元　　智　　大　　學

資 訊 工 程 學 系

碩　士　論　文

實現保有隱私持續計算之雙雲多私密分享架構

Dual-Cloud Multi-Secret Sharing Architecture for Privacy Preserving Persistent Computation

研　　究　　生：楊喆凱
指 導 教 授：張經略 博士
陳昱圻 博士

中 華 民 國 一 一 二 年 一 月

實現保有隱私持續計算之雙雲多私密分享架構

Dual-Cloud Multi-Secret Sharing Architecture for Privacy Preserving Persistent Computation

研　究　生：楊喆凱
指 導 教 授：張經略 博士
陳昱圻 博士

Student: Jhe-Kai Yang
Advisor: Dr. Ching-Lueh Chang
Dr. Yu-Chi Chen

元　智　大　學
資　訊　工　程　學　系
碩 士 論 文

A Thesis

Submitted to

Department of Computer Science and Engineering

Yuan Ze University

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

in

Computer Science and Engineering

January 2023

Chungli, Taiwan, Republic of China

中華民國一一二年一月

實現保有隱私持續計算之雙雲多私密分享架構

學生：楊喆凱　　　　　　　　　　　　　指導教授：張經略 博士
陳昱圻 博士

元智大學資訊工程學系

摘　　要

隨著人工智能的盛行，人們通過眾多感測器收集資料，並利用機器學習打造模型獲取智能服務。然而，隨著設備的增加，資料隱私和海量資料問題也隨之而來。儘管秘密共享可以提供隱私並對密文進行高效計算，但共享大量資料可能會對資源受限設備帶來相當大的存儲負擔。因此，我們的目標是減少參與者需要持有的共享，並且建構安全的計算協議，來形成具有多秘密共享的保有隱私資料運用。在本文中，我們研究了具有保有隱私的持續計算機制，避免參與者在保密計算期間需要存儲額外的資訊。我們使用掩碼將秘密轉化爲受保護資料，並且通過多秘密共享分別上傳到非共謀的雙雲伺服器。然後，我們可以通過揭露一部分受保護資料來在本地計算出運算結果的共享，其中眞正的秘密始終保持安全，因爲另一部分處於共享狀態。最終，我們將資料集外包到雲，並使用我們的方案建構了一個保有隱私多方 kNN 分類，來進行一些實驗證明我們工作的可行性和可用性。實驗結果表明，計算的正確性得到了證實，並且不存在安全性和效率之間的權衡。

關鍵字：保有隱私計算、雙雲、多秘密共享。

Dual-Cloud Multi-Secret Sharing Architecture for Privacy Preserving Persistent Computation

Student: Jhe-Kai Yang

Advisor: Dr. Ching-Lueh Chang
Dr. Yu-Chi Chen

Department of Computer Science and Engineering
Yuan Ze University

ABSTRACT

With the prevalence of artificial intelligence, people collect data through numerous sensors and use machine learning to create models for intelligent services. However, data privacy and massive data issues are born with the proliferation of devices. Although secret sharing can provide privacy and perform efficient computations on ciphertext, sharing large amounts of data may impose a considerable storage burden on resource-restricted devices. Therefore, our goal is to reduce the shares that participants need to hold and construct secure computation protocols that form privacy-preserving data utilization with multi-secret sharing. In this thesis, we study the mechanism of persistent computation with privacy protection, which avoids requirements for participants to store extra information during secure computation. We use masks to convert secrets into protected data and upload them separately to non-colluding dual-cloud servers through multi-secret sharing. Then, we can locally calculate the operation result shares by revealing a part of the protected data, where the true secret consistently remains secure as the other part stays in a shared state. Eventually, we outsource a dataset to the cloud and build a privacy-preserving multi-party kNN classification with our scheme, which conducts some experiments to demonstrate the feasibility and usability of our work. The experimental results indicate that the correctness of the calculation is confirmed, and there is no trade-off between security and efficiency.

Keywords: Privacy-preserving computation, Dual-cloud, Multi-secret sharing.

誌　　　　　謝

時光飛逝，這篇承載了我的碩士生涯的論文也終於順利完成。邁向下一個人生的篇章之前，我想對這段路上幫助我達到這個里程碑的所有人表達最眞誠的謝意。在這段求學旅途的完結篇，就讓我細數那些溫馨感人的點點滴滴做一個完美的收尾。

首先，我要特別感謝我的指導教授陳昱圻老師。在研究所期間，昱圻老師是影響我最深的一位貴人。老師熱情開朗的性格、認眞負責的工作態度，總是用充滿親和力的方式關心著學生。每次與老師討論問題，都能用平易近人的方式指導著我們專業知識。要不是有老師通過他豐富的經驗引領著我，我的研究恐怕難以盡善盡美。再者，承蒙張經略老師和左瑞麟老師在口試期間給予的提點，使本論文能更進一步破除盲點，在此致上最深的感激。

接著，謝謝密碼學與資訊安全實驗室的同門師兄、師姐、師弟、師妹。無論是陪我挑燈夜讀的冠鈞學長、細心的智柔學姊、爲我帶來許多靈感的柏嘉同學、一起打點日常工作的乃綺同學、交流新知的澤洋同學、聊天說地的章景學弟，各位都是一起奮鬥的好夥伴。每週的開會時間，除了能快速地吸收最新的研究，更重要的是能和大家聚在一起。我們彼此討論學習，友愛互助，相互勉勵，讓研究時光充滿歡樂的回憶，也讓實驗室彷彿成爲我的另一個歸宿。

最後，我要感謝家人的援助與支持。雖然爸爸已經與世長辭，但我知道您最擔心還不成熟的我，希望如今的我足以讓你安心並且了無牽掛。這幾年來，媽媽支撐起了整個家，也總是幫我打理好大小瑣事，讓我可以沒有後顧之憂的專心學習。兄弟姊妹、親朋好友，大家都互相關懷，相互幫助。因爲有你們的支持我才有源源不絕的前進動力，眞的非常謝謝大家。

感謝這一路上所有幫助過我的家人、老師、同學和朋友，因你們的存在才能造就今天的我。謹以此篇論文獻給人生道路上一同努力的我們。

楊喆凱 謹致

元智大學資訊工程學系

西元 2023 年 1 月

Table of Contents

Abstract in Chinese iii

Abstract in English iv

Acknowledgements v

Table of Contents vi

List of Tables viii

List of Figures ix

1 Introduction 1

1.1 Background 1

1.2 Contributions 3

1.3 Organization 3

2 Related Works 4

2.1 Secret Sharing 4

2.2 Extension for Secret Sharing 5

2.3 Secure Computation 5

3 Preliminaries 7

3.1 Multi-Secret Sharing 7

3.2 Beaver Triples 8

3.3 k-nearest Neighbor Algorithm 9

4 System Framework 10

4.1 System Model 10

4.2 Threat Model 12
4.3 Design Goals 13
5 Proposed Scheme 15
5.1 Distribution and Reconstruction 15
5.2 Scalar Decentralized Multiplication 16
5.3 Multi-Secret Sharing Multiplication 18
5.4 Multi-Secret Sharing Addition 20
5.5 Secure Comparison 22
6 Privacy-preserving kNN based on multi-secret sharing 24
6.1 System Initialization 24
6.2 Data Outsourcing 25
6.3 Request Generation 25
6.4 Service Processing 26
7 Security Analysis 28
7.1 Security of Composition Protocol 28
7.2 Security of kNN Classification 32
8 Discussions 33
8.1 Theoretical Analysis 33
8.2 Experimental Results 34
9 Conclusions 37
References 38

List of Tables

5.1 Notations. 15

8.1 Complexities (computation, communication, storage) of each protocol. 34

8.2 Parameters of each dataset. 35

List of Figures

3.1 MSS mechanism for sharing and recovery. 7

4.1 System model. 10

5.1 An example for distribution and reconstruction. 16

5.2 Scalar decentralized multiplication. 18

5.3 Multi-secret sharing multiplication. 20

5.4 Multi-secret sharing addition. 22

8.1 Average accuracy in different datasets. 35

8.2 Average execution time with different numbers of participants. 36

8.3 Average execution time with varying thresholds. 36

Chapter 1 Introduction

1.1 Background

Artificial intelligence (AI) is one of the critical developments in the future, which can solve problems through training and inferring. Many companies began integrating AI into their products and services, such as face recognition [1] and decision recommendation [2]. Generally, we will apply machine learning algorithms to data and make an induction analysis of instances into a model. After that, we can infer optimal decisions through statistical approximation situations in the model. Since considering more relevant data from different sources is necessary for training a more robust model, the opportunity to share and integrate data among multiple parties increases as we intend to form better intelligent services. However, data becomes an important asset. Although there are some studies [3–5] focusing on securely collecting data, people may worry that their sensitive data will be leakage due to the outflow of integrated data. Therefore, people still want to supervise their shared data while others usage.

Shamir's secret sharing [6] is a cryptographic primitive that can make a secret into multiple shares and recover the secret with enough shares. A t-out-of-n secret sharing protect scheme will distribute shares among n participants and retrieve a piece of confidential information only if sufficient t participants cooperate. Since the recovery requires participant collaboration, secret sharing has the advantage of overseeing the employment of shared resources. Additionally, secret sharing provides the property of homomorphic addition that does not rely on complex calculations, which is more suitable for IoT or mobile devices as a lightweight solution. Nevertheless, a share can only handle a single secret, which means each participant needs to hold massive shares for the integrated dataset.

As an extension of secret sharing, multi-secret sharing (MSS) can imply multiple secrets in a sharing process. That is, a share can maintain several secrets at the same time. Although some research [7] points out a bound of d secrets ($d \leq t < n$) in the conventional multi-secret sharing approach. We found a novel MSS [8] scheme that can bypass the barrier by fixing some shares in secret sharing, then producing others as public information. After that, we can recover secrets through combinations between the fixed part held by participants and the shares in the public. Note that we denote multi-secret sharing with public information as MSS since we do not consider conventional multi-secret sharing in this thesis.

Secure computation is a technique that allows performing calculations over ciphertexts and is often a functional need in practical secure systems. Secret sharing derives from polynomials, where support addition implies the sum of secrets. While multiplying shares is equivalent to the product of secrets in polynomials, a higher polynomial degree requires a higher reconstructing threshold. Thus, we lack enough shares to recover the product correctly, which makes secret sharing multiplication a problem. Some works use polynomial reduction [9] to solve this difficulty, but it becomes troublesome with complicated interactions in MSS schemes. Shingu et al. [10] gave a clever idea to deform two polynomials into a polynomial and a scalar value during secure multiplication. This scheme uses a random number as a mask to convert the secret into a hidden secret called concealed secret, then distributes the hidden secret by secret sharing. Accordingly, we can securely produce multiplication without changing the polynomial degree by revealing the concealed secret for performing scalar multiplication with other shares. Furthermore, they also proposed secure addition based on their construction which reproduces the homomorphic addition property in their work.

Unfortunately, secure computation will generate a new set of shares representing the calculation result. While persistent calculation occurs, we may do more secure computations above some result shares, which makes massive temporary shares a storage burden. Indeed, these additional shares go against the concept that a user maintains a single share in cloud assistance multi-secret sharing (CAMSS). Therefore, avoiding participants from holding more shares while secure computation has become a significant challenge. Finally, we propose a privacy-preserving computing architecture based on multi-secret sharing and design a series of protocols that support persistent computing for resource-restricted devices. Furthermore, we return origin and realize secure model training with our system. In the experiment, we demonstrate the feasibility and usability of our work under vast ciphertext computations through privacy-preserving k-nearest neighbor (kNN) classification.

1.2 Contributions

In this thesis, we propose a privacy-preserving persistent computation architecture based on dual-cloud and multi-secret sharing. The main contributions of this thesis are summarized as follows.

1. We build a secure computation architecture that can supervise and calculate simply for the shared secrets through the MSS scheme on dual-cloud.
2. Our solution designs a series of MSS protocols that support persistent computation without holding more shares at resource-restricted devices.
3. We test the feasibility of our work and show the usability with vast operations in privacy-preserving k-nearest neighbor classification. Moreover, in the experiment, we also demonstrate model training maintains high accuracy for several real-world datasets under our scheme.

1.3 Organization

The remainder of this thesis organizes as follows. In Chapter 2, we review the related works of secret sharing and secure computation. Chapter 3 introduces some preliminaries used in this work. Chapter 4 mentions our system model and design goals. Chapter 5 describes the structure and protocol we propose. Chapter 6 will talk about the application of our work. Chapter 7 gives the security analysis. Chapter 8 shows the experimental results. Finally, the conclusions of this thesis give in Chapter 9.

Chapter 2 Related Works

For dealing with the secure multiparty computation on IoT or mobile devices, we comprehensively investigate secret sharing and sort out previous research involving secure computation. Meanwhile, we will mention some existing studies about multi-secret sharing techniques in the survey.

2.1 Secret Sharing

Secret sharing is an essential primitive that does not need complicated computations, which is also a well-known unconditional security method. Without hardness assumptions, it has become a lightweight strategy for secure multiparty computations on IoT or mobile devices. Moreover, it is also expected as an approach to resist quantum attacks. There is some research [11–13] making secret sharing a solution for security issues.

Kaur et al. [11] proposed an interesting 2-line secret sharing approach for generating diverse pseudo-biometric identities for accessing different applications from a single user-specific token. Furthermore, they cascade this sharing concept over cancelable biometrics to form advanced template protection in remote biometric authentication.

Chor et al. [12] started the verifiable secret sharing scheme, which solves the Byzantine faults problems through simultaneous broadcast with fault-tolerant distributed computing. Then, Stadler [13] presented publicly verifiable secret sharing, which provides a verification mechanism that achieves everyone to verify the correctness of the shares, not only the participants.

2.2 Extension for Secret Sharing

Apart from the above secret sharing-based studies, Attasena et al. [14] survey the security for secret sharing and cloud data. From this survey, we not only can glance at various secret sharing but also peek at some multi-secret sharing technologies that deal with the drawback of multiplying initial data volume by the number of participants. After that, we feature two papers [15,16] about multi-secret sharing to display an extension for secret sharing.

Nag et al. [15] gave a multi-secret sharing scheme that hides multiple secrets at once and distributes them on n clouds by t-out-of-n conventional multi-secret sharing. While a user gets t private keys from the data owner, the user can recover whole secrets by downloading the shares in t clouds. Although MSS with more clouds brings the benefit of fault tolerance, it packs secrets together that lose the ability to modify particular secrets.

Regarding the expensive costs for publicly verifiable secret sharing on low computational ability devices, Lin et al. [16] presented a publicly verifiable multi-secret sharing scheme that outsourced the massive computational tasks to the cloud. Despite the verifiability, they could not accomplish other secure computations with the verification mechanism.

2.3 Secure Computation

To increase the utility of cryptographic tools, reducing the cost and requirements for the scheme is the crux of practicality. Indeed, the environment will not always have unbounded power as our assumption. Thus, computing over the confidential domain is an essential direction for studying. In particular, protecting the privacy of input test data and models is a critical task for machine learning in reality. Therefore, we review a previous work [17] that achieves efficient privacy-preserving k nearest neighbor classification.

Yang et al. [17] proposed a PPkNN to complete efficient supervised learning based on secret sharing. During the training process, neither the model nor the query data leaks by computations. Following the construction, decision tree-based preprocessing can speed up the classification time and obtain the results with a certain degree of accuracy.

In the chapters to come, we attempt to adopt multi-secret sharing into secure computation and emphasize the usability of resource-restricted devices.

Chapter 3 Preliminaries

3.1 Multi-Secret Sharing

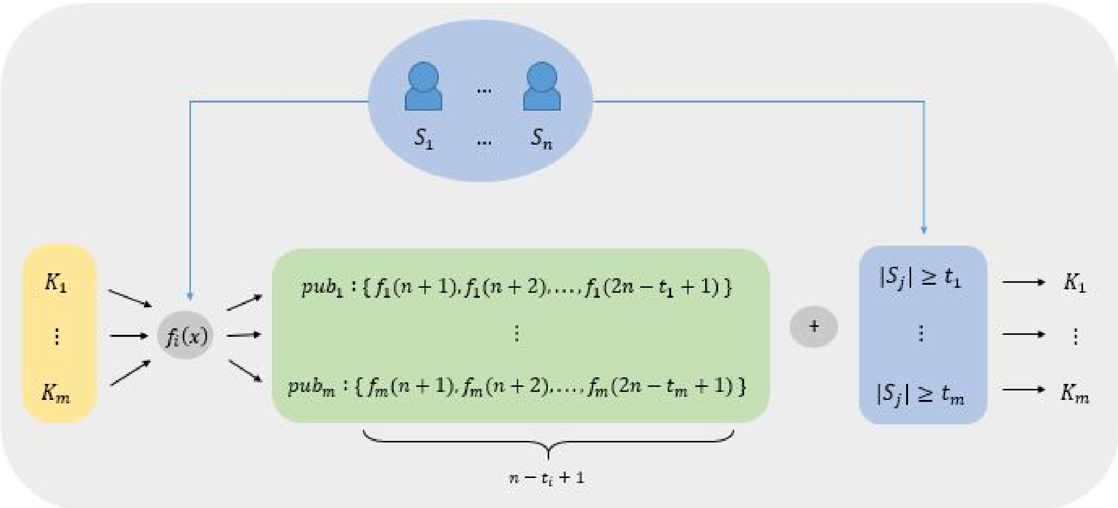


Figure 3.1: MSS mechanism for sharing and recovery.

Multi-secret sharing allows users to hold the same share for reconstructing multiple secrets. In our construction, we focus on the MSS scheme proposed in [8]. Firstly, we decide the participant shares S, which are some shares for each secret. Then, generate the other shares as public information pub from secrets and fixed shares. Finally, we can recover the secret K with at least t participant shares and the public shares. Note that we can also represent $[K]$ as the participant shares and public shares of the secret K in protocols.

Figure 3.1 shows the sharing and recovering process in the MSS scheme, where there are m secrets to be shared among n participants ($t \leq n$). Then MSS scheme needs to meet the following terms:

- According to the public shares pub_i, any t_i or more than t_i users can recover the secret K_i.
- If there is no public share, we cannot obtain any information even if all users participate.

- The secret should not be revealed while there are less than t_i participants.

In the sharing process, we got $n+1$ points $(0, K_i)$, (j, S_j) for each secret K_i by participant share S_j, where $j = \{1, 2, ..., n\}$. Then, we consist $f_i(x)$ of the following formula:

$$f_i(x) = K_i \prod_{j=1}^{n}(1 - \frac{x}{j}) + \sum_{j=1}^{n} S_j \prod_{l=0, l \neq j}^{n} \frac{x-l}{j-l}$$

, where $i = \{1, 2, ..., m\}$. After that, we can get the public shares $pub_i = \{f_i(n+1), f_i(n+2), ..., f_i(2n - t_i + 1)\}$. And finally, any t_i participant shares and $n - t_i + 1$ public shares in pub_i can recover secret K_i by reconstructing a n+1-out-of-n+1 secret sharing under a prime modulo p.

We can present the MSS scheme as follows:

- Share function creates public shares from the secret and the participant shares. $Share(K_i, S) \rightarrow pub_i$.
- Recover function will take enough participant shares and the public shares to rebuild the secret. $Recover(pub_i, S) \rightarrow K_i$.

3.2 Beaver Triples

Beaver et al. [18] introduce an approach to deal with secure multi-party multiplication issue in secret sharing, called multiplication triples (also often called Beaver triples). Let $[z]$ denote the share of secret $z = x \cdot y$, where the share of secret x and y are represented as $[x]$ and $[y]$. We can generate Beaver triples $([a], [b], [ab])$ to make $[\alpha] = [x] - [a]$ and $[\beta] = [y] - [b]$. Then, we recover α and β, which are secure for secrets since we distribute a and b as shares in each party. Lastly, we can locally computes $[z] = \alpha \cdot \beta + \alpha \cdot [b] + \beta \cdot [a] + [ab]$.

Correctness: The multiplication without degree increment supports local computation via scalar multiplication and share addition. We demonstrate the correctness below. Here we define R^* as a general secret sharing recover function.

$$
\begin{aligned}
z &= R^*([z]) \\
&= R^*(\alpha \cdot \beta + \alpha \cdot [b] + \beta \cdot [a] + [ab]) \\
&= \alpha\beta + \alpha \cdot R^*([b]) + \beta \cdot R^*([a]) + R^*([ab]) \\
&= (x-a)(y-b) + (x-a) \cdot b + (y-b) \cdot a + ab \\
&= xy - ay - bx + ab + bx - ab + ay - ab + ab \\
&= xy
\end{aligned}
$$

3.3 k-nearest Neighbor Algorithm

The k-nearest neighbor (kNN) algorithm is a case-based learning algorithm that computes the distances between the query data and the training instances, then determines the result by finding the k nearest labels with distance comparison.

k-nearest neighbor algorithm:

1. Calculate the distances from the test instance to the instances in the training set.
2. Compare the distances and get the labels of the k-th nearest neighbors.
3. Set the test label as the major category among the nearest data.

Since privacy-preserving kNN requires secure comparisons over the securely computed distances, each party may have at least as many temporary shares as the dataset as a storage burden for persistent calculations.

Chapter 4 System Framework

4.1 System Model

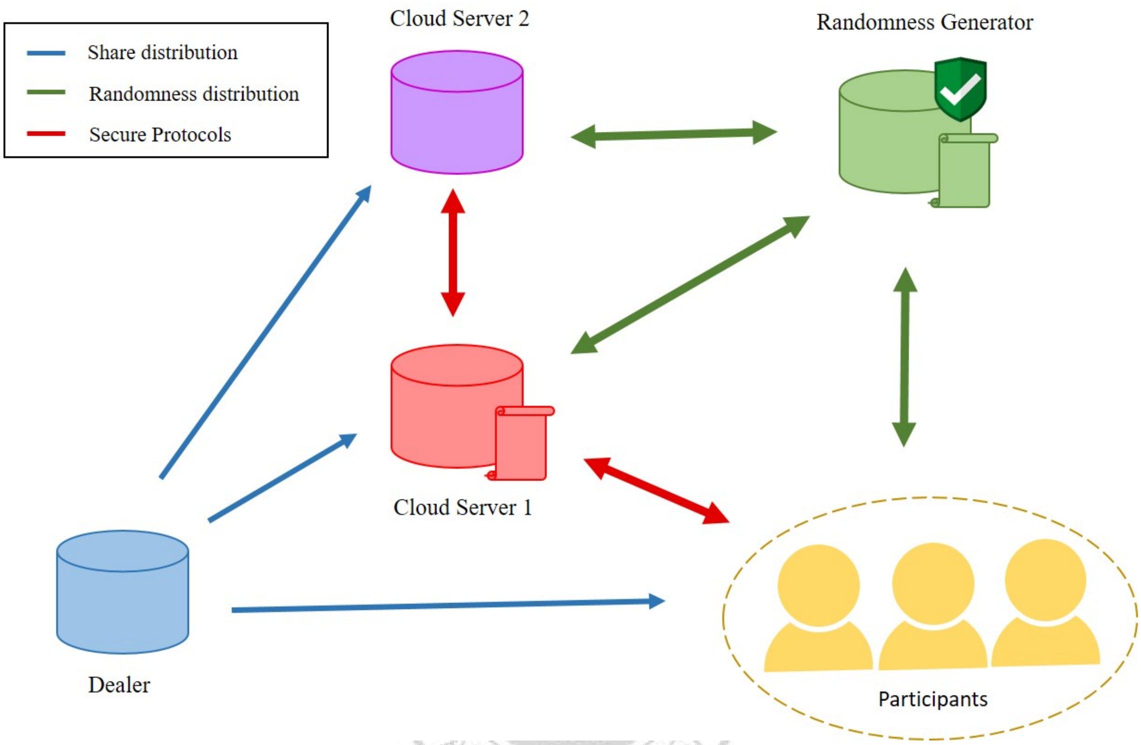


Figure 4.1: System model.

Figure 4.1 briefly presents the main flow of our system. Our construction contains four types of entities, e.g., a dealer, dual-cloud servers, multiple participants, and a randomness generator. Moreover, we define two record lists named randomness and operation in our system.

1. **Dealer**: The dealer is a trusted entity that collects data from multiple participants to form a database. Whenever we decide to build a service, the database will give a dataset D that integrates instances from various participants to train a more robust model. Considering the cost and the limited computing power, we cannot provide the service simply through the dealer. So we introduce cloud computing for training models and providing services. However, the dealer is suspicious about the security of the cloud. Therefore, the dealer

returns the management capabilities to all participants, as the dataset may contain sensitive data. In addition, the dealer will mask the data before outsourcing the dataset to the cloud. In our system, the dealer is responsible for system initialization which includes collecting and integrating data, distributing participant shares, masking the dataset, and outsourcing the public shares. After that, the dealer can go offline since we no longer need interaction with it in protocols.

2. **Dual-cloud servers**: The two non-colluding cloud servers are semi-honest entities, where CS_1 holds the public shares of the mask and CS_2 holds the public shares from the masked data. They both have abundant storage space and powerful computational ability, which support participants performing secure computations under the MSS system. We protect the privacy of raw data by preventing the simultaneous recovery of related secrets between the two servers. Furthermore, we build an operation record OpRec at CS_1, which saves a list of parameters for rebuilding the MSS operations. The OpRec allows more secure computations above the recorded operations, which makes temporary shares unnecessary stored in cloud servers and participants during persistent secure computation.

3. **Participants**: The participant P is a data provider who holds a single share to maintain secrets on dual-cloud servers. They are also the service user who launches a query Q to get the service from the outsourced dataset in our system. Since the participants can access the secret by satisfying the threshold, there must be enough collaborators to agree with the cooperative computation for completing the service. Although we do not initially share the query in the cloud, we can securely upload the query data through the randomness generator and obtain the result from privacy-preserving computations with the servers.

4. **Randomness Generator**: The randomness generator RG is a trusted third party that provides randomization parameters by converting the scalar multiplication whenever share usage. We can refresh the shares with different randomness parameters generated from the same number, which randomizes

the mask shares in CS_1 and the share of the masked data in CS_2, respectively. Besides, we can also take advantage of this computing manner to transform the query data for the participant in our system. Similarly, we build a randomness record RandRec in RG, which holds the parameters for backtracking the randomizing shares and supports privacy-preserving during persistent computation.

Remark: We have defined two record lists to reduce the storage burden of temporary shares in persistent computation. Besides, we denote two statements for record operation as follows.

- Append statement adds a new line of parameters at the end of the record list.
- Delete statement clears the record list after the temporary data is no longer needed.

4.2 Threat Model

In this thesis, we assume that only the dealer and the randomness generator are trustable, which generate shares honestly and distribute them securely to all parties. On the other hand, we consider the participants and the dual-cloud servers to be semi-trusted, where they perform interactive protocols curious-but-honestly. That means they will faithfully follow the protocol for getting the correct results and try to infer the information held by others from the intermediate messages. Besides, the dual-cloud servers that do not collude are valid and widely used in cryptography works [19]. Using cloud servers belonging to two different companies with high social reputations can prevent them from colluding with each other since they would not take the risk of losing their market credibility.

Therefore, we introduce an adversary $\mathcal{A}$, whose goal is to gain unknown shares for higher access and leak the privacy of the dataset and query data. The adversary $\mathcal{A}$ can eavesdrop on the communication between the cloud servers and the participants to obtain the encrypted data and also get the assistance of the colluded parties. Although the dual-cloud servers collusion is restricted, the adversary $\mathcal{A}$ still can work together with a single cloud server and several participants in the worst case. Note that a powerful threat model for two cloud servers colluding with each other is beyond our discussion scope.

4.3 Design Goals

Based on the system model and threat model, the multi-secret sharing computation architecture proposed in this thesis achieves the following design goals.

1. **Privacy protection**: In our system, the dataset is a data collection from multiple participants, in which the secret must recover by at least t participants. In other words, the computation should also reach a threshold in order to access and calculate the ciphertexts. Moreover, the non-collusive dual-cloud servers should learn no information and constitute a perfect cooperative pattern for multiple parties.

2. **Low storage**: Considering that the participant may use a resource-restricted device, we enable the participant to manage multiple secrets with the same share in multi-secret sharing. That is, the participant share does not grow with the size of the dataset, which gives more opportunities to apply in restricted situations, such as IoT devices, edge computing (autonomous vehicle driving system), or low Earth orbit satellite communication networks (Starlink).

3. **Persistency**: Computations based on secret sharing often generate new shares for the calculation results. However, this violates the low storage advantage of multi-secret sharing. Therefore, we expect to build secure protocols that would not increase shares during computations. Then extend them to a persistency calculation that can perform more ciphertext calculations on results without producing new shares. Note that we claim the computing manner is "persistent" as the computation is traceable back to the initial shares, which

supports maintaining a single share for everyone under cloud assistance multi-secret sharing.

4. **Correctness**: The secure protocols we designed must be able to obtain correct results through ciphertext calculations. In the following experiment, we will compare the classification results obtained by the ordinary kNN algorithm and the privacy-preserving kNN built by our scheme. As long as the accuracy is close, we can confirm that it is a possible privacy-preserving solution for applications whose calculation amount is under kNN massive computations.

Chapter 5 Proposed Scheme

In this chapter, we first describe the mechanism of the distribution and reconstruction. Then, we present secure protocols as basic operations above multi-secret sharing. Table 5.1 are some notations we have in our scheme.

Table 5.1: Notations.

Notation	Description
n	The number of participants.
t	The threshold of the secret.
$\langle x \rangle = ([\alpha], [\alpha x])$	The protected data of secret x.
$S_i (= [\alpha]_i = [\alpha x]_i)$	A share of participant P_i in MSS that can recover any protected data in a cloud server.
$[\alpha]_{pub}$	A set of public shares in CS_1 that stands for the mask α.
$[\alpha x]_{pub}$	A set of public shares in CS_2 that stands for the masked data αx.
$Share_{(S)}(x, t)$ $= ([\alpha]_{pub}, [\alpha x]_{pub})$	Distributing public shares above the participant shares S, where secret x split as the protected data.
$Recover_{(t)}(S, [\alpha x]_{pub})$ $= \alpha x$	Recovering the value of protected data from public shares and at least t participant shares.
$\langle x + y \rangle$	The MSSAdd computes the addition of protected data $\langle x \rangle$ and $\langle y \rangle$, where x and y are secrets in MSS.
$\langle x * y \rangle$	The MSSMul computes the multiplication of protected data $\langle x \rangle$ and $\langle y \rangle$, where x and y are secrets in MSS.

5.1 Distribution and Reconstruction

Figure 5.1 shows an outsourced format and protected recovery of each secret. In the distribution phase, we randomly select a mask α for each data x and obtain the masked data $\alpha \cdot x$. Then, we can outsource the protected data through MSS that involves the participant shares. That is, CS_1 holds the mask part as $[\alpha]_{pub}$, and CS_2 stores the masked data as $[\alpha x]_{pub}$. In the reconstruction phase, we pick a random number r to refresh the shares of the protected data, where the mask and the masked data change but still maintain the same secret. Thus, we can protect the recovery in different rounds from exposing the information of shares due to the components previously disclosed on the servers. Finally, we can reconstruct the secret through components on CS_1 and CS_2 at the same round. Note that for MSS consisting of secret sharing under a prime modulo p, all secrets should be less than p, including the mask, the masked data, and the refreshed ones.

	(Distribution phase)	
Secret	x	
Masking secret	α	αx
Outsourcing to different cloud server	$[\alpha]_{pub}$	$[\alpha x]_{pub}$

	(Reconstruction phase)	
Shares	$[\alpha]$	$[\alpha x]$
Refreshing shares	$[r\alpha]$	$[r\alpha x]$
Recovering on servers	$r\alpha$	$r\alpha x$
Reconstructing secret	x	

Figure 5.1: An example for distribution and reconstruction.

5.2 Scalar Decentralized Multiplication

In our construction, we rely on the randomness generator to multiply the scalar into the share. While secret sharing already has the property of scalar multiplication, revealing the scalar value is unacceptable in MSS. In other words, the recovery of the multiplication result would expose the original secret by knowing the constant value during the local computation, where each party directly multiplies the constant value c by their share $[x]$ to form a result share $[cx]$. Therefore, we design Protocol 1 to securely refresh the shares by decentralizing a scalar multiplication.

Protocol 1 Scalar Decentralized Multiplication (SD)

RG	**Input**: multiplier y, $\mathsf{RandRec}$
CS	**Input**: $[x]_{pub}$
P_i	**Input**: $S_i(=[x]_i)$
RG	**Output**: $\mathsf{RandRec}'$
CS, P_i	**Output**: $[xy]_i$

1. RG randomly select a, b and computes $c = ab$.

2. RG generates $2n - t + 1$ shares $[y], [a], [b], [c]$.

3. RG adds the parameters into the randomness record
$\mathsf{RandRec}.append([y], [a], [b], [c]) \rightarrow \mathsf{RandRec}'$.

4. RG sends $[y]_i, [a]_i, [b]_i, [c]_i$ to P_i and CS, respectively.

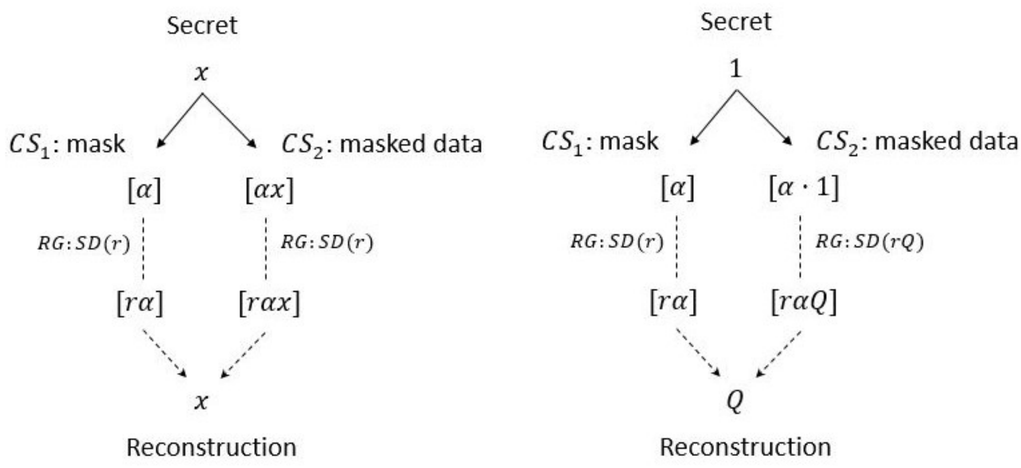
5. P_i send $[x]_i - [a]_i$ and $[y]_i - [b]_i$ to CS.

6. CS obtain $\varepsilon = x - a$ and $\rho = y - b$, and send them to P_i.

7. Finally, P_i and CS can locally computes $[xy]_i = \rho([x]_i - [a]_i) + \varepsilon[b]_i + \rho[a]_i + [c]_i$.

In this protocol, the randomness generator converts the multiplier into the form of secret sharing and applies Beaver triples for efficient share multiplication among multiple parties. Since secret sharing can make different shares out of the same number, we can securely refresh the shares by multiplying a random number into the mask and the masked data. Additionally, RG will record the sending parameters $([y], [a], [b], [c])$ so that we can recall the previous refreshed shares through $\mathsf{RandRec}$ to form persistent computations.

Moreover, we can modify the reconstructed secret by multiplying not equal numbers on the protected data. Whenever the mask increase, the reconstructed secret gets divided more. On the contrary, the reconstructed secret can multiply a number as large as the scalar multiplication of the masked data, which can become an alternative for share generation after the system initialization. Figure 5.2 illustrates an example of refreshing shares and uploading data.



(a) Refreshing shares with a random number r.

(b) Uploading data Q with a random mask r.

Figure 5.2: Scalar decentralized multiplication.

5.3 Multi-Secret Sharing Multiplication

Different from the SD protocol, which only performs scalar multiplication of the mask or the mask data. Protocol 2 can construct the product share of two secrets through the protected data. At the beginning of the MSS multiplication, RG refreshes the protected data of both x and y. Interestingly, even though the participants share $S_i(= [\alpha]_i = [\alpha x]_i)$ and multiplier r_1 are the same in Protocol 1 for the mask and the mask data, $[r_1\alpha]_i$ and $[r_1\alpha x]_i$ are distinct since the multiplier r_1 splits into different shares $[r_1]$ each time. That is, the refreshed participant share can only use for specific objects. As $[r_1\alpha]_i \neq [r_1\alpha x]_i$ for the same scalar r_1, CS_2 could recover nothing with $[r_1\alpha x]_{pub}$ even if CS_2 eavesdrop on the message $[r_1\alpha]_i$ that the participant passed to CS_1.

Protocol 2 Multi-Secret Sharing Multiplication (MSSMul)

CS_1 **Input**: $[\alpha]_{pub}$, $[\beta]_{pub}$, OpRec

CS_2 **Input**: $[\alpha x]_{pub}$, $[\beta y]_{pub}$

P_i **Input**: $S_i(= [\alpha]_i = [\beta]_i = [\alpha x]_i = [\beta y]_i)$

CS_1 **Output**: $[r_1 r_2 \alpha\beta]_{pub}$, OpRec'

CS_2 **Output**: $[r_1 r_2 \alpha\beta xy]_{pub}$

P_i **Output**: $[r_1 r_2 \alpha\beta]_i$, $[r_1 r_2 \alpha\beta xy]_i$

1. RG randomly selects two random multipliers r_1 and r_2 for $[\alpha], [\alpha x]$ and $[\beta], [\beta y]$ in Protocol 1.

2. P_i sends $[r_1\alpha]_i$ to CS_1 then obtains $r_1\alpha$.

3. P_i sends $[r_2\beta y]_i$ to CS_2 then obtains $r_2\beta y$.

4. CS_1, CS_2 adds the parameters to the operation record $\mathsf{OpRec}.append(*, r_1\alpha, r_2\beta y) \rightarrow \mathsf{OpRec}'$.

5. Finally, CS_1, CS_2, P_i receives $r_1\alpha$ and $r_2\beta y$. Then CS_1 computes $r_1\alpha[r_2\beta]_{pub}$, CS_2 computes $r_2\beta y[r_1\alpha x]_{pub}$, and P_i locally computes $r_1\alpha[r_2\beta]_i, r_2\beta y[r_1\alpha x]_i$.

After that, we recover $r_1\alpha$ in CS_1 and $r_2\beta y$ in CS_2. The multiplication of two polynomials on secret sharing is deformed into a scalar and a polynomial so that the degree of the polynomial is not changed and the recovery can maintain the equivalent threshold for the calculation result. Figure 5.3 shows the calculation process of MSSMul. OpRec only needs to store two recovery values for rebuilding the product shares, and we can perform more MSS calculations above these shares.

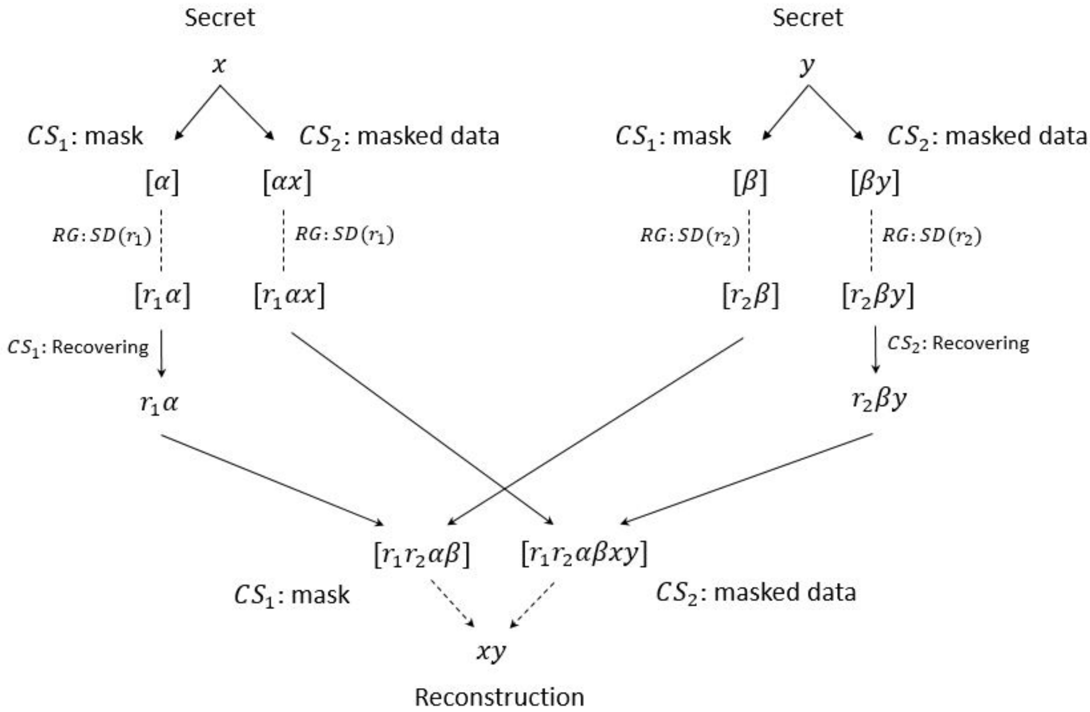


Figure 5.3: Multi-secret sharing multiplication.

5.4 Multi-Secret Sharing Addition

The linear relationship of secret sharing has the characteristic of summing secrets through shares addition. However, this could only help adjust a part of the protected data. Since the additive on the masks or the masked data does not directly mean the sum for the original secrets. We design Protocol 3 to build secret addition based on the masks and masked data.

Similarly, MSSAdd begins by refreshing the protected data of x and y. Then, CS_1 recovers the masks $r_1\alpha$ and $r_2\beta$. After that, we multiply the mask data by the other mask value so the protected data can transform into the same mask $r_1r_2\alpha\beta$, and the sum of the masked data can imply the secret addition. Although we expose the value of the new mask, we compute $[r_1r_2\alpha\beta x] + [r_1r_2\alpha\beta y]$ without recovering $[r_1r_2\alpha\beta x]$ and $[r_1r_2\alpha\beta y]$. Thus, we do not leak the original secret while generating the summation shares, where the feature of available but invisible to the masked data also reflects in secret.

Protocol 3 Multi-Secret Sharing Addition (MSSAdd)

CS_1 **Input**: $[\alpha]_{pub}$, $[\beta]_{pub}$, OpRec

CS_2 **Input**: $[\alpha x]_{pub}$, $[\beta y]_{pub}$

P_i **Input**: $S_i(= [\alpha]_i = [\beta]_i = [\alpha x]_i = [\beta y]_i)$

CS_1 **Output**: $[r_1 r_2 \alpha\beta]_{pub}$, OpRec'

CS_2 **Output**: $[r_1 r_2 \alpha\beta x + r_1 r_2 \alpha\beta y]_{pub}$

P_i **Output**: $[r_1 r_2 \alpha\beta]_i$, $[r_1 r_2 \alpha\beta x + r_1 r_2 \alpha\beta y]_i$

1. RG randomly selects two random multipliers r_1 and r_2 for $[\alpha], [\alpha x]$ and $[\beta], [\beta y]$ in Protocol 1.
2. P_i sends $[r_1\alpha]_i$ to CS_1 then obtain $r_1\alpha$.
3. P_i sends $[r_2\beta]_i$ to CS_1 then obtain $r_2\beta$.
4. CS_1 adds the parameters into the operation record
   $\mathsf{OpRec}.append(+, r_1\alpha, r_2\beta) \rightarrow \mathsf{OpRec}'$.
5. Finally, CS_1, CS_2, P_i receives $r_1\alpha$ and $r_2\beta$. Then CS_1 computes $r_2\beta[r_1\alpha]_{pub}$, CS_2 computes $r_2\beta[r_1\alpha x]_{pub} + r_1\alpha[r_2\beta y]_{pub}$, and P_i locally computes
   $r_2\beta[r_1\alpha]_i, r_2\beta[r_1\alpha x]_i + r_1\alpha[r_2\beta y]_i$.

Figure 5.4 shows the calculation process of MSSAdd. OpRec provides the two masks to each party so that they can compute the summation share locally and do more MSS calculations with the result.

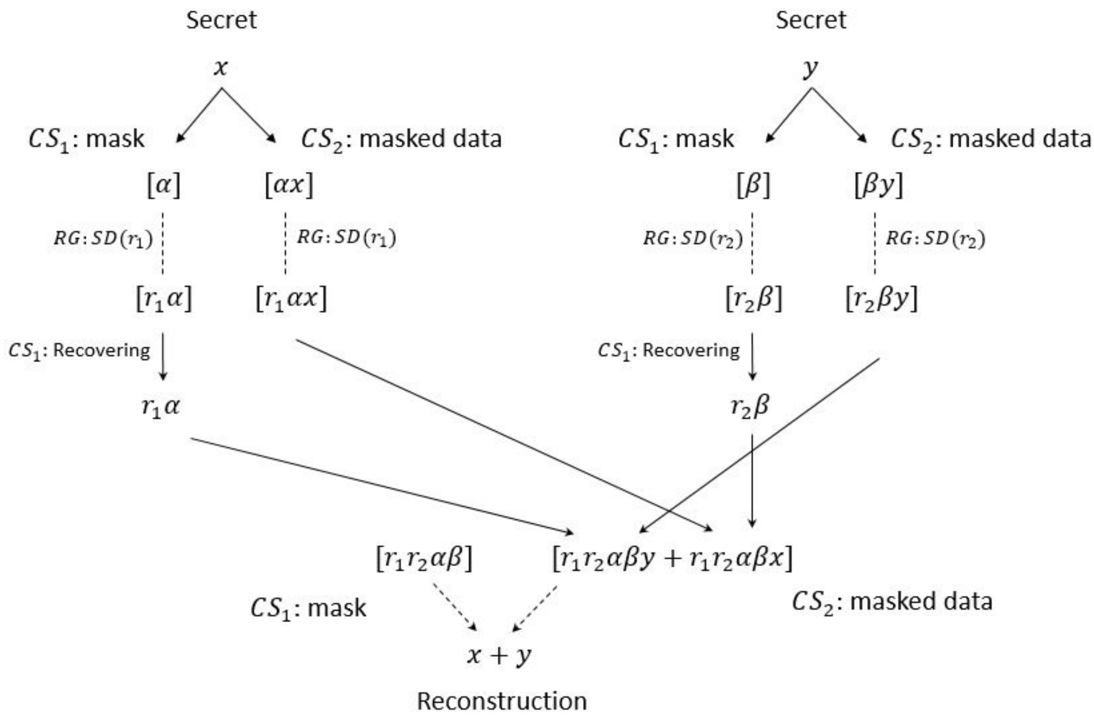


Figure 5.4: Multi-secret sharing addition.

5.5 Secure Comparison

While refreshing protected data is helpful for computational security, the secret remains intact from the mask and the masked data. Considering the comparison may leak the difference as soon as revealing the relationship between secrets, we should add randomness to them throughout the secure comparison.

Protocol 4 applies the above MSS operations to construct the modification on the standard value ℓ and the comparison value. For simplicity, we denote $\langle x \rangle = ([\alpha], [\alpha x])$ as the shares that offer by multiple parties, which is associated with secret x for collaboratively computing MSS operations. Additionally, we represent MSSMul as $\langle \cdot * \cdot \rangle$ and MSSAdd as $\langle \cdot + \cdot \rangle$, which cooperatively calculates on protected data.

Protocol 4 Secure Comparison (SC)

CS_1, CS_2, P_i **Input**: $\langle x \rangle = ([\alpha], [\alpha x])$, $\langle y \rangle = ([\beta], [\beta y])$

CS_1, CS_2, P_i **Output**: 1 if $(x > y)$ or 0 if $(x \leq y)$

1. RG produces $\langle \ell \rangle$, where $x, y \leq \ell$ (ℓ is a public parameter).
2. CS_1, CS_2, P_i collaboratively compute $\langle z \rangle = \langle x - y + \ell \rangle$.
3. RG uniformly picks two positive integers r and r', where $2\ell r + r' < q$, and then generates $\langle r \rangle, \langle r' \rangle$.
4. CS_1, CS_2, P_i collaboratively compute $\langle e \rangle = \langle z * r + r' \rangle$ and $\langle h \rangle = \langle \ell * r + r' \rangle$.
5. CS_1, CS_2, P_i collaboratively reconstruct e and h. Then, outputs 1 if $e > h$ or 0 if not.

In SC protocol, RG generates the shares of parameters ℓ, r, r' by uploading them as shown in Figure 5.2(b). After that, we calculate the subtraction of $\langle x \rangle$ and $\langle y \rangle$, where the MSS subtraction $\langle \cdot - \cdot \rangle$ can regard as adding the inverse element of the subtrahend to the minuend through MSSAdd. Since the additive inverse always exists in modulus, the negative can represent by shares in MSS under modulo p. And finally, we imply the difference into $\langle e \rangle$ and $\langle h \rangle$ through consistent changes, which the reconstruction could reveal the relationship but not the exact gap between the secrets.

Chapter 6 Privacy-preserving kNN based on multi-secret sharing

In this section, we return to the background and use our proposed MSS tools to establish a privacy-preserving model computation method among multiple parties. During the process, we outsource the main calculation work to the cloud servers and compute the dataset and the query data in the shared state to keep data information from leakage. Additionally, each participant only holds a single share that provides access to the data, which is universal for the entire calculation and maintains the cloud assistance multi-secret sharing throughout the service. Our privacy-preserving k-nearest neighbor classification consists of four stages, i.e., system initialization, data outsourcing, request generation, and service processing.

6.1 System Initialization

This system contains four types of entities, namely the dealer, dual-cloud servers (CS_1, CS_2), multiple participants ($P_1, P_2, ..., P_n$), and a randomness generator (RG). First of all, the dealer provides the dataset $D = (d, l)$ for the classification service, where $d = \{d_{1,1}, ..., d_{v,m}\}$ is the total data from the v instances and m attributes, and $l = \{l_1, l_2, ..., l_v\}$ is the label for each instance. Then, the participant may have some queries $Q = \{Q_1, Q_2, ..., Q_u\}$, where u is the number of the request and $q = \{q_1, q_2, ..., q_m\}$ is the query data for each request. Since we assume that the data is composed of rounding the floating-point numbers into integers, the modulo calculation always holds during secret sharing. Moreover, the dealer adds 1 into the dataset as an assistant value for MSS calculation. After that, the dealer generates the participant shares $S = \{S_1, S_2, ..., S_n\}$ and sends S_i to participant P_i, respectively.

6.2 Data Outsourcing

With the participant shares S, the dealer can convert the dataset into public shares in MSS and outsources the kNN classification service on cloud servers. To protect dataset D, the dealer randomly picks the masks and multiplies them with the data in the dataset to produce the masked data. Then, the dealer generates public shares of the protected data by multi-secret sharing and distributes them to CS_1 and CS_2.

Carrying out function $Share_{(S)}(x, t) = ([\alpha]_{pub}, [\alpha x]_{pub})$ with dataset $D = (d, l)$. We denote the MSS shares related to secret x as $\langle x \rangle$ that consists of $[\alpha]_{pub}$, $[\alpha x]_{pub}$ and $S_i(= [\alpha]_i = [\alpha x]_i)$, which are shares in CS_1, CS_2 and participant P_i, respectively. Therefore, the system has $\langle \boldsymbol{d} \rangle = \{\langle d_{1,1} \rangle, ..., \langle d_{v,m} \rangle\}$ and $\langle \boldsymbol{l} \rangle = \{\langle l_1 \rangle, \langle l_2 \rangle, ..., \langle l_v \rangle\}$ after outsourcing. From now on, the dealer can get offline in our system.

6.3 Request Generation

After outsourcing the model to cloud servers, participants can upload query data and launch the request with participants that meet the threshold t for completing calculations in the classification service. Although multi-secret sharing asks the share generation to involve participant shares S, we can form the MSS shares of the query based on some existing MSS shares.

To securely upload the query q, we add an assistance value 1 at the system initialization that becomes $\langle 1 \rangle$ in our system. Then, we rely on RG to transform $\langle 1 \rangle$ into $\langle \boldsymbol{q} \rangle = \{\langle q_1 \rangle, ..., \langle q_m \rangle\}$ through the SD protocol. Since we can upload the data without knowing other shares, we can generate the request anytime and perform calculations with the protected dataset in the system.

6.4 Service Processing

Finally, we have the dataset $\langle \boldsymbol{d} \rangle$, $\langle \boldsymbol{l} \rangle$ and query data $\langle \boldsymbol{q} \rangle$ in our system. Next, we can accomplish the privacy-preserving kNN classification service by applying our proposed protocols. In Algorithm 1, we use the SC protocol to compare the relationship between the attributes of the query data and each dataset instance. Then, we can get the positive differences $\langle \triangle \rangle$ without learning any information about the query data and dataset. After that, we aggregate the differences and define the distance between the query and each instance label.

Once we have the distance, we can search the nearest neighbors through secure comparison, where distances are also an MSS share that sums the positive differences $\langle \triangle \rangle_i$ through MSSAdd. After finding the k-th nearest neighbors, we reconstruct the selected labels, where CS_1 obtains the mask and CS_2 gets the masked data. Then, we can determine which category belongs to the query. Before we end the classification, we can clear the calculation process by $\mathsf{OpRec}.delete()$ and $\mathsf{RandRec}.delete()$. Thus, we can cast off the accumulation of residual data that we no longer need for persistent computation.

Algorithm 1: Privacy-preserving kNN Classification

Input: dataset $\langle \boldsymbol{d} \rangle = \{\langle d_{1,1} \rangle, ..., \langle d_{v,m} \rangle\}$, $\langle \boldsymbol{l} \rangle = \{\langle l_1 \rangle, \langle l_2 \rangle, ..., \langle l_v \rangle\}$ and query $\langle \boldsymbol{q} \rangle = \{\langle q_1 \rangle, \langle q_2 \rangle, ..., \langle q_m \rangle\}$.
Output: respond category c.

for $i = 1$ *to* v **do**
  for $j = 1$ *to* m **do**
    if $SC(\langle d_{i,j} \rangle, \langle q_j \rangle)$ **then**
      $\langle \triangle_i \rangle \leftarrow \langle \triangle_i + d_{i,j} - q_j \rangle$;
    else
      $\langle \triangle_i \rangle \leftarrow \langle \triangle_i + q_j - d_{i,j} \rangle$;
    end
  end
  $dist[i] \leftarrow \langle \triangle_i \rangle$;
  $label[i] \leftarrow \langle l_i \rangle$;
end
for $i = 1; i < v; i++$ **do**
  $min \leftarrow i$;
  for $j = i + 1$ *to* v **do**
    if $SC(dist[min], dist[j])$ **then**
      $min \leftarrow j$;
    end
  end
  Swap $dist[i]$ and $dist[min]$;
  Swap $label[i]$ and $label[min]$;
  if $i == k$ **then**
    Reconstruct $label[1:k]$;
    Select the majority label as category c;
    break;
  end
end
Runs OpRec.$delete()$;
Runs RandRec.$delete()$;
return c;

Chapter 7 Security Analysis

In this chapter, we analyze the security of the proposed protocols, which serve as MSS tools for outsourcing services. To prove the composition is secure, we adopt the framework of universal composability (UC) [20] and show the view of the semi-honest adversaries. According to the threat model, we may consider the adversary with a view involving their inputs, the received messages, and the information from the colluding parties. Although the adversary may learn some protected data through the protocols, secrets never leak during persistent computations.

7.1 Security of Composition Protocol

Functionality 1 $\mathcal{F}_{SD}$: $\mathcal{F}_{SD}$ interacts with RG, CS, P_i and an adversary $\mathcal{S}$.

1. Upon receiving a message $([y]_i, [a]_i, [b]_i, [c]_i)$ from RG, send the message $([y]_i, [a]_i, [b]_i, [c]_i)$ to P_i, CS and $\mathcal{S}$.
2. Upon receiving a message $([x-a]_i, [y-b]_i)$ from P_i, send the message $([x-a]_i, [y-b]_i)$ to CS and $\mathcal{S}$.
3. Upon receiving a message (ϵ, ρ) from CS, send the message (ϵ, ρ) to P_i and $\mathcal{S}$.

Theorem 1 (Security of Π_{SD}). *Π_{SD} UC-realizes $\mathcal{F}_{SD}$.*

Proof. To explain the privacy of Π_{SD}, we first consider the view of RG. While $\mathsf{View}_{RG} = (y, a, b, c)$, RG has no information about xy and cannot learn x in Π_{SD}. Then, we consider an adversary $\mathcal{A}$ colludes with less than t participants, where $\mathsf{View}_{\mathcal{A}} = ([x]_i, [y]_i, [a]_i, [b]_i, [c]_i, \epsilon, \rho, [xy]_i)$ and $i = \{j_1, ..., j_{t-1}\}$. Since secret sharing is information-theoretic security, $\mathcal{A}$ does not have enough share to recover the real values of a, b, and xy. Consequently, x and y are secure even if ϵ and ρ is a known in Π_{SD}. Altogether, $\mathcal{A}$ causes nothing leakage and is a perfect simulation for $\mathcal{S}$ in an ideal world. Therefore, we can state Π_{SD} UC-realizes $\mathcal{F}_{SD}$.

Functionality 2 $\mathcal{F}_{MSSMul}$: $\mathcal{F}_{MSSMul}$ interacts with CS_1, CS_2, P_i and $\mathcal{S}$.

1. Run $\mathcal{F}_{SD}$ to refresh $\langle x \rangle = ([\alpha], [\alpha x])$ and $\langle y \rangle = ([\beta], [\beta y])$.

2. Upon receiving a message $([r_1\alpha]_i)$ from P_i, send $([r_1\alpha]_i)$ to CS_1 and $\mathcal{S}$.

3. Upon receiving a message $([r_2\beta y]_i)$ from P_i, send $([r_2\beta y]_i)$ to CS_2 and $\mathcal{S}$.

4. Upon receiving a message $(r_1\alpha, r_2\beta y)$ from CS_1 and CS_2, send the message $(r_1\alpha, r_2\beta y)$ to CS_1, CS_2, P_i and $\mathcal{S}$.

Theorem 2 (Security of Π_{MSSMul})**.** Π_{MSSMul} *UC-realizes* $\mathcal{F}_{MSSMul}$ *in the* ($\mathcal{F}_{SD}$)*-hybrid model.*

Proof. We define Π_{MSSMul} in the ($\mathcal{F}_{SD}$)-hybrid model. Since Π_{SD} randomly selects a and b, the environment machine $\mathcal{Z}$ cannot distinguish the worlds from the timely information. Thus, Π_{MSSMul} can UC-emulate Π_{SD} in the ($\mathcal{F}_{SD}$)-hybrid model. Moreover, we observe the views of CS_1 and CS_2, respectively. For $\mathsf{View}_{CS_1} = ([\alpha]_{pub}, [\beta]_{pub}, r_1\alpha, [r_2\beta]_{pub}, r_2\beta y\ , [r_1r_2\alpha\beta]_{pub})$, CS_1 cannot obtain y since participants do not offer $[r_2\beta]_i$. For $\mathsf{View}_{CS_2} = ([\alpha x]_{pub}, [\beta y]_{pub}, r_1\alpha, [r_1\alpha x]_{pub}, r_2\beta y, [r_1r_2\alpha\beta xy]_{pub})$, CS_2 cannot obtain x since participants do not offer $[r_1\alpha x]_i$. Hence, x and y keep confidential while revealing $r_1\alpha$ and $r_2\beta y$. Not to mention that $r_1r_2\alpha\beta$ and $r_1r_2\alpha\beta xy$ can only recover on the participant side with enough shares. In summary, adversary $\mathcal{A}$ is impossible to obtain additional information, which well-simulates $\mathcal{S}$ in the ideal world. Therefore, we can prove Π_{MSSMul} UC-realizes $\mathcal{F}_{MSSMul}$ with the universal composition theorem.

Functionality 3 $\mathcal{F}_{MSSAdd}$: $\mathcal{F}_{MSSAdd}$ interacts with CS_1, CS_2, P_i and $\mathcal{S}$.

1. Run $\mathcal{F}_{SD}$ to refresh $\langle x \rangle = ([\alpha], [\alpha x])$ and $\langle y \rangle = ([\beta], [\beta y])$.

2. Upon receiving a message $([r_1\alpha]_i)$ from P_i, send $([r_1\alpha]_i)$ to CS_1 and $\mathcal{S}$.

3. Upon receiving a message $([r_2\beta]_i)$ from P_i, send $([r_2\beta]_i)$ to CS_1 and $\mathcal{S}$.

4. Upon receiving a message $(r_1\alpha, r_2\beta)$ from CS_1, send the message $(r_1\alpha, r_2\beta)$ to CS_1, CS_2, P_i and $\mathcal{S}$.

Theorem 3 (Security of Π_{MSSAdd}). *Π_{MSSAdd} UC-realizes $\mathcal{F}_{MSSAdd}$ in the $(\mathcal{F}_{SD})$-hybrid model.*

Proof. Similar to Theorem 2, we define Π_{MSSAdd} in the $(\mathcal{F}_{SD})$-hybrid model. Since Π_{SD} randomly selects a and b, the real word and the ideal world are indistinguishable for $\mathcal{Z}$. Then, Π_{MSSAdd} can UC-emulate Π_{SD} in the $(\mathcal{F}_{SD})$-hybrid model. After that, we observe the views of CS_1 and CS_2, respectively. CS_1 obtains the refreshed masks, where $\mathsf{View}_{CS_1} = ([\alpha]_{pub}, [\beta]_{pub}, r_1\alpha, r_2\beta, [r_1r_2\alpha\beta]_{pub})$. CS_2 will learn nothing because the refreshed masked data remain as shares, where $\mathsf{View}_{CS_2} = ([\alpha x]_{pub}, [\beta y]_{pub}, [r_1\alpha x]_{pub}, [r_2\beta y]_{pub}, r_1\alpha, r_2\beta, [r_1r_2\alpha\beta(x+y)]_{pub})$. Since a part of the secret stays secure, performing calculations on shares will not disclose the real value in MSS. To conclude, adversary $\mathcal{A}$ learns nothing about the secret as long as the participant does not reach the threshold. As $\mathcal{A}$ perfectly simulates $\mathcal{S}$ in the ideal world. Therefore, we can show Π_{MSSAdd} UC-realizes $\mathcal{F}_{MSSAdd}$ in the universal composition theorem.

Functionality 4 $\mathcal{F}_{SC}$: $\mathcal{F}_{SC}$ interacts with CS_1, CS_2, P_i and an adversary $\mathcal{S}$.

1. Run $\mathcal{F}_{SD}$ to produce $\langle \ell \rangle$.
2. Run $\mathcal{F}_{MSSAdd}$ to compute $\langle z \rangle$.
3. Run $\mathcal{F}_{SD}$ to generate $\langle r \rangle$ and $\langle r' \rangle$.
4. Run $\mathcal{F}_{MSSMul}$ and $\mathcal{F}_{MSSAdd}$ to computes $\langle e \rangle = \langle z * r + r' \rangle$ and $\langle h \rangle = \langle \ell * r + r' \rangle$.
5. Upon receiving a message $(\langle e \rangle, \langle h \rangle)$ from CS_1 and CS_2, send the message $(\langle e \rangle, \langle h \rangle)$ to P_i and $\mathcal{S}$.
6. Upon receiving a message (*output*) from P_i, send the message (*output*) to CS_1, CS_2, P_i and $\mathcal{S}$.

Theorem 4 (Security of Π_{SC}). *Π_{SC} UC-realizes $\mathcal{F}_{SC}$ to complete secure comparison.*

Proof. Π_{SC} invokes secure protocols as subroutines, which we already prove UC-secure in Theorem 1, Theorem 2 and Theorem 3. Looking at the proving process above, we could find Π_{MSSMul} and Π_{MSSAdd} pick random numbers r_1 and r_2 for refreshing secrets through Π_{SD}. Since Π_{SD} also constructs from randomness parameters, Π_{SC} can UC-emulate these subroutines. Next, the view of adversary $\mathcal{A}$ is $(\langle x \rangle, \langle y \rangle, \langle \ell \rangle, \langle z \rangle, \langle r \rangle, \langle r' \rangle, e, h, output)$. Since multi-secret sharing carries on the information-theoretic security from secret sharing, $\mathcal{A}$ could reconstruct nothing in the view, and knowing e and h is helpless for getting more information. Because $\mathcal{Z}$ cannot obtain any clues to distinguish the real-world execution from the ideal-world execution, Π_{SC} securely realizes functionality $\mathcal{F}_{SC}$ for secure comparison. To sum up, we can demonstrate Π_{SC} UC-realizes $\mathcal{F}_{SC}$ when the subroutines interact as in the real-life model.

7.2 Security of kNN Classification

We have analyzed the composition security of the proposed protocols and guarantee that privacy will not leak during the persistent computations. Eventually, we take privacy-preserving kNN as an example and briefly describe the security of an MSS outsourcing service.

First, the phases of system initialization and data outsourcing are non-interactive, so we know that the system is safe through the distribution settings. Then, we upload the query data with SD protocol in the request generation phase. As the Π_{SD} is proved UC-secure, so is the data uploading. Finally, the classification consists of several persistent computations and numerous secure comparisons, where the protocols proved UC-secure in the above analysis. Afterward, we can consider an adversary $\mathcal{A}$ who colludes with participants less than t to get access to dataset D and query data x. Let π be the classification process, and function f represents the privacy leakages. We can get the probabilities below.

- $Pr[\mathcal{A}^{\pi}(D) \rightarrow f(x)] < \mathsf{negl}$
- $Pr[\mathcal{A}^{\pi}(x) \rightarrow f(D)] < \mathsf{negl}$

While the adversary colludes with someone who knows the dataset, they cannot learn the query data through the classification service. On the other hand, classifying the query data launched from the adversary will not cause privacy leakage of the outsourced dataset. Therefore, we can claim that the ciphertext does not reveal any information about the plaintext through the outsourcing service based on MSS protocols.

Chapter 8 Discussions

To evaluate our designed protocols, we theoretically analyze the cost of each function and measure the performance by massive persistent computations from kNN classification in the experiment.

8.1 Theoretical Analysis

We use (n, t) to denote the t-out-of-n multi-secret sharing, where n is the number of participants and t is the threshold ($t \leq n$) for each secret. In our scheme, CS_1, CS_2, and RG have the communication complexity of $O(n)$ through sending and receiving messages among n participants. Since there is no communication between participants, the communication cost of the participant is $O(1)$. For secure computations, each entity implies the operation by locally calculating their shares, so the participant has the computation cost of $O(1)$, and the cloud servers that modify $n - t + 1$ shares of a secret have the computational complexity of $O(n - t)$.

Assuming that we perform kNN classification of a single query to an outsourced dataset with v instances and m attributes, the storage complexity of each cloud server is $O(v \cdot m \cdot (n - t))$ and the storage overhead of the participant is always 1 in our system. Since we upload the query through SD protocol, RandRec adds about three times $2n - t + 1$ shares for each Beaver triple. Hence, the storage complexity of RG is $O(m \cdot n)$ in a request generation with m query data.

Table 8.1 is the complexity of construction for each protocol. While SC consists of three MSS share generations and several MSS operations, the complexities of SC are equal to the highest order of adding up the operations together. Accordingly, we can evaluate the classification by breaking the algorithm into two parts. The first part takes $O(v \cdot m)$ times to calculate the distance between the query and the dataset, mainly composed of SC, MSSMul, and MSSAdd. The second part uses $O(k \cdot v)$ times to compare the distance shares and find the k nearest labels through the SC protocol. At last, we can clean up the calculations that will no longer need after this classification, and the system can restore to the initial state.

Table 8.1: Complexities (computation, communication, storage) of each protocol. n: participants, t: thresholds

Protocol	P	CS_1	CS_2	RG
SD	$O(1)$, $O(1)$, N/A	$O(n-t)$, $O(1)$, N/A	$O(n-t)$, $O(1)$, N/A	$O(1)$, $O(n)$, $O(n)$
MSSMul	$O(1)$, $O(1)$, N/A	$O(n-t)$, $O(n)$, $O(1)$	$O(n-t)$, $O(n)$, N/A	$O(1)$, $O(n)$, $O(n)$
MSSAdd	$O(1)$, $O(1)$, N/A	$O(n-t)$, $O(n)$, $O(1)$	N/A, N/A, N/A	$O(1)$, $O(n)$, $O(n)$
SC	$O(1)$, $O(1)$, N/A	$O(n-t)$, $O(n)$, $O(1)$	$O(n-t)$, $O(n)$, N/A	$O(1)$, $O(n)$, $O(n)$

8.2 Experimental Results

We set up the experiments on a desktop computer with Windows 11 Pro operating system, Intel(R) Core(TM) i7-8700 3.20GHz CPU, and 16GB memory as specification. Then, we realize our scheme with python 3.9.2 and simulate all actions of the privacy-preserving kNN classification service on several datasets. We point out our work applies to various tasks by choosing datasets with different backgrounds from the UCI Machine Learning Repository. More specifically, we take iris [1] to test pattern recognition, bankruptcy [2] to build an expert decision, and banknote [3] to authenticate from extracted features of images. Besides, we show the possibility of our scheme in some industries through tic-tac-toe [4], car [5], and breast cancer [6].

In the implementation, we pick a large prime greater than any secret of the dataset to fulfill the MSS in our system. Then, we randomly sample 10 test cases for requests and evaluate the average with epochs of 10. Figure 8.1 shows the accuracy comparison between the ordinary kNN and our outsourced kNN. Thus, we can prove

[1] https://archive.ics.uci.edu/ml/datasets/Iris
[2] https://archive.ics.uci.edu/ml/datasets/Qualitative_Bankruptcy
[3] https://archive.ics.uci.edu/ml/datasets/banknote+authentication
[4] https://archive.ics.uci.edu/ml/datasets/Tic-Tac-Toe+Endgame
[5] https://archive.ics.uci.edu/ml/datasets/Car+Evaluation
[6] https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

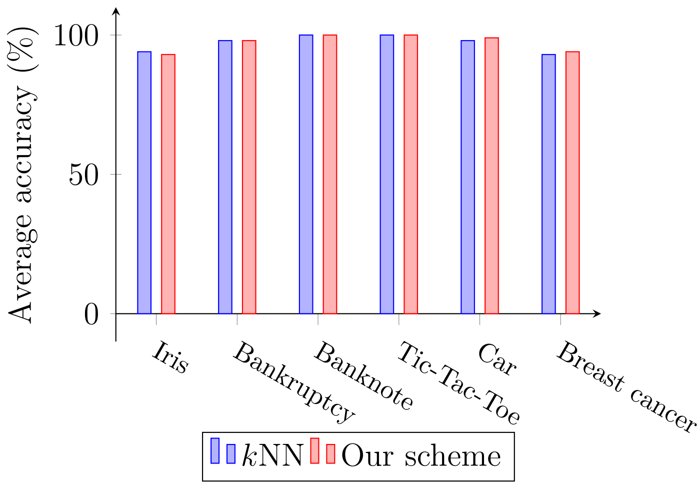


Figure 8.1: Average accuracy in different datasets.

the applicability by highlighting the close accuracy of the outsourced service based on our scheme. Figure 8.2 and Figure 8.3 demonstrate the performance under different settings. Looking up to Table 8.2, we can observe the execution time mainly grows with the number of participants and the total data of the dataset. Although the execution time increases with the size of the participants, performance remains the same on the varying threshold. Therefore, there is no trade-off between security and efficiency as we raise the threshold to form a more secure system that requires higher access rights.

Table 8.2: Parameters of each dataset.

Dataset	Instances	Attributes	Classes	Total Data
Iris	150	4	3	600
Bankruptcy	250	6	2	1500
Banknote	1372	4	2	5488
Tic-Tac-Toe	958	9	2	8622
Car	1728	6	4	10368
Breast cancer	569	30	2	17070

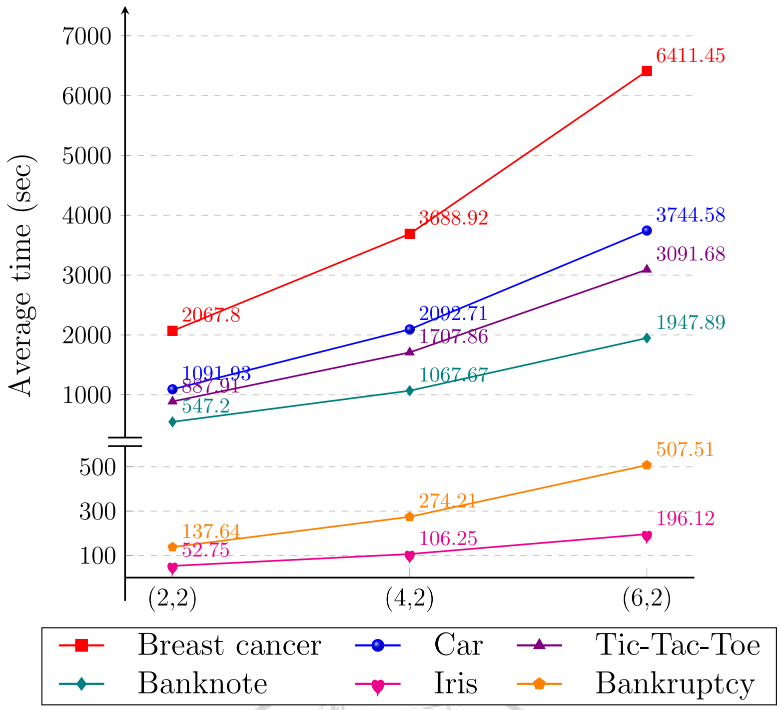


Figure 8.2: Average execution time with different numbers of participants.

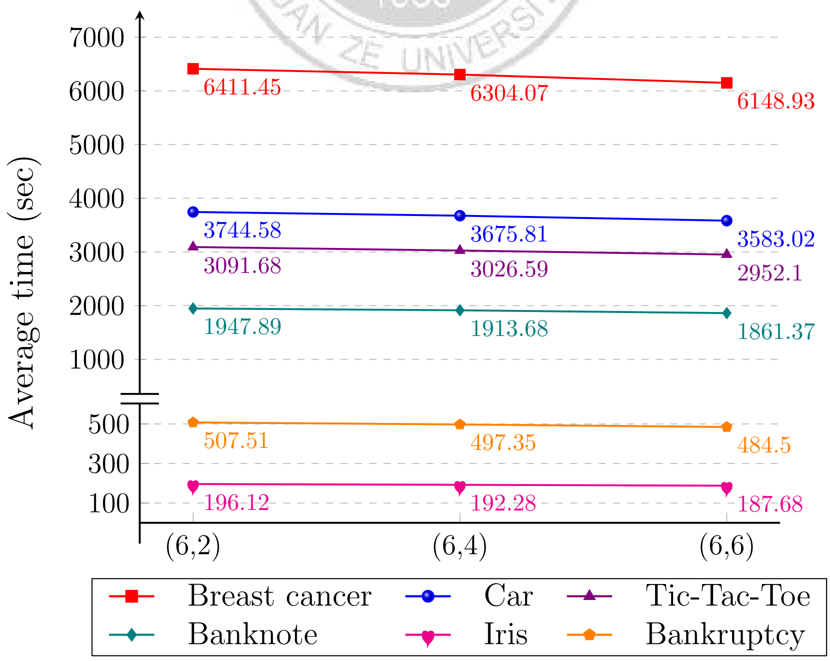


Figure 8.3: Average execution time with varying thresholds.

Chapter 9 Conclusions

In this thesis, we proposed a persistent computing architecture based on multi-secret sharing and constructed some privacy-preserving operation protocols above it. Our scheme can achieve cloud assistance multi-secret sharing (CAMSS) by exchanging the storage costs with communication and computation, which helps to form a secure and efficient computing pattern on resource-restricted devices. We ensure computational feasibility and security for this form and build a kNN classification to evaluate the performance. In the future, we would like to extend this research to more applications, discuss and address particular security issues, and create other possible operational protocols.

References

[1] D. Richins, D. Doshi, M. Blackmore, A. Thulaseedharan Nair, N. Pathapati, A. Patel, B. Daguman, D. Dobrijalowski, R. Illikkal, K. Long, D. Zimmerman, and V. Janapa Reddi, "Missing the forest for the trees: End-to-end ai application performance in edge data centers," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2020, pp. 515–528.

[2] M. Hind, D. Wei, M. Campbell, N. C. F. Codella, A. Dhurandhar, A. Mojsilović, K. Natesan Ramamurthy, and K. R. Varshney, "Ted: Teaching ai to explain its decisions," in *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, ser. AIES '19, 2019, p. 123–129.

[3] H. W. Lim, G. S. Poh, J. Xu, and V. Chittawar, "PrivateLink : Privacy-preserving integration and sharing of datasets," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 564–577, 2019.

[4] Q. Xue, Y. Zhu, J. Wang, and X. Li, "Distributed set intersection and union with local differential privacy," in *2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS)*, 2017, pp. 198–205.

[5] V. M. Shelake and N. M. Shekokar, "Smspprl: A similarity matching strategy for privacy preserving record linkage," in *2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, 2020, pp. 481–485.

[6] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[7] M. Franklin and M. Yung, "Communication complexity of secure computation (extended abstract)," in *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, ser. STOC '92. New York, NY, USA: Association for Computing Machinery, 1992, p. 699–710. [Online]. Available: https://doi.org/10.1145/129712.129780

[8] D. Chen, W. Lu, W. Xing, and N. Wang, “An efficient verifiable threshold multi-secret sharing scheme with different stages,” *IEEE Access*, vol. 7, pp. 107 104–107 110, 2019.

[9] A. Wigderson, M. Or, and S. Goldwasser, “Completeness theorems for noncryptographic fault-tolerant distributed computations,” in *Proceedings of the 20th Annual Symposium on the Theory of Computing (STOC’88)*, 1988, pp. 1–10.

[10] T. Shingu., K. Iwaumura., and K. Kaneda., “Secrecy computation without changing polynomial degree in shamir’s (k, n) secret sharing scheme,” in *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications - DCNET, (ICETE 2016)*, 2016, pp. 89–94.

[11] H. Kaur and P. Khanna, “Privacy preserving remote multi-server biometric authentication using cancelable biometrics and secret sharing,” *Future Generation Computer Systems*, vol. 102, pp. 30–41, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18330553

[12] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, “Verifiable secret sharing and achieving simultaneity in the presence of faults,” in *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*. IEEE, 1985, pp. 383–395.

[13] M. Stadler, “Publicly verifiable secret sharing,” in *Advances in Cryptology — EUROCRYPT ’96*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 190–199.

[14] V. Attasena, J. Darmont, and N. Harbi, “Secret sharing for cloud data security: a survey,” *The VLDB Journal*, vol. 26, no. 5, pp. 657–681, Oct 2017. [Online]. Available: https://doi.org/10.1007/s00778-017-0470-9

[15] A. Nag, S. Choudhary, S. Dawn, and S. Basu, “Secure data outsourcing in the cloud using multi-secret sharing scheme (msss),” in *Proceedings of the First International Conference on Intelligent Computing and Communication*, J. K. Mandal, S. C. Satapathy, M. K. Sanyal, and V. Bhateja, Eds. Singapore: Springer Singapore, 2017, pp. 337–343.

[16] C. Lin, H. Hu, C.-C. Chang, and S. Tang, “A publicly verifiable multi-secret sharing scheme with outsourcing secret reconstruction,” *IEEE Access*, vol. 6, pp. 70 666–70 673, 2018.

[17] J.-K. Yang, K.-C. Huang, C.-Y. Chung, Y.-C. Chen, and T.-W. Wu, “Efficient privacy preserving nearest neighboring classification from tree structures and secret sharing,” in *ICC 2022 - IEEE International Conference on Communications*, 2022, pp. 5615–5620.

[18] D. Beaver, “Efficient multiparty protocols using circuit randomization,” in *Advances in Cryptology — CRYPTO ’91*, J. Feigenbaum, Ed., 1992, pp. 420–432.

[19] P. Mohassel and Y. Zhang, “Secureml: A system for scalable privacy-preserving machine learning,” in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 19–38.

[20] R. Canetti, “Universally composable security: a new paradigm for cryptographic protocols,” in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, 2001, pp. 136–145.