to use encrypted cluster centroids while creating models

from clustered data. To allow the client to use the encrypted clustering model, a new system model must be developed.

Based on the properties of the data, client side may need extra information about the clustered data to explain it accurately and describe the model they create. So as an another future work, gathering that extra information from data authority can be studied about. That information gathering should work without violating the data privacy, should eliminate the need of having a data scientist at data authority to create a metric (which will be used to discover contents of clustering results) and doing these without putting on extra processing load onto the client side shall be studied.



Bibliography

- [1] R. Bryant, R. H. Katz, and E. D. Lazowska. Big-data computing: creating revolutionary breakthroughs in commerce, science and society, 2008.
- [2] N. M. Nasrabadi. Pattern recognition and machine learning. *Journal of electronic imaging*, 16(4):049901, 2007.
- [3] G. J. Simmons. Symmetric and asymmetric encryption. *ACM Computing Surveys (CSUR)*, 11(4):305–330, 1979.
- [4] R. M. Redlich and M. A. Nemzow. Data security system and method, September 5 2006. US Patent 7,103,915.
- [5] R. R. Kacker, G. Appenzeller, M. J. Pauker, and T. Spies. Identity-based encryption system for secure data distribution, February 21 2006. US Patent 7,003,117.
- [6] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *ACM Sigmod Record*, 33(1):50–57, 2004.
- [7] O. Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, pages 86–97, 1998.
- [8] R. Cramer, I. Damgård, and U. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 316–334. Springer, 2000.
- [9] W. Du and M. J. Atallah. Secure multi-party computation problems and their applications: a review and open problems. In *Proceedings of the 2001 workshop on New security paradigms*, pages 13–22. ACM, 2001.
- [10] Y. Lindell and B. Pinkas. Privacy preserving data mining. In Mihir Bellare, editor, *Advances in Cryptology — CRYPTO 2000*, pages 36–54, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-540-44598-2.

- [11] K. Chaudhuri and C. Monteleoni. Privacy-preserving logistic regression. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 289–296. Curran Associates, Inc., 2009. URL <http://papers.nips.cc/paper/3486-privacy-preserving-logistic-regression.pdf>.
- [12] R. Agrawal and R. Srikant. Privacy-preserving data mining. *SIGMOD Rec.*, 29(2):439–450, May 2000. ISSN 0163-5808. doi: 10.1145/335191.335438. URL <http://doi.acm.org/10.1145/335191.335438>.
- [13] K. Xu, H. Yue, L. Guo, Y. Guo, and Y. Fang. Privacy-preserving machine learning algorithms for big data systems. In *2015 IEEE 35th International Conference on Distributed Computing Systems*, pages 318–327, June 2015. doi: 10.1109/ICDCS.2015.40.
- [14] S. Merugu and J. Ghosh. Privacy-preserving distributed clustering using generative models. In *Third IEEE International Conference on Data Mining*, pages 211–218, Nov 2003. doi: 10.1109/ICDM.2003.1250922.
- [15] R. Shokri and V. Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security, CCS '15*, pages 1310–1321, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3832-5. doi: 10.1145/2810103.2813687. URL <http://doi.acm.org/10.1145/2810103.2813687>.
- [16] Z. Yang, S. Zhong, and R. N. Wright. *Privacy-Preserving Classification of Customer Data without Loss of Accuracy*, pages 92–102. 2005. doi: 10.1137/1.9781611972757.9.
- [17] F. Emekci, O.D. Sahin, D. Agrawal, and A. El Abbadi. Privacy preserving decision tree learning over multiple parties. *Data Knowledge Engineering*, 63(2):348 – 361, 2007. ISSN 0169-023X.
- [18] P. Li, J. Li, Z. Huang, T. Li, C.-Z. Gao, S.-M. Yiu, and K. Chen. Multi-key privacy-preserving deep learning in cloud computing. *Future Generation Computer Systems*, 74:76 – 85, 2017. ISSN 0167-739X. doi: <https://doi.org/10.1016/j.future.2017.02.006>. URL <http://www.sciencedirect.com/science/article/pii/S0167739X17302005>.
- [19] X. Yi and Y. Zhang. Privacy-preserving naive bayes classification on distributed data via semi-trusted mixers. *Information Systems*, 34(3):371 – 380, 2009. ISSN 0306-4379. doi: <https://doi.org/10.1016/j.is.2008.11.001>. URL <http://www.sciencedirect.com/science/article/pii/S0306437908000914>.

- [20] J. Secretan, M. Georgiopoulos, A. Koufakou, and K. Cardona. Aphid: An architecture for private, high-performance integrated data mining. *Future Generation Computer Systems*, 26(7):891 – 904, 2010. ISSN 0167-739X. doi: <https://doi.org/10.1016/j.future.2010.02.017>. URL <http://www.sciencedirect.com/science/article/pii/S0167739X10000336>.
- [21] A. L. N. Fred and A. K. Jain. Data clustering using evidence accumulation. In *Object recognition supported by user interaction for service robots*, volume 4, pages 276–280 vol.4, 2002. doi: 10.1109/ICPR.2002.1047450.
- [22] R. C. Tryon. *Cluster analysis: Correlation profile and orthometric (factor) analysis for the isolation of unities in mind and personality*. Edwards brother, Incorporated, lithoprinters and publishers, 1939.
- [23] K. Bailey. Numerical taxonomy and cluster analysis. *Typologies and Taxonomies*, 34:24, 1994.
- [24] C.-H. Wu, C.-S. Ouyang, L.-W. Chen, and L.-W. Lu. A new fuzzy clustering validity index with a median factor for centroid-based clustering. *IEEE Transactions on Fuzzy Systems*, 23(3):701–718, 2015.
- [25] Z. Sun, G. Fox, W. Gu, and Z. Li. A parallel clustering method combined information bottleneck theory and centroid-based clustering. *The Journal of Supercomputing*, 69(1):452–467, 2014.
- [26] X. Xu, M. Ester, H. P. Kriegel, and J. Sander. A distribution-based clustering algorithm for mining in large spatial databases. In *Proceedings 14th International Conference on Data Engineering*, pages 324–331, Feb 1998. doi: 10.1109/ICDE.1998.655795.
- [27] J. D. Banfield and A. E. Raftery. Model-based gaussian and non-gaussian clustering. *Biometrics*, 49(3):803–821, 1993. ISSN 0006341X, 15410420. URL <http://www.jstor.org/stable/2532201>.
- [28] F. Cao, M. Ester, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 328–339. SIAM, 2006.
- [29] Y. He, H. Tan, W. Luo, H. Mao, D. Ma, S. Feng, and J. Fan. Mr-dbscan: an efficient parallel density-based clustering algorithm using mapreduce. In *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*, pages 473–480. IEEE, 2011.

- [30] L. Duan, L. Xu, F. Guo, J. Lee, and B. Yan. A local-density based spatial clustering algorithm with noise. *Information systems*, 32(7):978–986, 2007.
- [31] W. H. Day and H. Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, 1(1):7–24, 1984.
- [32] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 407–416. ACM, 2000.
- [33] G. Karypis, E.-H. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.
- [34] J.B. Phipps. Dendrogram topology. *Systematic zoology*, 20(3):306–308, 1971.
- [35] Anil K. J. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651 – 666, 2010. ISSN 0167-8655. doi: <https://doi.org/10.1016/j.patrec.2009.09.011>. URL <http://www.sciencedirect.com/science/article/pii/S0167865509002323>. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR).
- [36] A. Likas, N. Vlassis, and J.J. Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.
- [37] S. Lele and J. T. Richtsmeier. Euclidean distance matrix analysis: A coordinate-free approach for comparing biological shapes using landmark data. *American Journal of Physical Anthropology*, 86(3):415–427, 11 1991. ISSN 1096-8644. doi: 10.1002/ajpa.1330860307. URL <http://doi.org/10.1002/ajpa.1330860307>.
- [38] K. Alsabti, S. Ranka, and V. Singh. An efficient k-means clustering algorithm. 1997.
- [39] M. J. Li, M. K. Ng, Y.-M. Cheung, and J. Z. Huang. Agglomerative fuzzy k-means clustering algorithm with selection of number of clusters. *IEEE transactions on knowledge and data engineering*, 20(11):1519–1534, 2008.
- [40] S. White and P. Smyth. A spectral clustering approach to finding communities in graphs. In *Proceedings of the 2005 SIAM international conference on data mining*, pages 274–285. SIAM, 2005.
- [41] A. Damle, V. Minden, and L. Ying. Robust and efficient multi-way spectral clustering. *arXiv preprint arXiv:1609.08251*, 2016.
- [42] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141–182, 1997.

- [43] A. Garg, A. Mangla, N. Gupta, and V. Bhatnagar. Pbirch: A scalable parallel clustering algorithm for incremental data. In *Database Engineering and Applications Symposium, 2006. IDEAS'06. 10th International*, pages 315–316. IEEE, 2006.
- [44] A. Rosenberg and J. Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007.
- [45] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1): 193–218, Dec 1985. ISSN 1432-1343. doi: 10.1007/BF01908075. URL <https://doi.org/10.1007/BF01908075>.
- [46] N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *J. Mach. Learn. Res.*, 11:2837–2854, December 2010. ISSN 1532-4435. URL <http://dl.acm.org/citation.cfm?id=1756006.1953024>.
- [47] P. 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 0377-0427. doi: [https://doi.org/10.1016/0377-0427\(87\)90125-7](https://doi.org/10.1016/0377-0427(87)90125-7). URL <http://www.sciencedirect.com/science/article/pii/0377042787901257>.
- [48] S. Samet and A. Miri. Privacy-preserving back-propagation and extreme learning machine algorithms. *Data Knowledge Engineering*, 79-80:40 – 61, 2012. ISSN 0169-023X. doi: <https://doi.org/10.1016/j.datak.2012.06.001>. URL <http://www.sciencedirect.com/science/article/pii/S0169023X12000602>.
- [49] D. Hrestak and S. Picek. Homomorphic encryption in the cloud. In *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1400–1404, May 2014. doi: 10.1109/MIPRO.2014.6859786.
- [50] I. Damgård, V. Pastro, N. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pages 643–662, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-32009-5.
- [51] F. Zhao, C. Li, and C. F. Liu. A cloud computing security solution based on fully homomorphic encryption. In *16th International Conference on Advanced Communication Technology*, pages 485–488, Feb 2014. doi: 10.1109/ICACT.2014.6779008.

- [52] L. Morris. Analysis of partially and fully homomorphic encryption. *Rochester Institute of Technology*, pages 1–5, 2013.
- [53] M. Naehrig, K. Lauter, and V. Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop, CCSW '11*, pages 113–124, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-1004-8. doi: 10.1145/2046660.2046682. URL <http://doi.acm.org/10.1145/2046660.2046682>.
- [54] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, pages 223–238, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. ISBN 978-3-540-48910-8.
- [55] Z. Min, G. Yang, and J. Shi. A privacy-preserving parallel and homomorphic encryption scheme. *Open Physics*, 15(1):135–142, 2017.
- [56] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélicier, and P. Zimmermann. Mpfr: A multiple-precision binary floating-point library with correct rounding. *ACM Trans. Math. Softw.*, 33(2), June 2007. ISSN 0098-3500. doi: 10.1145/1236463.1236468. URL <http://doi.acm.org/10.1145/1236463.1236468>.
- [57] M. A. Malcolm. Algorithms to reveal properties of floating-point arithmetic. *Commun. ACM*, 15(11):949–951, November 1972. ISSN 0001-0782. doi: 10.1145/355606.361870. URL <http://doi.acm.org/10.1145/355606.361870>.