DISTRIBUTED TRUST EVALUATION PROTOCOL AND SECURE DATA QUERY SCHEMES FOR INTERCLOUD

DOU YI

PhD

The Hong Kong Polytechnic University

2018

The Hong Kong Polytechnic University

Department of Computing

Distributed Trust Evaluation Protocol and Secure Data Query Schemes for Intercloud

Dou Yi

A thesis
submitted in partial fulfilment of the requirements
for the degree of

Doctor of Philosophy

June 2018

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

______________________________ (Signed)

Dou Yi (Name of student)

Abstract

The aim of Intercloud is to facilitate the sharing of data and cloud resources so that more co-operative cloud services can be provided. In this thesis, we investigate two important security issues for supporting Intercloud, namely distributed trust evaluation and secure data query. *In the first part of the thesis*, we present a distributed trust evaluation protocol with privacy protection for Intercloud. First, feedback privacy is protected by homomorphic encryption with verifiable secret sharing. Second, to cater for the dynamic nature of Intercloud, trust evaluation can be conducted in a distributed manner and is functional even when some of the parties are offline. Third, to facilitate customized trust evaluation, an innovative mechanism is used to store feedback, such that it can be processed flexibly while protecting feedback privacy. The protocol has been proved based on a formal security model. Simulations have been performed to demonstrate the effectiveness of the protocol. *In the second part of the thesis*, we design and evaluate a privacy-preserving range query scheme for cloud storage, which can protect the privacy of record and range queries. During range comparison, our scheme neither leaks the order relationship between the upper/lower bound of a range query and the encrypted index, nor produces false positives in the query results. The experimental result indicates that our scheme can achieve higher security while maintaining good efficiency. *In the third part of the thesis*, we investigate another secure data query issue, which is about access pattern leakage attack on searchable encryption under an Intercloud environment. Basically, both records and queries are distributed among servers of different cloud service providers, so that each cloud server can only have partial information about queries and their results. To minimize the query response time while protecting information disclosure, we formulate the record and query assignment as an optimization problem, and solve the problem (i.e., finding the best possible solution) by the minimum cut algorithm. Numerical results show that certain access pattern information can be saved by our assignment strategy while maintaining good query response time.

Acknowledgements

First and foremost, I would like to express my heartfelt appreciation and acknowledgement to my supervisor, Dr. Henry C. B. Chan, for offering me this great opportunity of a research study in The Hong Kong Polytechnic University. His critical assessment and appreciation of my work make me on the right track to finish my PhD study. His patience and tremendous inspirations give me confidence in overcoming difficulties of research problems. Next, I would like to convey my sincere gratitude to Dr. Man Ho Allen Au, for his valuable advice on security analysis and proof techniques.

I would like to give my thanks to my previous and current office mates and group mates. Every discussion with them can directly or indirectly help me to finish my PhD study no matter the research problem or coursework. I am also grateful to my friends in Hong Kong for their emotional support, sharing, encouragement, and company.

I would like to dedicate this thesis to all people who have guided, instructed, or helped me with this research study. This study cannot be accomplished without all of your kind support. The most importantly, I would like to express my special thank to my parents for their encouragements and love. The thesis is dedicated to them all.

Table of Contents

List of Figures

List of Tables

List of Acronyms

AES	Advanced Encryption Standard
APIs	Application Program Interfaces
ASS	Additive Secret Sharing
AWS	Amazon Web Services
CA	Certification Authority
CE	Comparable Encryption
CSP	Cloud Service Provider
DNS	Domain Name System
FBS	Distributed Feedback Storage
FHE	Fully Homomorphic Encryption
GMP	Multiple Precision Arithmetic Library
GRASP	Greedy Randomized Adaptive Local Search Procedure
HbbTV	Hybrid broadcast broadband TV
IaaS	Infrastructure as a Service
IEEE-SA	IEEE Standards Association
IND-CKA2	Indistinguishability against Adaptive Chosen Keyword Attacks
NoSQL	Not Only SQL
OPE	Order Preserving Encryption
ORAM	Oblivious Random Access Memory
ORE	Order Revealing Encryption
PBtree	Privacy Bloom Filter Tree
PKI	Public Key Infrastructure
PPE	Prefix-Preserving Encryption

PRF	Pseudo Random Function
QoS	Quality of Service
RSSE	Range Searchable Symmetric Encryption
SaaS	Software as a Service
SAML	Security Assertion Markup Language
SCAP	Successful Collusion Attack Probability
SFRR	Successful Feedback Recovery Rate
SLA	Service Level Agreement
SPKI	Simple Public Key Infrastructure
SSA	Secret Sharing Address
SSE	Searchable Symmetric Encryption
TDAG	Tree-like Directed Acyclic Graph

List of Major Symbols

$\mathcal{A}$	Adversary controls dishonest participants in the security analysis
$\mathcal{A}_R$	Assignment of records to a set of cloud servers
$\mathcal{A}_Q$	Assignment of queries to a set of cloud servers
$\mathcal{C}$	A challenger responds to the adversary in the security analysis
$\mathcal{C}_{sum}$	Encrypted feedback sum of raters in the same set
$\mathcal{D}$	A collection of attribute values to be queried
d	An attribute value
f	Feedback given by rater to a cloud service provider
G	A directed graph to describe the connections of Intercloud
$\mathcal{H}$	Honest participants in the security analysis
H	Hash function
$\mathcal{I}$	Encrypted searchable indexes of attribute values
I_d	Index of attribute value d
k	Secret key of a rater for feedback encryption
$\mathcal{M}$	Dishonest participants controlled by adversary
m	Minimum number of raters for secret key sum reconstruction
n	Number of raters in the same set
params	System parameter
$privK$	Private key of a rater
$pubK$	Public key of a rater
P_s	Prefix string set of any numerical value s
Q	A range query requested from users

$\mathcal{Q}$	A sequence of queries sent by users
r	Rater who gives feedback to a cloud service provider
$\mathbb{R}$	Set of raters of a cloud service provider
$\mathcal{R}$	A set of database records with several attributes
$\mathcal{R}_{sum}$	Feedback sum of raters in the same set
sk	Secret key used to generate the index and trapdoor
$\mathcal{S}$	A set of cloud servers
S_s^0	0-encoding set of any numerical value s
S_s^1	1-encoding set of any numerical value s
$\widetilde{S_s^1}$	New 1-encoding set of any numerical value s
$\widetilde{Sign}$	Blind signature
$\mathcal{T}$	Trusted third party in the security analysis
$\mathcal{T}r$	Trust evaluation result
T_Q	Secure trapdoor of a query
u	User of a cloud service provider
$\mathbb{U}$	Set of users of a cloud service provider
w_L	Lower bound of a range query
w_H	Upper bound of a range query
z	Total rating choice level of satisfaction with the cloud service
λ	Security parameter
ℓ	Length binary strings for an attribute value
ϖ, σ	Random nonces
θ	Mean query arrival rate of a cloud server
μ	Mean query process rate of a cloud server

1 Introduction

With the rapid advancement of cloud computing, there is an increasing number of cloud services. Each provides different service qualities, pricing and access strategies. In the conventional cloud computing environment, once a cloud user decides to select a cloud service, it is difficult and costly to switch to a new cloud service provider. To address this vendor lock-in problem and to support more cooperative cloud services, Intercloud has been proposed [1, 2, 3, 4]. Basically, Intercloud can be implemented from two perspectives. The first one is provider-centric, where cloud service providers directly interact with each other for resource/data sharing and cooperative operation. For instance, cloud service providers can process user requests by leveraging services from other clouds [5, 6]. The infrastructure of different cloud service providers can be shared to improve overall resource utilization [7, 8]. The second one is user-centric, where cloud users deploy their applications among multiple clouds for various purposes. For instance, applications can be migrated from one cloud service provider to another cloud service provider [9, 10] and workloads can be distributed among clouds for disaster recovery or multi-region application delivery [11, 12]. In this thesis, we investigate two important security issues for supporting Intercloud, namely distributed trust evaluation and secure data query.

From the provider-centric perspective, a cloud service provider typically trusts another cloud service provider based on certain trust attributes, such as service reliability, quality of service and service efficiency. In the Intercloud environment, one cloud may want to select a number of reliable clouds to help run a time-consuming program. For mobile Intercloud, a mobile user may want to select a cost-effective cloud service in a foreign city. The trustworthiness of cloud services is an important consideration for making cloud selection decision (i.e., knowing the expected performance of a cloud service). Trust evaluation is often conducted based on existing ratings or feedbacks (i.e., reputation-based trust evaluation) before choosing/using a service. Note that for Intercloud, the environment is highly dynamic and distributed, and relationships can be one-way or two-way (i.e., clouds provide services to each other). There is more and more mutual co-operation in the Intercloud, a cloud user or his/her business could be another type of cloud service provider in future business transactions. This possible mutual relationship makes the privacy requirement even more important in the Intercloud scenario. If feedback information cannot be made private, cloud users may only give positive feedback, as they want to maintain a good relationship or are fearful of retaliation [13, 14]. Hence, it is important to develop an effective and flexible trust evaluation protocol with privacy protection for Intercloud. Currently, there has been little work done to study trust evaluation for the Intercloud environment satisfying these requirements. Chapter 3 of this thesis seeks to contribute to this important topic for the development of Intercloud.

From the cloud user-centric perspective, data can be stored, accessed and processed flexibly through the cloud for various applications, such as financial services, public health services, and traffic services. To use a cloud data storage service, data owners need to upload data to the cloud provider. Hence, data privacy and security are two key concerns. Instead of storing plaintext data, encrypted data can be stored in order to protect data privacy, but it is neither convenient nor efficient to process encrypted data (e.g., data searching). When someone wants to search for the encrypted data using certain query conditions, he/she needs to download the encrypted data from the cloud, decrypt them and process the query locally. This method is obviously not desirable in terms of efficiency and energy usage. It is also not practical when the terminal's processing power or network bandwidth is limited, such as when processing data through a mobile phone. Searchable symmetric encryption (SSE) is a promising technique to tackle the aforementioned problem [15]. Most current SSE schemes focus on handling keyword search over encrypted data [16, 17, 18, 19, 20, 21]. However, they do not address the problem of range query over encrypted data which are required for some applications. Existing range query over encrypted data schemes either provide faster search time at the expense of disclosing sensitive information or achieve higher security at the expense of extra cost. In Chapter 4 of this thesis, we design and evaluate a privacy-preserving range query scheme to address the above-mentioned limitations.

Searchable symmetric encryption (SSE) allows database searching to be conducted over encrypted data, which is particularly useful in the cloud environment. Basically, each index is computed based on a keyword to be queried. To allow the cloud server to do searching itself, trapdoors are securely generated for queries based on query keywords and query operators. During the query phase, the cloud server matches the trapdoor with the indexes to target the satisfied documents and returns the encrypted documents as the query results to the data user. As a result, the cloud server obtains a collection of indexes, a sequence of trapdoors and encrypted query results of each trapdoor. The cloud server can learn the search pattern (i.e., if the query has been issued before) and access pattern (i.e., which encrypted documents satisfied which trapdoors). Based on these observations, the cloud server still can estimate the plaintext value of documents or query even without decryption keys. Oblivious Random Access Memory (ORAM) [22, 23] addresses the access pattern leakage problem. However, the computational complexity of most ORAM schemes is high and they can only support limited types of queries. Vertical fragmentation [24] seeks to make attackers unable to discover the sensitive association between record attributes. However, this technique cannot prevent the access pattern leakage attacks on a single attribute, and can lead to a long response time for multi-keyword search. In Chapter 5 of this thesis, we study the problem of access pattern leakage attack on searchable encryption under a multi-cloud environment. Basically, both database records and queries are distributed among different cloud servers, so that each cloud server can only have partial information about queries and their results.

1.1 Research Objectives

In this thesis, we investigate two important security issues for supporting Intercloud, namely distributed trust evaluation and secure data query. In general, the main research objectives are as follows:

- To design a distributed trust evaluation protocol with privacy protection for Intercloud. In the first problem, we consider a reputation-based trust evaluation system for Intercloud. The key design goal of the protocol is to protect user anonymity and confidentiality of feedbacks in a distributed and secure manner. Furthermore, the protocol should support customized processing of evaluation results and still function well even when some raters are malicious or offline.

- To design and evaluate an order-hiding range query over encrypted data without search pattern leakage for cloud storage. In the second problem, we mainly focus on the following scenario. A data owner would like to store some sensitive records from one cloud to another cloud. Before storing the records to the second cloud, each record is encrypted. If required (e.g., due to data loss on the first cloud), data can be recovered efficiently and securely. The main challenges are to solve the security leakage problem in existing secure range query schemes (i.e., statistical relationships among indexes, comparison operator, and search pattern of queries) and achieve a better balance between security and efficiency than the existing schemes.

- To investigate and design an access pattern hidden query over encrypted data scheme. In the third problem, we consider the following scenario. In order to achieve faster response time and avoid the vendor lock-in risk, developers migrate applications to multiple clouds at the same time. Leveraging this Intercloud deployment, our aim is to reduce the access pattern disclosure of the existing searchable symmetric encryption schemes. As a result, cloud servers cannot obtain sufficient statistical information from the observed query results while maintaining query efficiency.

1.2 Original Contributions

Corresponding to the research objectives, the main original contributions made in this thesis are summarized as follows:

- We designed a distributed trust evaluation protocol with privacy protection for Intercloud. To encourage honest evaluation and prevent possible retaliatory attacks, our protocol contributes to the Intercloud trust evaluation with feedback privacy and anonymity protection by employing homomorphic encryption with verifiable secret sharing. As a result, the corresponding cloud service provider cannot access the individual feedback. The trust evaluation results are computed as a weighted mean of feedbacks. In addition, our protocol can still function if certain users are offline (i.e., not available) as well.

- We designed a privacy-preserving range query scheme that can hide query search patterns and is also secure against inference attacks. The index generation algorithm in our scheme is non-deterministic which can prevent the inferring order of records based on the corresponding indexes. The range comparison method developed in our scheme does not reveal the binary bit that differs between a record and a query which avoids the order leakage of records during the comparison. Our scheme also generates the trapdoor in a non-deterministic manner which can hide the query search pattern. A security proof has been conducted properly to show that our scheme is more secure than existing schemes. The experimental results indicate that our scheme has a shorter index size and search time than the existing schemes when the processing unit is large.

- We designed a security scheme that distributes database and queries among multiple cloud servers. Since none of the cloud servers have the entire database nor the full picture of queries, the database and user queries are protected from statistical analysis and access pattern leakage attacks. To minimize the query response time while preventing information disclosure, we formulated the record and query assignment as an optimization problem, and solved the problem (i.e., finding the best possible solution) by the minimum $s - t$ cut algorithm. Experimental results indicate that our strategy can reduce the access pattern leakage without sacrificing query efficiency.

1.3 Thesis Structure

Chapter 2

This chapter aims to conduct a comprehensive background and literature review of related topics for the research. The related topics include: overview of Intercloud, trust evaluation related work, privacy-preserving range query related work, and access pattern leakage of searchable encryption and related countermeasures.

Chapter 3

This chapter presents the first part of this research: a distributed trust evaluation protocol with privacy protection for Intercloud. First, this chapter presents the models and overview. Second, this chapter explains the feedback computation. Third, it discusses the trust evaluation protocol. The forth part of this chapter presents the security proof and discusses the security analysis. In the last two parts of this chapter, it presents the simulation settings and discusses the simulation results.

Chapter 4

In Chapter 4, we design and evaluate a search pattern hidden range query over encrypted data scheme for cloud storage. The first part of this chapter provides the scheme's general construction and security goal. Second, it describes the building block utilised in the scheme. Third, this chapter presents the details of our proposed privacy-preserving range query scheme. Experimental setup and results are illustrated in the forth part of this chapter. The last part of this chapter analyses the security of the scheme and proves that it can achieve the defined security goals.

Chapter 5

This chapter discusses the third part of this research: the access pattern leakage of query over encrypted data scheme. The first part of this chapter formulates our proposed record and query assignment strategies to prevent the access pattern leakage during the query over encrypted data. The second part of this chapter describes how to search for an optimal assignment strategy by solving the assignment optimization problem. The experimental results and conclusion are shown in the third and forth part of this chapter, respectively.

Chapter 6

This final chapter presents the overall conclusion of this thesis and outlines future work.

2 Literature Review

In this chapter, we conduct a literature review of the following topics: overview of Intercloud, related work on trust evaluation, related work on secure range query over encrypted data, and and access pattern leakage attack to searchable encryption and related countermeasures.

2.1 Overview of Intercloud

In this section we give an overview of Intercloud including the motivations and basic architecture. As cloud computing technology continues to mature, an increasing number of people and companies have used cloud services. More applications require data distributed on more than one cloud [28]. However, cloud outages occurred frequently these years, seriously affecting cloud users [29]. The high price of transferring data out of clouds prevent users from changing the cloud service providers. Furthermore, the differences between cloud service application program interfaces (APIs) also impede the cooperation between different clouds and their users. To address these problems, the need for cloud interoperation has been raised. In recent years, Intercloud has been proposed to facilitate cloud interoperation [30, 31, 4]. The purpose of Intercloud is to make the clouds functioning like the Internet and telephone networks, where resources and services of different cloud service providers can be shared. In the Intercloud, a cloud can collaborate with other clouds to serve user requests. Hence, cloud resources can be utilized in a collaborative manner. It has the potential to create new business opportunities through the extension of single cloud services. We consider that Intercloud can provide benefits for all cloud service providers. While the existing commercial cloud services are typically based on a business-to-consumer model, Intercloud can be viewed as a business-to-business model. With Intercloud, cloud service providers can fulfill the users' demand more cost-effectively by leveraging other clouds' services (i.e., instead of developing the services locally [5, 32]). For instance, even in a region without its own service, a cloud provider can still serve its customers through another cloud service provider. Clouds providing different services can collaborate to enhance customer service and loyalty. For a private cloud environment, Intercloud can enhance cloud resources utilization through the interconnection of local infrastructures. For example, [8, 33] studied cloud federation for scientific applications, such as EnergyPlus [34], Octave [35] and DII-HEP [36]. Specifically, by distributing workloads through cloud federation, datacenter capacity and processing speed can be enhanced in a flexible manner, resulting in better performance and lower cost [33].

Recent Intercloud research has mainly focused on resource management (e.g., an adaptive cloud resource allocation scheme [37], a simulation framework for Intercloud job scheduling [7]), data and virtual machine migration (e.g., a software defined network-based Intercloud virtual machine migration scheme [9], a decision-making model for Intercloud migration from the user perspective [10]), service integration (e.g., an algorithm for selecting and composing services among multiple clouds [6], a mechanism to interconnect virtual machines at different clouds [8]) and Intercloud security (e.g., a single sign-on authentication scheme for clouds to inter-operate with one another [38], an authentication mechanism for components running at different clouds [39]). The limitations of only relying on single public or private cloud are discussed as follows.

2.1.1 Motivations of Intercloud

The following summarizes the motivations of Intercloud.

- **Avoiding Vendor Lock-in**

  Currently, each cloud service provider only offers its specific and various cloud services and APIs to the cloud users based on their own definitions and naming systems. Cloud users need to tailor their applications to fit different cloud interfaces, which results in considerable technical efforts in the future migration. Furthermore, cloud providers usually set higher prices for data transfer out of the cloud data centres, which ensures their customers depend on their services. Thus, applications or data are restricted within a particular cloud provider. Once the cloud user uploads sufficient amount of workloads to the cloud, it is hard to relocate their digital properties out of the current cloud cost-effectively. Therefore, uniform interfaces among different clouds facilitate cloud users to migrate their workloads whenever they are negatively impacted by the current cloud. Thus, cloud users can avoid vendor lock-in. At the same time, every cloud provider wants to grab business from competitors, so that they would like to build tools for customers to migrate from other clouds.

- **High Availability and Disaster Recovery**

  While clouds have various internal and external guarantee mechanisms to ensure the high availability of their services, unexpected failures still occur frequently these years. A cloud outage seriously affects cloud users, especially for users fully hosting their servers on one cloud. Apart from cloud outages, there are also other problems when using virtual machines. For instance, virtual machines may become unresponsive or unreachable due to the virtual machine monitor failure [40] or unexpected restart of the virtual machine caused by cloud environment updating [41]. These problems cause the downtime of applications hosted by virtual machines, which is not acceptable for application developers with high availability requirements.

- **Diverse Geographic Distribution**

  Although leading commercial public clouds offer services at the world's major regions, there are many regions not covered by the datacenters of these clouds. When a cloud user needs on-demand infrastructures to develop an application for geographically dispersed users and unclear usage pattern, relying on a sole cloud provider is infeasible. In order to achieve faster response time, developers often migrate applications to multiple clouds at the same time, so as to deploy server as close to the users as possible. In addition, cloud users from some countries or regions have legislative regulations about where to host their data and applications. Hence, cloud interconnection allows the developers to migrate applications to any local cloud provider to comply with the local legislation requirements.

- **Scalability and Wider Resource Availability**

  Since the resource usage pattern of applications vary from time to time, it is hard to accurately predict the potential cloud resources usage volume. When there are immediately surge demands or temporarily resources requirements, the current solution is to over-provision the private cloud datacenter capacity. Finally, the total capacity is several times larger than the average resources demands which incurs extra expense. A feasible solution to this peak-load issue is to rely on idle resources on the other cloud service providers. The small size private cloud migrates workloads to public clouds or its federated peer private clouds for resources sharing.

Hence, instead of maintaining additional servers in the local datacenter, the interconnection between multiple clouds allows small size private cloud to expand its IT environment in a flexible way and save a substantial amount of money.

To summarize, the main value of cloud interconnection or Intercloud is to allow cloud users to expand their cloud hosted businesses in both of cloud providers and geographic locations. Thus, they could make have a more efficient deployment strategy. At the same time, cloud providers also have great incentives to join the Intercloud. Cloud provider can scale up or down their data center by offloading workloads to other clouds so that it can provide better services and support more demands of its users.

2.1.2 Architecture of Intercloud

The concept of Intercloud ("cloud of clouds") is an analogy with the Internet ("network of networks"). Intercloud is viewed as the final goal of cloud interoperation, where multiple cloud infrastructures are ubiquitously interconnected based on a unified standard [31]. From the cloud provider's perspective, each cloud can act as another cloud user having the permission to migrate their workload, access to the computational and storage resources or services from its peer clouds. From the cloud user's perspective, Intercloud allows the interoperability between user's paid resources and services on multiple clouds [42]. At present, there is no unified definition of Intercloud.

The existing definitions of Intercloud are too generic and none of them clearly define who initiates the Intercloud connections. And Intercloud essentially is the interconnection of multiple cloud resources. On one hand, a cloud provider connects its own resources to other clouds' infrastructures. On the other hand, a cloud user connects its paid resources on different clouds. Hence, there are two perspectives to realize the vision of Intercloud, one is driven by *cloud providers*, the other is driven by *cloud users*.

Provider-centric Intercloud

In the provider-centric Intercloud model, the cloud providers voluntarily cooperate with each others by adopting the same standard interfaces. We define this collaboration as provider-centric Intercloud. It is usually a static collaboration where a prior agreement needs to be achieved before the establishment of the federation. Fig.2.1 describes the vertical and horizontal federation for provider-centric scenarios.

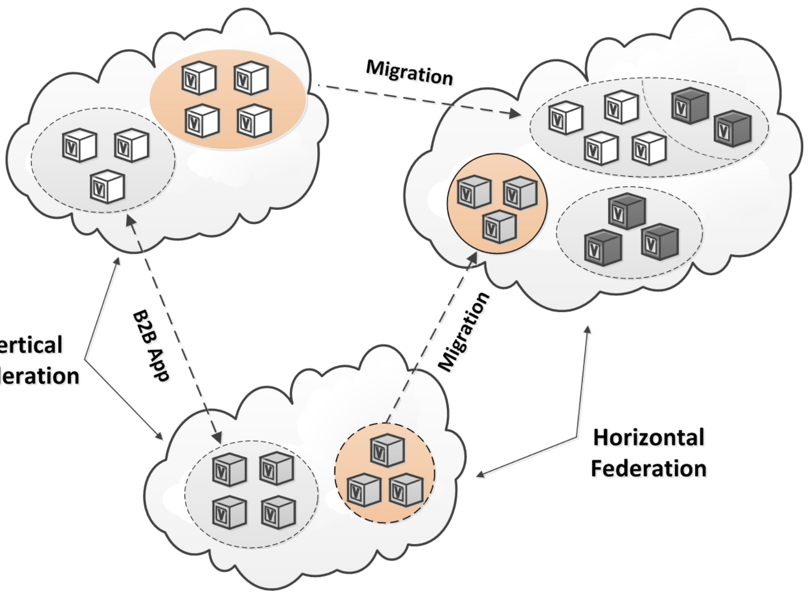


Fig. 2.1: Provider-centric Intercloud.

- **Vertical federation**

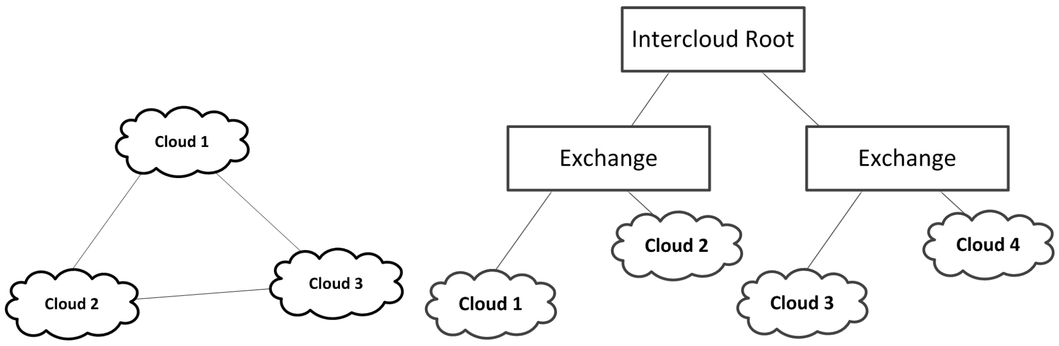
  Vertical federation is similar with business-to-business scenarios, in which some clouds meet the demand of its own customers by leveraging services from other clouds [42]. Vertical federation can be formed between both of public and private clouds, as long as they have common interests.

Cooperation projects with privacy requirements can be carried out by vertical federation between private clouds, such as scientific research applications. Taking the advantages of complementary services of different clouds, the vertical federation can enlarge the capability of each cloud in the federation to meet better Quality of Service (QoS) target. For instance, Philips is a Software as a Service (SaaS) provider supplying rapid cloud-based healthcare data recovery services. It collaborates with Amazon Web Services (AWS) which is an Infrastructure as a Service (IaaS) provider. By utilizing AWS Import/Export Snowball, Philips improves its own user experience [43].

- **Horizontal federation**

  Horizontal federation happens at the same layer of cloud providers, such as IaaS layer. The main purpose of the horizontal federation is for resources sharing so that clouds with surplus resources can be used by resources constrained clouds during its peak load times. A cloud provider has administrator privileges to scale or transit part of the workloads in the datacenter to other members of the federation. At the present stage, horizontal federation is less likely to happen between commercial public clouds. Since leading commercial cloud providers selling similar services are competitors, none of them have intentions to utilize competitors' resources to solve its own problem. In contrast, multiple private clouds belonging to the same organization are more likely to join the federation for resource sharing. They do not have competition and horizontal federation achieves higher overall resource utilization.

Horizontal and vertical federation classify cloud federation based on whether cloud providers are from the same cloud service layer. When it comes to how different clouds network with each other, there are two networking topologies: Peer-to-Peer and Hierarchical federation as shown in Fig.2.2.



(a) Peer-to-Peer Federation (b) Hierarchical Federation.

Fig. 2.2: Cloud federation architectures

- **Peer-to-Peer Networking**

  All the cloud providers directly interconnect and negotiate with each other as a mesh of clouds without middle layers [31] as depicted in Fig.2.2a.

- **Hierarchical Networking**

  Bernstein et al. proposed the first blueprint of Intercloud [30]. Subsequently, an IEEE Standards Association (IEEE-SA) P2302 Working Group was formed to develop the standards for Intercloud [44]. In parallel to the P2302 Working Group, an IEEE Intercloud Testbed Project was established to develop, prove, and improve vendor-neutral, open source, Intercloud technology globally [45]. The Intercloud standard focuses on the communication protocols and networking mechanisms for Intercloud, such as cloud providers' resources discovery, selection and allocation problem in the Intercloud. To establish the Intercloud networking structure, there are three main entities defined in the standard: Intercloud Roots, Intercloud Exchanges, and Intercloud Gateways. Different clouds can be connected with each other via an Intercloud gateway and there are Intercloud exchanges and a root in the Intercloud architecture. Intercloud exchange acts as a middle layer between Intercloud gateway and Intercloud root. Fig.2.2b shows the architecture of the hierarchical federation.

- *Intercloud Root* is comprised a set of root servers. The servers are similar to the Domain Name System (DNS) root. Intercloud Root is responsible for providing cloud resources directory services by hosting and managing a cloud resource catalog. Also, Intercloud Root provides authentication services by acting as a certificate authority to issue and verify certificates of different entities. In addition, an Intercloud Root instant is the parent of a set of Intercloud Exchanges. Intercloud Root connects and works with Intercloud Exchanges to facilitate Intercloud operations.

- *Intercloud Exchanges* are the second level servers which operate under Intercloud Root. One of the main functions of Intercloud Exchanges is to facilitate the process of resources provisioning and matching. The ontology information is copied from Intercloud Root instance and store in the Intercloud Exchanges. Intercloud Exchanges are responsible for mediating the communication between clouds and Intercloud Root.

- *Intercloud Gateways* are responsible for mediating the negotiation dialogs between cloud entities. There is an Intercloud Gateway for every cloud entities such as Intercloud Root, Intercloud Exchange and Cloud Provider. The role of Intercloud Gateways is to provide an interface for heterogeneous cloud environment to communicate effectively and securely. A standardized protocol is used for the negotiation process among clouds.

User-centric Intercloud

Even if cloud providers do not support cloud interconnection, cloud users are still able to gain the benefits from multiple clouds, which is called user-centric Intercloud. In the user-centric Intercloud model, the communication between cloud providers are not directly established and does not require cloud providers to adopt a common interface. Cloud users initiate interconnections between clouds by developing clients or via broker services. Multi-cloud is a typical user-centric Intercloud scenario, where cloud users integrate their resources among multiple independent clouds or expand their cloud-based business in terms of vendor and location coverage to achieve better performance. The main difference of multi-cloud with the previous model is that a cloud provider needs to use the administrator level of permission to facilitate the multi-cloud operation. As shown in Fig.2.3, the main initiator in the multi-cloud model is a cloud user. It can be implemented by the following two different approaches:

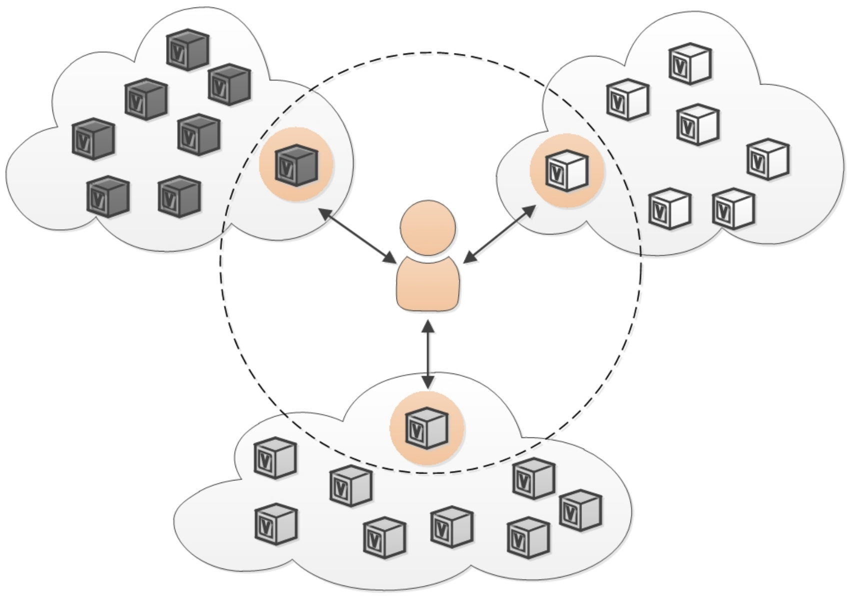
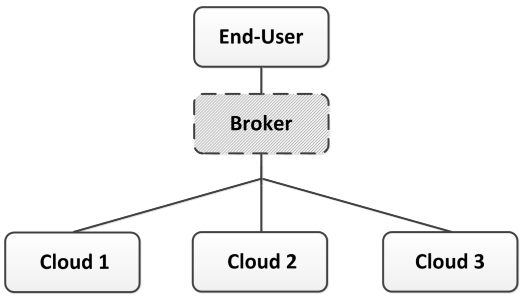


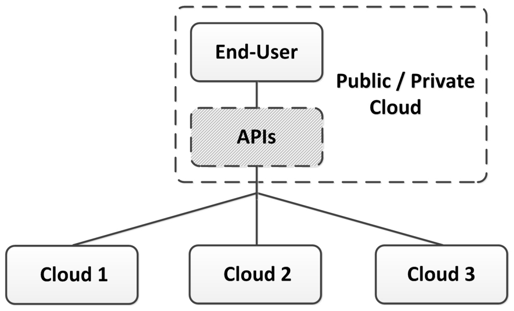
Fig. 2.3: User-centric Intercloud

- *Broker* is a third party providing the services of aggregating and managing resources on multiple clouds on behalf of the cloud users. A cloud user only describes a Service Level Agreement (SLA) to the broker. Then the broker automatically completes all the deployments and executions in the background and outputs aggregated services to its users. The advantage of the broker is that cloud user can integrate its paid resources on multiple clouds through a single entry point at the broker as illustrated in Fig.2.4a. In this case, the broker functions as a separate adapter layer between multiple clouds and end users. A cloud user needs to grant permission for the broker to use resources on all clouds in advance.

- *APIs.* Cloud users are responsible for the management of its paid services on multiple clouds. They negotiate with each cloud provider separately and integrate all the resources directly in the development of their applications. The applications either use different cloud services APIs or some abstracted unified cloud services libraries. The cloud user can host its applications on the client side independently at any clouds. And they can deploy applications on the virtual machines of private or public clouds as shown in Fig.2.4b. In this case, it allows the interoperations between clouds. However, it still belongs to the multi-cloud scenario, since the entire scheduling process only handles the users' paid resources via the user level APIs opened by cloud providers. None of the cloud providers participating in the multi-cloud process uses their administrator privileges.



(a) Broker.



(b) APIs.

Fig. 2.4: Multi-cloud architectures

To summarize, cloud heterogeneity is a major reason for the above issues. To overcome this barrier, either standard interfaces among different clouds are required or an intermediate layer undertakes the integration work. When there is a widely adopted standard interface among cloud providers, users do not need to change their cloud applications for a certain cloud service provider. At the same time, a cloud user is capable of taking advantage of strengths of all kinds of cloud services. A more flexible workload migration can also be achieved via standard interfaces among different clouds. In other words, interoperability of cloud resources and services across different cloud service providers are significant for maximizing their return on cloud investment.

2.2 Related Work on Trust Evaluation

From the provider-centric perspective, the first problem targeted in this thesis is Intercloud trust. Currently, the majority of the cloud users only employ the services on several giant public cloud providers. With the expansion and diversity of cloud-based business, a growing number of clouds look for cloud cooperation. Lacking of agreements among cloud providers leads to the infeasibility of interconnected cloud collaboration, particularly for federation among multiple private clouds. Security agreement is one of the most important prerequisites in the negotiation of unified agreements among cloud providers. The trust evaluation is the first step for establishing the security agreement. In this section, we firstly introduce main trust evaluation models for cloud computing. Then, we specifically discuss the drawbacks of proposed Intercloud trust evaluation protocols. Later, we also discuss the limitations of existing reputation-based evaluation schemes when applying to the Intercloud environment.

2.2.1 Trust Evaluation for Cloud Computing

Trust is defined as a subjective probability that one party believes that another party would perform expected actions. In cloud computing, cloud users' data are scattered and their applications are executed remotely in different datacenters of cloud providers, which is highly non-transparent. When cloud users develop commercialized products on their paid cloud resources, they have to fully trust their cloud service providers. Therefore, the trust evaluation results directly influence the decision making when choosing cloud providers. According to whether the trust assessment happens after or before using the cloud services, we classify trust management into two types. The first type is post trust evaluation carried out by cloud users after using a cloud service. The evaluation result is based on the cloud user's own experience, for instance, the fulfillment of SLA. Based on SLA compliance, Pawar et al. and Ko et al. proposed a trust model [46] and a TrustCloud framework [47], respectively. Another direction to build the trust model is to measure the performance of cloud service. In [48], Li et al. proposed a trust model based on the number of illegal connections, denial of services, the average task failure ratio and response time of the cloud service. Hwang et al. proposed a security-aware cloud system to assess the credibility of cloud service providers by the predefined policies [49] and fuzzy logic techniques. The second type is prior trust evaluation, which happens before a cloud user chooses a cloud provider. The trust result is mainly based on the previous experience of other parties. Credential-based trust evaluation is one of the approaches. Some trusted third parties assess and issue credentials for members in a federation to prove the trustworthiness of their services. Simple Public Key Infrastructure (SPKI) [50], Security Assertion Markup Language (SAML) [51] and X.509v3 [52] are standards used in the description of credential, which are still inadequate. Then, derived by the peers' knowledge, a recommendation-based trust model has been proposed.

Recommendations from friends are more credible and give the finer-grained trust evaluation results, which are usually shown in the form of feedbacks. This model has also been widely adopted in cloud computing environment. Krautheim et al. developed a recommendation-based trust model based on transitive trust relationship in [53]. The reputation-based trust model is another approach in which the evaluators take into account the feedbacks given by others. Some cloud evaluation websites are designed for consumers to identify the reliable cloud providers. For example, SoftwareInsider allows cloud users to write reviews based on their usage experience and outputs a final rating about different cloud services [54]. Habib et al. [55] developed a multi-faceted trust management system, which reassesses the uncertainty of received feedbacks from various aspects. Noor et al. [56] designed a reputation-based trust management system (CloudArmor) for cloud services. The system investigates the credibility of feedbacks which is able to handle collusion and sybil attacks. By analyzing the feedbacks from several aspects, different aggregated weights are computed to minimize the influence of dishonest feedbacks and so that the trust result can be adjusted. CloudArmor only calculates the frequency of credential value for the same consumer. Since the feedbacks are given by ordinary cloud service users, CloudArmor ignores the source of a feedback. However, for Intercloud, feedbacks can be provided by cloud providers as well. The reputation of the source of feedback is an important factor for assessing the credibility of feedbacks. Our work is classified as the second type of trust model, which is the reputation-based trust model. Unlike the previous works, we consider that the protection of feedback privacy is important for reaching an objective trust result.

2.2.2 Intercloud Trust Model

Under the Intercloud architecture proposed by Bernstein, several further works were published [57], [58], [59], [60]. Lloret and Garcia et al. described a tiered Intercloud architecture in [61]. Based on the Intercloud architecture, security issues have also been studied [62]. Powell et al. designed single sign-on authentication solution for clouds to interoperate with each other [38]. Apart from identity authentication, in the Intercloud environment, researchers also have studied the problem on access control policy conflict in [63] and [64]. The following summarizes the proposed trust models for Intercloud.

- **PKI-based trust model**

  Bernstein and Vij discussed Public Key Infrastructure (PKI) based trust model for Intercloud in [65]. Instead of classifying the clouds as trusted or non-trusted providers, this model evaluates clouds by ”Trust Index”. Basically, a certain group of cloud providers belong to the same trust domain. Each Intercloud exchange is in charge of one trust domain and exchanges the dynamic “Trust Index” with other domains. And Intercloud root is implemented as a static certification authority (CA). Nevertheless, trust is a subjective measure. Cloud providers are highly autonomous even in the Intercloud environment, personalized trust measurement criteria cannot be ignored. Assessing the trust of clouds also needs to take multiple factors into consideration and is highly related to specific contexts. PKI-based trust model, however, only releases the evaluation results of third parties.

- **Reputation-based Trust Model**

  Abawajy proposed a reputation-based trust management model [66]. Three kinds of ratings (personal experience, reputation, and honest rating) are stored and maintained by Reputation Manager. Each cloud is able to calculate the reputation by exchanging their first hand Intercloud service experience with each other through the Reputation Manager. The issue of Abawajy’s scheme is that personal experience managed by Reputation Manager is public. In the Intercloud scenario, the trust evaluation is carried out between cloud providers, which is a mutual evaluation. When all the ratings are public, cloud providers would avoid giving poor ratings. Besides they did not provide the way of choosing

Reputation Manager and correcting second-hand ratings to a reputation value.

- **Attribute-based Trust Model**

  Ngo et al. [67] discussed that the trust relationship between clouds should be built based on a particular context. By means of recommendations, direct and indirect trust chains can be dynamically set up among cloud providers. Since different clouds have their own description languages about resources, semantic transformation is applied in comparing recommendation contexts. The main drawback of this approach is that some contexts are hard to be semantic transformed and compared with each other. Limited work has been considered to verify the credibility of second-hand recommendations [68]. Clouds can launch self-promotion attacks by collusion with each other.

Most existing solutions depends on third parties to establish trust relationship and exchange trust information in plaintext. Nevertheless, it is hard to find independent trusted third parties who are accepted by all cloud providers for the trust evaluation process. Anonymous evaluation is an alternative approach to handle retaliation attack. But it is unable to judge whether the trust value is given by real service users or fabricated by clouds themselves. Furthermore, malicious clouds can also give negative recommendations without being detected.

2.2.3 Trust Management with Privacy Protection

For trust management, various reputation-based trust evaluation protocols with privacy protection have been proposed for different purposes. For instance, Clark et al. proposed a reputation system with privacy protection for mobile

Table 2.1: Comparison of trust-related protocols

Protocol	**Core architecture**	**Scenario**	**Customized result**	**Feedback privacy**	**User anonymity**
Bernstein and Vij [65]	Centralized	Intercloud	No	No	No
Abawajy [66]	Centralized	Intercloud	No	No	No
Ngo et al. [67]	Distributed	Intercloud	No	No	No
Clark et al. [69]	Distributed	Ad-hoc network	No	Yes	No
Schaub et al. [70]	Distributed	E-commerce	No	No	Yes
Tormo et al. [71]	Centralized	HbbTV	Weighted mean	No	No
Hasan et al. [72]	Distributed	Multi-agent	No	Yes	No
Our protocol	Distributed	Intercloud	Weighted mean	Yes	Yes

ad-hoc networks [69] with the focus on handling dynamic configuration (i.e., parties may join and leave dynamically). Since each party in Clark et al.'s scheme shares its feedback with all other participants, the complexity of message exchange in Clark et al.'s scheme is $O(n^2)$, n is the number of parties in the feedback exchange. Schaub et al. [70] proposed an anonymous reputation system for e-commerce websites, with the aim of guaranteeing user anonymity in the trust evaluation process. However, feedback privacy cannot be protected. Tormo et al. proposed a reputation management system for hybrid broadcast broadband TV (HbbTV) [71], which aims to compute personalized trust results based on the similarity of two users when making choices. However, the scheme needs to disclose both feedback and user identities to trusted third parties. Hasan et al. designed a privacy-preserving reputation protocol that can tackle attacks by malicious participants in a multi-agent environment [72]. However, this scheme leaks the existing trust relationships between users of the same cloud service provider. And it cannot compute customized evaluation results and ensure user anonymity. Although some of these protocols may be adapted for Intercloud with some modifications, it is still desirable to develop an Intercloud-specific trust evaluation protocol with the advent of cloud computing, as well as for effectiveness and efficiency considerations.

Table 2.1 compares different trust-related protocols in terms of different attributes. As discussed in Chapter 3, our protocol has distinctive advantages.

Apart from using a distributed architecture, our protocol can still accurately compute the trust result, even when some of the users are unavailable. This is an important requirement for a distributed system or dynamic network. As shown by the aforementioned simulation results, our protocol can handle this kind of attack effectively. Our protocol can compute customized trust evaluation results while protecting feedback privacy. However, unlike [72, 69], our protocol employs a verifiable secret sharing scheme to facilitate flexible processing and feedback updating. Our protocol also supports user anonymity during the trust evaluation process. Compared to [72] and [69], our protocol is also more efficient in terms of message transfer complexity during the enquiry phase. Note that during the enquiry/query phase, an enquirer in Hasan et al.'s scheme [72] needs to forward vn encrypted secret shares and $vn + n$ verifiable values between raters, where $v \ll n$ is the number of shares prepared by each rater. Furthermore, raters need to reply to the enquirer with n sum of shares for aggregation. Thus, in total, Hasan et al.'s scheme needs to transmit $2vn + 2n$ (i.e., $O(n)$) messages. Our protocol requires downloading m encrypted secret shares and m commitments for verification ($m < n$ is the threshold value for secret key reconstruction). Furthermore, raters need to reply to the enquirer with m sum of shares for secret key reconstruction. In total, our protocol needs to exchange $3m$ (i.e., $O(m)$) messages. Hence, our protocol is more efficient. The details of our protocol will be discussed in Chapter 3.

2.3 Secure Range Query over Encrypted Data

From the cloud user-centric perspective, the second problem studied in this thesis is about secure data query. Although sensitive data can be encrypted before they are stored in a cloud, the encrypted data can hardly be processed efficiently. Hence, a lightweight solution is required to satisfy both high security and high efficiency requirements. Searchable symmetric encryption (SSE) is a promising technique to tackle the aforementioned problem [15]. Currently, the majority of SSE schemes focus on handling keyword search over encrypted data [16, 17, 18, 19]. Some dynamic SSE schemes are designed for similarity search [20] and multi-keyword query [21] while achieving forward-privacy protection. Their main purpose is to reduce information leakage when an encrypted database is updated. However, they do not address the problem of comparison query and range query over encrypted data which are required for some applications. Compared with keyword queries, there are more technical challenges in designing an effective and secure scheme for range queries on encrypted data. We classify the existing schemes into two categories. The first type of schemes [73, 74, 75, 76, 77, 78] performs a range query $[a, b]$ by checking whether the data item is larger than the lower bound $(a \leq d_u)$ and smaller than the upper bound $(d_u \leq b)$ of a query. The comparison result directly discloses whether the unsatisfied data item is larger/smaller than the upper/lower bound of a query. The second type of schemes [79, 80] conducts a range query by treating the query range as a single keyword. This type of scheme only outputs whether the data values are within the query range or not (i.e., $d_u \in [a, b]$). The server is unable to learn the ordering of unsatisfied records. Therefore, it is more secure than the first type. However, it introduces plenty of unmatched query results, and requires a large amount of index storage space. To maintain similar search time complexity (i.e., as $O(\log N)$ or even faster), these schemes usually organize indexes of the entire dataset using special data structures.

2.3.1 Order Preserving Encryption

Order preserving encryption (OPE), one of the methods for supporting numerical comparison between ciphertexts has been studied in [73, 74, 75, 76]. OPE ciphertexts are deterministic and preserve the numerical order between their plaintexts, that is $a \leq b$, if and only if $OPE(a) \leq OPE(b)$. Since the server can directly obtain the ordering of data items from their OPE ciphertexts, any comparison-based index structures (e.g., B+ Tree) used for indexing plaintext data can be directly applied to the OPE ciphertexts. Although OPE can achieve the same search efficiency as in the plaintext cases, it is unable to prevent "inference attacks" [81]. This attack mainly exploits the ordering information and the frequency of data items disclosed from OPE ciphertexts.

2.3.2 Order Revealing Encryption

To prevent "inference attacks", Lewi and Wu designed the latest Order revealing encryption (ORE) scheme [78]. Their scheme splits the data ciphertext into two parts. The right ciphertext is assigned as the index for records, and the left ciphertext is the trapdoor of range query, upper or lower bound. The comparison is performed between the right ciphertext of data $Enc_R(d_u)$ and left ciphertext of query boundary $Enc_L(a)/Enc_L(b)$. The scheme begins with proposing a small-domain ORE scheme with best-possible semantic security, for the right ciphertexts. In other words, the right ciphertexts generated from the small-domain ORE can hide both the frequency and the ordering of records which are robust against inference attacks. However, the length of each right ciphertext grows linearly in the size of the entire plaintext space. To improve comparison efficiency and also reduce the index size of small-domain ORE, the scheme is designed to build indexes by grouping the message space of data into blocks. However, this approach leaks the first block that is different between two ciphertexts during the comparison. This leakage also indicates the different relative distances between different data and the same query, and in turn, these distances implies the ordering between data items. When the block size is relatively small, the server can precisely estimate the relative ordering information. When the block size is large, however, the ciphertext size of ORE grows exponentially. To achieve sublinear search time $O(\log N)$ (N is the dataset size), the scheme saves the indexes in sorted order before searching. This step indicates that the server learns the ordering of indexes during the query; even the ordering of indexes at rest is indistinguishable one from the other. The left ciphertext is deterministically created in the ORE scheme for each query condition. The search pattern of queries in the ORE scheme is leaked from trapdoors to the server. The ORE scheme also inherits the weakness of the comparison query. Through the searching result, the entire encrypted dataset can be divided into three ordered parts ($\mathcal{R}_1 < \mathcal{R}_2 \in [a, b] < \mathcal{R}_3$). ORE scheme has been used in other scenarios. Yuan et al. leveraged the ORE to generate indexes for range queries in distributed key-value stores [82]. Their scheme compares two ORE ciphertexts to get a boolean result which prevents the disclosure of order relationships from the comparison results. Shen et al. adopted ORE to support the approximate constrained shortest distance queries over encrypted graphs [83]. To filter out paths in an encrypted graph with the constrained total cost, the scheme encrypts the cost of each edge using ORE encryption, and employs an efficient tree-based ORE ciphertext comparison protocol. However, this scheme also inherits the weaknesses of the ORE scheme, such as disclosing the search patterns and order relationship among the costs.

2.3.3 Comparable Encryption

Comparable encryption (CE) was proposed by Furukawa [84, 85]. The basic idea of comparable encryption is to separately encrypt the record value and query boundaries with different approaches [85]. For example, $Enc(d_u)$, $Enc(d_u^{'})$ are the ciphertexts of two record values, and $Token(a)$ is the trapdoor of the lower bound of a range query $[a, b]$. Based on ciphertexts of $Enc(d_u)$ and $Enc(d_u^{'})$, the server cannot recognize the numerical order between d_u and $d_u^{'}$. Whereas based on $Enc(d_u)$ and $Token(a)$, the server can learn the numerical order between d_u and a. In other words, comparable encryption ciphertexts are semantically secure against inference attacks, and they are unable to be directly compared with each other. To support a comparison, comparable encryption generates tokens for the query boundaries. This feature ensures that any two data items can be compared only when either of their tokens is obtained. In the token generation, comparable encryption adopts prefix-preserving encryption (PPE). When any two data items have the same first n digits of prefix strings, their PPE tokens must also have the same sequence of n high elements as their prefixes. Hence, the comparison seeks to check the equivalence between ciphertext and token starting from the high elements until finding the first one that is different. Comparable encryption has a similar weakness as ORE, which leaks the length of the longest common prefix string of record value and query boundary during the comparison [86]. Moreover, it tells the attacker about the comparison operator, since it separately compares data items with the upper and lower bounds of a range query. This leakage indicates the relative numerical difference between data items of the same query. Since tokens of queries are deterministic and prefix preserving, the repetition and numerical order between upper/lower bounds of issued queries are also leaked in comparable encryption.

2.3.4 Bloom Filter-based Range Query Scheme

Li et al. proposed a privacy-preserving range query scheme that can achieve index indistinguishability [79]. The main idea of the scheme is to test whether in a range query $d_u \in [a, b]$, the prefix set of d_u and prefix union set of $[a, b]$ have the same elements [87]. This scheme can avoid a full dataset scan, and order leakage between left and right nodes during the binary search as well. It organizes the prefix sets of data items in random order using a special complete binary tree called Privacy Bloom Filter Tree (PBtree). Specifically, the root node stores the prefix union of all of the data items. It then recursively splits the data items of each node into its left and right child nodes. This split only ensures that each child node has an equal number of data items.

Since data items are randomly split, the number of left child data items is not necessarily smaller than that of the right child data items. To achieve PBtree structure, this scheme sacrifices the storage cost of $O(N\ \log N\ \log M)$, where M is the domain size of data items. To improve the speed of checking the overlap between two prefix sets, the scheme employs the data structure of Bloom filter [88], but this creates false positives in the query results. Although the PBtree structure seeks to achieve a sublinear search time, the actual search time is $\Omega(\log N \log Q + R)$, where Q and R are the sizes of query range and result, respectively. This search time has no upper bound because of the random placement of the data items in the PBtree and possible false positive results [80]. This scheme is only proved to be secure under certain conditions (i.e., non-adaptive adversaries following Goh's definition [89]). Goh's security definition does not guarantee trapdoor privacy. The trapdoor generation is always deterministic, and thus the search pattern is disclosed [87].

2.3.5 Range Searchable Symmetric Encryption

Searchable Symmetric Encryption is widely used for keyword search over encrypted data. Inspired by this idea, Demertzis et al. proposed the concept of Range Searchable Symmetric Encryption (RSSE) [80]. This concept turns the problem of range query into a multi-keyword search problem, such that any secure SSE schemes can be employed to realize the RSSE concept. IND-CKA2 (indistinguishability against adaptive chosen keyword attacks) is the strongest security model available for SSE schemes, introduced by Curtmola et al. [90]. The scheme proposed by Demertzis et al. also satisfies the IND-CKA2, but has additional leakages. This means that an attacker can learn nothing more than the formulated leakages. In this scheme, a range query condition is replaced by several sub-ranges, with each sub-range represented by a keyword. Data items belonging to the same sub-range are considered as documents containing the same keyword. To prevent multiple sub-ranges from being

Table 2.2: Comparison of privacy-preserving range query schemes

Scheme	Index Order	$\geq / \leq$ $> / <$	Search Pattern	Space Usage	Search Time	False Positive
OPE [74]	leak	leak	leak	$O(N)$	$O(\log N)$	no
ORE [78]	no/ leak	leak	leak	$O(N\ 2^b\ (\log M)/b)$	$O(N)/$ $O(\log N)$	no
CE [84]	leak	leak	leak	$O(N\ \log M)$	$O(N)$	no
PBtree [79]	no	no	leak	$O(N\ \log N\ \log M)$	$\Omega(\log N\ \log Q + R)$	$O(R)$
RSSE [80]	no	no	partial leak	$O(N\ \log M)\times$ Size(record)	$O(N)$	$O(N)$
Our Scheme	no/ leak	no	no	$O(N\ \log M)$	$O(N)/$ $O(\log N)$	no

associated with the same data item, the scheme needs to duplicate records into all sub-ranges having its value. However, this duplication requirement generates a much larger dataset (i.e., compared to the original dataset). With the aim of reducing storage cost and the number of sub-ranges, the scheme designs a new structure called TDAG (i.e., Tree-like Directed Acyclic Graph). TDAG tree is constructed by inserting a middle node between any two peer nodes on the binary tree of sub-ranges. During the search phase, each query is mapped to a single TDAG node based on the lowest common ancestor node that covers the query range. However, the TDAG structure creates an unacceptable number of false positive results when the data skew rate is high. These false positives reduce the search time to $O(N)$. Queries with similar range values are targeted to the same node on the TDAG structure, such that the search pattern is partially leaked.

Table 2.2 shows the comparison of different privacy-preserving range query schemes, including our scheme which will be discussed in Chapter 4. Parameter N is the dataset size, R is the result size, Q is the query range size, M is the domain size of data items, and b is the block size in bits of the ORE scheme. To summarize, the first three range query over encrypted data schemes [78, 74] provide higher search efficiency, but a lower security guarantee than those of the last two range query over encrypted data schemes [79, 80]. Compared to existing schemes, our scheme supports both types of range query (i.e., $a \leq d_u$

and $d_u \in [a, b]$) and can hide the comparison operator during the query, even using the first type of comparison approach. The search pattern of queries (i.e., non-deterministic trapdoor) is also concealed. Compared with the listed schemes, our scheme does not have a high space cost, and produces no false positives in the query results. Our scheme can support binary search (i.e., search time is $O(\log N)$) when ciphertexts are sorted before searching, which inevitably leaks the ordering of the index during the query.

2.4 Access Pattern Leakage in Searchable Symmetric Encryption

As mentioned in the last section, searchable symmetric encryption (SSE) allows database searching to be conducted over encrypted data. Currently, the majority of SSE schemes are secure index-based. This means that each encrypted record is associated with some secure indexes. Each index is created based on a record to be queried using one of its attribute values. To allow a cloud server to search for the encrypted records, a trapdoor is created for each query. During the search phase, the server finds the matched records by comparing trapdoors of queries with indexes of records. The main reason behind access pattern leakage attacks is that most searchable symmetric encryption schemes can only safeguard the confidentiality of documents and query values at rest. The strongest security model available for SSE schemes is formulated by Curtmola et al. in [91]. The schemes satisfying this model ensure that the indexes and trapdoors themselves do not leak the documents and query values even when the attackers adaptively issue the queries. However, the model allows the attackers to learn the statistical information by observing the searching results (i.e., access pattern of queries). Moreover, most of these schemes are designed to perform on a single server only. Hence, the server is able to gain all the trapdoors and their query results. When the dataset to be searched is large enough, the server can statistically analyze the actual value of queries or documents. In this section, we give a literature review of the related work on both access pattern leakage attacks and countermeasures.

2.4.1 Access Pattern Leakage Attacks

- **IKK Attack**

  Islam, Kuzu, and Kantarcioglu (IKK) presented the first access pattern attack on SE schemes that can recover the trapdoor keywords [92]. IKK attack mainly uses the unique co-occurrence frequency of any two keywords. An IKK attacker has two square matrices M_p and M_c with both the size of the number of keywords. M_p is obtained from the plaintext of the indexed dataset before searching. M_c is built by observing a sequence of trapdoors and their query results. Each element in the matrix M_p and M_c represents a co-occurrence frequency of any two keywords or any two trapdoors appearing in the same document. For instance, (i, j)th element in M_c is $M_c(i, j) = \frac{|R_i \cap R_j|}{N}$, where R_i and R_j are the result sets of the ith and jth trapdoors, and N is the total number of documents. The process of recovering trapdoor keywords is to match the elements between M_p and M_c (M_c is a permutated submatrix of M_p). However, the successful rate of IKK attack declines with the increasing number of keywords under attack. This is because larger matrix M_p leads to less pairs of keywords with unique co-occurrence frequency [93].

- **Count Attack**

  Cash et al. [93] presented another access pattern attack called count attack, which targets on a large number of keywords. In addition to the co-occurrence frequency of keywords, the server can also learn the result count of each keyword (i.e., number of documents that match with each trapdoor). Count attack is based on the Zipfian distribution of keywords in the natural language text. That is the most frequent keywords often have unique and higher frequencies than the other keywords. Hence, trapdoor keywords can be identified by the unique result count. Basically, an attacker is assumed to know the frequency of each plaintext keyword w (i.e., $count(w)$) before searching. Then during SE searching, the server counts the number of documents in each trapdoor query result (i.e., $count(R_q)$). When a keyword in the plaintext dataset has the same unique frequency as a trapdoor in the searchable encryption scheme, that is $count(w) = count(R_q)$, the attacker can immediately reveal the trapdoor T_q with the keyword w. For the queries that do not have unique result counts, the attacker resorts back to compare their co-occurrence with queries which have unique result counts. Thus, the count attack can be reduced, when there are less number of keywords that have unique result counts. In the IKK attack, matching of the ciphertext and plaintext keywords is initially by random mapping. In the Cash et al.'s attack, it only compares the co-occurrence counts of trapdoors with the same result count. This narrows the candidate keywords before the co-occurrence comparison. Hence, the count attack has a higher efficiency and successful recovery rate than the IKK attack, especially when the number of keywords under attack is large.

2.4.2 Countermeasures Against Attacks

- **Query Result Padding**

To prevent the IKK attack, Islam et al. proposed a padding countermeasure in the same paper [92]. The basis idea is to make the query responses of different keywords as similar as possible. As a result, there are less pairs of queries having the unique co-occurrence frequency. To achieve this goal, they generated files with keywords that they do not contain originally. Thus, the padding includes false positives in the query results. Specifically, the appearance of each keyword will be padded up to at least α similar keywords. After padding, an IKK attacker can only correctly guess the trapdoor keyword with the probability of at most $1/\alpha$.

However, a small amount of padding in the dataset is not sufficient to resist attacks. Cash et al. have shown that the padding method is hard to reduce the count attack [93]. This is because the huge difference between the appearance of each keyword. It is infeasible to fill the query result of all keywords to be the same, which leaves the padded dataset similar to the original one. By adjusting co-occurrence comparison from exact to approximate, the recovery rate of count attack is still the same, even when the Enron dataset [94] is increased by 15% after padding.

- **Oblivious RAM**

Data encryption can only protect content confidentiality. Oblivious Random Access Memory (ORAM) was designed to prevent the physical memory access pattern leakage [95]. It is a middle layer in between the running programs and physical memory. Once the memory is being visited, ORAM keeps on reshuffling the data allocation in the memory [96]. Hence, ORAM can ensure independence between physical and logical access pattern. That is the attacker cannot know whether the program is reading/writing and if the program has accessed the same data before [97]. ORAM can conceal all the data access patterns. However, the computational cost of ORAM is very high, due to the multiple rounds of interactions between client and server [22]. The shuffle operation of ORAM algorithm is another performance bottleneck, which leads to a long response time and high computational cost [98, 99]. In addition, ORAM is hard to support complex query types (e.g., equality, prefix and range SQL query in the relational database) [23]. Therefore, existing ORAM-based schemes are not practical for large datasets.

- **Vertical Fragmentation**

  Vertical fragmentation of database [100, 101, 24, 102] is designed to hide the sensitive association among the database attributes. For instance, separating the database table columns of employees' name and salary in two servers can avoid the leakage of private information about specific persons. Some of the vertical fragmentation schemes do not encrypt sensitive records, while assuming that servers will not collude with each other. This type of schemes cannot protect the privacy of single attribute [103]. Vertical fragmentation can be used to counter the IKK attack. For any two attributes having the unique co-occurrence appearance among records, vertical fragmentation distributes them to different servers. Then, the co-occurrence count about queries observed on a single server can be reduced to zero. However, the frequency of a single attribute is fully revealed, count attack still occurs on the single server, even when the records are encrypted. Hence, vertical fragmentation alone cannot resist the access pattern leakage attack [104].

2.4.3 Security Strategies

The success of IKK attack and count attack both rely on the accurate matches between ciphertext observation and plaintext dataset knowledge. If either the ciphertext observation or the plaintext dataset knowledge is inaccurate, the trapdoor recovery rate will be quite low. As shown in [93], when the attacker can only get 90% of the plaintext dataset, the recovery rate of IKK and count attack declines to less than 0.6 and 0.2, respectively. On the other hand, when the attacker can only observe partial query results about the encrypted dataset, the recovery rate will also be reduced, even if the attacker has the complete knowledge about the same plaintext dataset. As discussed in Chapter 5, our scheme is designed based on this idea, which is secure against the attackers even having the complete plaintext knowledge about the same dataset.

Apart from reducing the accuracy of matching in the IKK and count attack, hiding the uniqueness of observed access pattern is another solution. In the keyword padding approach [92], the unique co-occurrence is reduced by padding nonexistent query results. However, the frequency between different keywords is extremely large (e.g., the most frequent keyword occurs nearly 70 times as often as the 100 most frequent keywords). It is unable to fill all keywords to have the same query result, otherwise it will bring a significant number of false positives. As discussed in Chapter 5, we consider using multi-clouds to tackle the security problems. Our scheme removes the records in the trapdoor query results to other cloud servers, in order to make the co-occurrence probability and keyword frequency observed by a single server incorrect. At the same time, our scheme distributes queries with similar results on the same server, in order to make the query result of trapdoors on the same cloud server as similar as possible.

3

A Distributed Trust Evaluation Protocol with Privacy Protection for Intercloud

3.1 Introduction

Trust in a service is generally concerned with a belief in whether the service can be delivered satisfactorily, in accordance with certain trust attributes. In the Intercloud context, a cloud service provider (or user) typically trusts another cloud service provider based on certain trust attributes, such as service reliability, quality of service and service efficiency. Before choosing/using a service, trust evaluation is often conducted based on the feedback of existing users (i.e., reputation-based trust evaluation). Indeed, feedback provided by past cloud users is a good reference for trust evaluation [68, 56]. Based on this feedback or rating, a cloud user can evaluate how likely (e.g., a probability) that a cloud service will be performed as expected. However, the credibility of feedback is often difficult to guarantee [56] as cloud users often avoid leaving honest comments, especially negative ones [72]. The main reason for this behavior is the unequal status between cloud service providers and cloud users (e.g., a cloud service provider can easily remove negative comments about its services). This problem becomes more serious in the Intercloud environment. As there is more and more mutual co-operation, a cloud user or his/her business could be another type of cloud service provider in future business transactions. This possible mutual relationship makes the privacy requirement even more important in the Intercloud scenario. If feedback information cannot be made private, cloud users may only give positive feedback, as they want to maintain a good relationship or are fearful of retaliation [13, 14]. Hence, it is important to develop an effective and flexible trust evaluation protocol with privacy protection for Intercloud.

Inspired by related work on trust and reputation evaluation, this chapter presents a distributed trust evaluation protocol with privacy protection to address the following important requirements. Our contributions are summarized as follows.

- **Cloud user protection.** To encourage honest feedback/ratings and to prevent possible retaliatory attacks, both user identity and user feedback privacy should be protected. Ideally, feedback should not be linked with the user and business privacy of the user (i.e., which user has performed business with which cloud service provider should not be disclosed). Our protocol uses an innovative mechanism to store feedback, and employs homomorphic encryptions [25] and [26] with verifiable secret sharing [27] to protect feedback privacy. Finally, neither the cloud service provider nor the enquirer can obtain individual feedback.

- **Cloud service provider protection.** Malicious users can generate a large volume of misleading feedback or faked ratings to damage the reputation of a cloud service provider. To address this important issue, our proposed protocol allows a cloud service provider to certify a rater's eligibility. Furthermore, as explained later, our protocol allows the filtering of extreme ratings without leaking privacy information.

- **Trust result availability.** Existing distributed protocols typically require all concerned parties to remain online to facilitate feedback collection. This requirement is not practical in the Intercloud environment.

The proposed protocol can still function well, even if concerned parties are not available to contribute to trust evaluation.

- **Flexible processing of protected feedback.** To facilitate customized trust evaluation and reduce the influence of misleading ratings, it is desirable to provide a flexible way to subjectively process protected feedback results. For example, suppose there are two sets of ratings: 1, 5, 5, 5 and 4, 4, 4, 4. Although they both give an average rating of 4, one or the other set may be preferred by different enquirers. Our protocol provides an innovative mechanism to store and process ratings in a flexible manner (e.g., assigning a lower weight to de-emphasize or filter extreme ratings) while protecting feedback privacy.

The remaining sections of this chapter are organized as follows. Section 3.2 presents the models and overview. Section 3.3 explains the feedback computation. Section 3.4 discusses the trust evaluation protocol. Section 3.5 presents the security proof and discusses the security analysis. Section 3.6 presents the simulation settings and Section 3.7 discusses the simulation results. Section 6 concludes this chapter.

3.2 Models and Overview

In this section, we first discuss the system model, main protocol phases and adversary model with assumptions.

3.2.1 System Model

Fig. 3.1 shows the system model or architecture with five main components: cloud service provider (CSP); user/rater; enquirer; distributed feedback storage

(FBS) and secret sharing network. Under the Intercloud system model, the **CSP**s provide cloud services collaboratively to users, and serve each other as well. In general, there are two types of cloud service **users**: consumer users and business users. Consumer users use a cloud service. They have only a one-way trust/service relationship with a CSP (i.e., the service is one-way CSPs serving consumer users). Business users may provide services to each other (i.e., two-way trust/service relationship). After using a cloud service or under certain arrangements, the users can rate the service or a trust attribute (e.g., availability, response time, price, technical support). That means the users are also **raters** during the feedback/rating process. Note that to facilitate the explanation, we focus on evaluating one service or trust attribute. It can easily be extended to evaluate multiple services or trust attributes. For a business user, a human representative of the respective organization can provide the rating/feedback. Furthermore, with advances in intelligent computing technologies, a software agent or software robot can perform the feedback/rating task as well. These software agents/robots can communicate through Intercloud gateways (e.g., using an Intercloud communication protocol with predefined XML-based messages). In this case, the Intercloud system becomes a highly autonomous system. The **enquirers** can be any parties who want to use the trust evaluation protocol to evaluate the trustworthiness of a CSP based on the feedback/ratings.

In the traditional web-based trust evaluation system, it usually depends on a single third party to maintain the feedback information and conduct the trust evaluation process. However, the this kind of system has some weaknesses due to its centralized nature. Firstly, a cloud service provider can easily compromise the system by only publishing positive comments on its service. Therefore, the fairness of trust evaluation is hard to guarantee. Secondly, relying on a single third party is vulnerable to single point of failure, which affects the availability of the trust evaluation service. In the Intercloud environment, the trust evaluation relationship includes the evaluation between different cloud service providers. It becomes even harder or impossible to find a single third party to handle the trust evaluation workload. Therefore, in our protocol, we adopt the distributed architecture in which the distributed feedback storage can be implemented either by the cloud service providers themselves or by several trusted third parties. We assume there is a distributed **FBS** for storing feedbacks. For the Intercloud system, the FBS can be implemented by the Intercloud exchanges, or by the cloud service providers themselves using a blockchain-based system. Basically, a feedback is submitted as a record in the blockchain. With the blockchain consensus mechanisms, feedback integrity can be preserved. Alternatively, other distributed storage systems, with integrity guarantees, can also be used to store feedbacks. We also assume there is a **secret sharing network** for protecting encrypted feedback. Detailed operations will be explained later. The secret sharing network can be formed by the users/raters to protect their feedback privacy. Alternatively, it can be provided by a third party as a service. In the aforementioned autonomous Intercloud system, secrets can be shared by software agents through the Intercloud gateways. Again, communications can be facilitated by the Intercloud communication protocol. Note that the secret sharing mechanism is conducted in a distributed manner, without direct interaction between raters. Through a possibly anonymous secret sharing address (SSA) (e.g., a server or Intercloud gateway), each rater only needs to reply to the enquirer by decrypting a certain secret published/provided by the FBS.

3.2.2 Main Protocol Phases

In this subsection, we give an overview of the three main phases of the protocol. The detailed protocol will be presented in Section 3.4. As shown in Fig. 3.1, the first phase is **rater registration**. After using a cloud service provided by a CSP, a (business/consumer/agent) user registers with the FBS as a rater so that feedback/ratings on the service can be submitted (i.e., for a trust attribute).

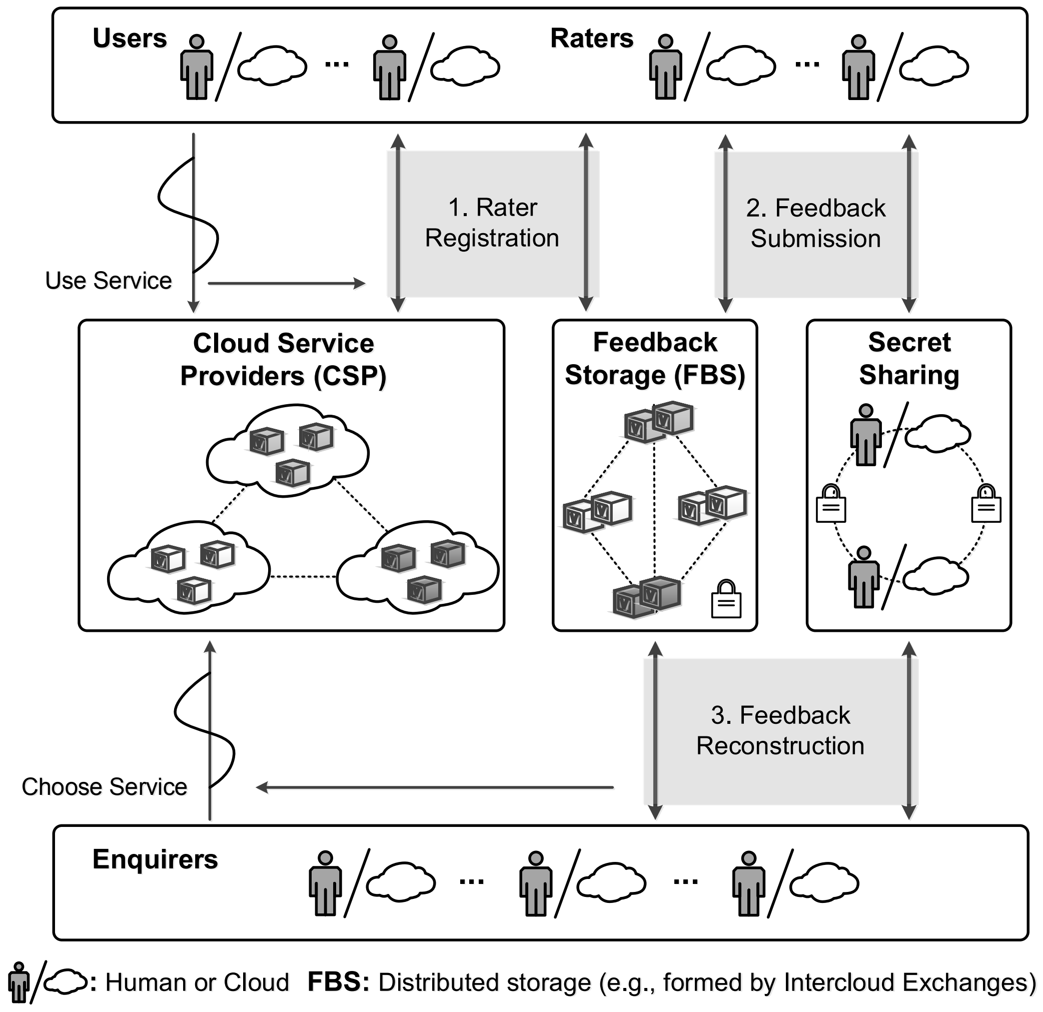


Fig. 3.1: System architecture and protocol overview.

For the registration, it is important to ensure that a rater is a real user. To fulfill this requirement, a user/rater (e.g., the corresponding public key) is certified by the CSP by means of a blind signature [105]. For extra verification, an additional blind signature certification can be provided by a bank or a payment system (i.e., to verify that the user has a payment transaction with the CSP). Note that there can be no linkage between the user identity and rater identity. During registration, the user/rater also needs to provide a secret sharing address (SSA). After rater registration, a secure channel can be set up between the user/rater and the FBS for submitting feedback. Anonymous evaluation is an alternative solution to prevent the retaliation attack. However, it has the weakness of untraceability. In other words, the enquirer cannot determine whether the feedback or comment provided by an anonymized user is based on a real cloud service experience. Furthermore, using fully anonymous evaluation also facilitates the generation of fake feedbacks. Due to the untraceability of fully anonymous evaluation, slander or self-promotion attack cannot be easily detected. In our protocol, the cloud service provider adopts the blind signature method to prove that a user has the right to submit a feedback. As a result, neither the cloud provider nor the user/enquirer can link the unblinded signature (i.e., rater identity) with a blinded signature (i.e., user identity). At the same time, the feedback storage can still use the public key of the cloud provider to verify the rater identity without affecting user privacy.

The second phase is **feedback submission**. In this phase, raters choose different secret keys to encrypt their feedback, using a symmetric homomorphic encryption scheme. Due to the additively homomorphic property of the encryption scheme, different raters' encrypted feedback can be added together directly to produce overall feedback, which can only be decrypted with the sum of the raters' secret keys. To support the decryption of the overall feedback while preventing the disclosure of individual feedback, the raters form a secret sharing network based on a verifiable secret sharing scheme. The encrypted feedback and aforementioned SSAs are maintained by the FBS (i.e., a secure distributed storage to protect data integrity). Detailed approaches will be presented in the Section 3.4.2. The last phase is the **feedback reconstruction** for trust evaluation. To conduct a trust evaluation based on encrypted feedback, an enquirer progressively sends request to each rater based on SSAs provided/published by the FBS. Note that there can be no linkage between a SSA and the corresponding rater. A rater replies to the request based on the information maintained on the FBS. When there are sufficient replies (i.e., secret shares) from the raters, the enquirer can regenerate the required secret key for decrypting the feedback or rating result.

We rely on the distributed feedback storage FBS (i.e., a group of cloud providers or trusted third parties) to verify rater identities (i.e., to check whether a rater has really used the cloud service before). Therefore, when a malicious cloud service provider colludes with some fake users to submit negative feedbacks to other clouds. These fake users can be detected during the rater verification. This is because a fake user cannot generated the required digital signature. Another possible attack is that a malicious cloud service provider colludes with a number of eligible users to submit misleading feedbacks. While this cannot be detected by FBS during the rater registration phase, our protocol allows an enquirer to flexibly process the ratings by using different evaluation weights. Hence, the enquirer can filter extreme ratings which can reduce the influence of misleading feedbacks. Note that the evaluation can be performed while protecting feedback privacy. The detailed protocol will be presented in Section 3.4.3.

3.2.3 Adversary Model and Assumptions

In the protocol, we assume that the FBS is reliable for verifying rater identities and ensuring feedback integrity (e.g., by using a blockchain-based system formed by the Intercloud exchanges). That means, once a feedback is submitted, it cannot be altered or removed from the FBS. However, the FBS itself may not guarantee privacy protection (i.e., its main purpose is to protect data integrity). We also assume that all communication channels are secure (i.e., communication security is outside our scope). Our security goal is to prevent misbehaving parties from jeopardizing the system operation. As previously mentioned, with the aim of protecting cloud users and CSPs and safeguarding system availability, the scope of this chapter is to tackle the following major security attacks using a distributed system.

- ***Attack 1:*** *Malicious users.* These malicious users try to affect trust evaluations by submitting misleading feedbacks to the FBS. They may have never used the cloud services before, but pretend to be real users.

- ***Attack 2:*** *Selfish raters.* Attacks from selfish raters can be further divided into two types.

  a) Selfish raters prepare incorrect secret sharing information.

  b) Selfish raters refuse an enquirer's request for secret key reconstruction, such that the enquirer cannot conduct the trust evaluation.

- ***Attack 3:*** *Malicious FBS.* Attacks from a malicious FBS can be further divided into two types.

  a) It discloses the privacy information of a rater so that the user identity can be found.

  b) It leaks rater information to a CSP so that negative raters can be identified.

- ***Attack 4:*** *Malicious CSPs.* Attacks from malicious CSPs can be further divided into two types.

  a) It may impersonate a valid enquirer to ask for honest raters to obtain individual feedbacks.

  b) It may manipulate the FBS and collude with a number of malicious raters to reconstruct the secret keys of honest raters so as to decrypt individual feedback.

3.3 Computation of Feedback Results

Before presenting the trust evaluation protocol in detail, this section first introduces how to compute feedback results for trust evaluation, while protecting feedback privacy.

3.3.1 Processing of Feedbacks in Plaintext

In this subsection, we first discuss the processing of feedback or ratings in plaintext. Let $\mathbb{R} = \{r_1, ..., r_n\}$ be the set of users/raters that have used a cloud service. These raters can provide ratings for trust evaluation. As mentioned above, to facilitate the explanation, we focus on one service or trust attribute. There is a set of rating choices $\mathbb{B} = \{b_1, ..., b_z\}$ for the raters to choose from. Each rating choice represents a different level of satisfaction with the cloud service or a trust attribute (e.g., in a five-star rating system, one-star ($\star$) indicates not satisfactory and five-stars ($\star\star\star\star\star$) indicates very satisfactory. We define f_l as the feedback of a rater r_l $(1 \leq l \leq n)$ for a service or trust attribute. Each rater chooses one of the rating levels in the feedback, that is $f_l \in \{b_1, ..., b_z\}$. To facilitate the later computation in our scheme, we present each rating choice as a big integer with $z(\lfloor \log n \rfloor + 1)$ binary bits. Each set of $(\lfloor \log n \rfloor + 1)$ binary bits allows the counting on each rating choice. For example, consider that there are four raters $(n = 4)$ giving feedback to a five-star rating system. For each rating choice, three binary bits are used for counting purposes (i.e., how many raters chose the choice). That means that each rater's feedback f_l is one of the following ratings:

b_5	⋆⋆⋆⋆⋆	0 0 **1**, 0 0 0, 0 0 0, 0 0 0, 0 0 0
b_4	⋆⋆⋆⋆	0 0 0, 0 0 **1**, 0 0 0, 0 0 0, 0 0 0
b_3	⋆⋆⋆	0 0 0, 0 0 0, 0 0 **1**, 0 0 0, 0 0 0
b_2	⋆⋆	0 0 0, 0 0 0, 0 0 0, 0 0 **1**, 0 0 0
b_1	⋆	0 0 0, 0 0 0, 0 0 0, 0 0 0, 0 0 **1**

By storing the ratings using the above mechanism, the number of raters for each choice can be found by adding the feedback. For example, raters r_1, r_3, r_4 rate for ⋆⋆ and r_2 rates for ⋆⋆⋆⋆⋆. By adding their feedback together, we can obtain the following result in binary.

$u_1, f_1 = b_2$	0 0 0, 0 0 0, 0 0 0, 0 0 **1**, 0 0 0
$u_2, f_2 = b_5$	0 0 **1**, 0 0 0, 0 0 0, 0 0 0, 0 0 0
$u_3, f_3 = b_2$	0 0 0, 0 0 0, 0 0 0, 0 0 **1**, 0 0 0
$u_4, f_4 = b_2$	0 0 0, 0 0 0, 0 0 0, 0 0 **1**, 0 0 0
$R_{sum} = \sum_{l=1}^{l=4} f_l$	$\underline{\mathbf{0\ 0\ 1}}$, 0 0 0, 0 0 0, $\underline{\mathbf{0\ 1\ 1}}$, 0 0 0
$\underline{\mathbf{0\ 0\ 1}}_2 = 1_{10}$, $0\ 0\ 0_2 = 0_{10}$ $\underline{\mathbf{0\ 1\ 1}}_2 = 3_{10}$	

By converting the binary sum to the corresponding decimal number, we can obtain the number of raters for each rating choice, that is $|b_j|$. In the above example, the binary bits $\underline{001}$ indicate that one rater rated ⋆⋆⋆⋆⋆ (i.e., $|b_5| = 1$). Also, the binary bits $\underline{011}$ indicate that three raters rated ⋆⋆ (i.e., $|b_2| = 3$). An enquirer can determine the number of raters for each choice by computing the following sum:

$$\mathcal{R}_{sum} = \sum_{l=1}^{l=n} f_l. \tag{3.1}$$
$$|b_1|, ..., |b_z| = \text{Convert}(\mathcal{R}_{sum}).$$

$\mathcal{R}_{sum}$ is the feedback sum of raters in the set $\mathbb{R}$. After converting the binary bits in $\mathcal{R}_{sum}$ to the corresponding decimal number, the number of raters for each rating choice can be found, that is $|b_1|, ..., |b_z|$. In subsequent sections, we describe how to obtain $\mathcal{R}_{sum}$ without disclosing individual rating of raters. The trust evaluation result can then be computed as follows:

$$\mathcal{T}r = \frac{\sum_{j=1}^{j=z} w_j \times |b_j| \times j}{n}. \tag{3.2}$$

$\mathcal{T}r$ represents the trust evaluation result for an enquirer based on the feedback ratings. It is a weighted mean of different rating choices. w_j is the evaluation weight assigned by each enquirer to the jth rating choice. $|b_j|$ is the number of raters choosing the j-th rating choice. Note that besides Equation (3.2), other similar formulas can also be used. Depending on the enquirer's preference, different evaluation weights can be used for evaluation purposes. Note that if the evaluation weight for all ratings is equal to 1, the result will give the average rating. By using the aforementioned approach, an enquirer can process the ratings more flexibly. For example, suppose that an enquirer wants to choose either cloud 1 or cloud 2. The ratings for cloud 1 and cloud 2 are $\{3, 3, 3, 3\}$ and $\{1, 1, 5, 5\}$, respectively. That means, the average ratings are both 3. However, if the enquirer prefers to choose a cloud service with more five-star ratings, a higher weight can be assigned (e.g., 1 for five-star ratings, 0.8 for other ratings). Thus, the trust evaluation result for cloud 1 is $(0.8 \times 4 \times 3)/4 = 2.4$. And the trust evaluation result for cloud 2 will become $(0.8 \times 2 \times 1 + 1 \times 2 \times 5)/4 = 2.9$. So cloud 2 will be chosen. On the contrary, if an enquirer wants to assign a lower weight to filter or de-emphasize extreme ratings (e.g., 0.8 for one-star and five-star ratings, 1 for other ratings). Then the trust evaluation result of cloud 1 is $(1 \times 4 \times 3)/4 = 3$. And the trust evaluation result of cloud 2 will become $(0.8 \times 2 \times 1 + 0.8 \times 2 \times 5)/4 = 2.4$. Hence, cloud 1 will be chosen. Note that as explained later, this flexible processing of ratings can be performed while protecting feedback privacy.

3.3.2 Homomophic Encryption of Feedback

In the last subsection, we have introduced how to process feedback in plaintext. To encourage frank feedback and prevent a retaliatory attack, it is important to protect feedback privacy. In this subsection, we present how to process encrypted feedback for trust evaluation. The majority of the homomorphic encryption schemes are based on asymmetric key encryption. That is, plain text/values are encrypted using the same public key, such that they can be directly added or multiplied together. However, in our scenario, it is difficult for raters to share the same pair of public and private keys. On the contrary, we should allow each rater to use a different secret key to protect feedback privacy. In our protocol, we adopt the symmetric homomorphic encryption scheme proposed in [25]. The encryption of a private feedback f is defined as:

$$c = E_k(f) = [(f + k) \times MK] \;\; mod \; \alpha \tag{3.3}$$

where parameter α is a prime, MK is a master key, k is a random secret key. The decryption of $E_k(f)$ is defined as:

$$f = D_k(c) = [c \times \;\; MK^{-1} - k] \;\; mod \; \alpha \tag{3.4}$$

where parameter MK^{-1} is the multiplicative inverse of MK modulo e. Assuming that c_1 and c_2 are the ciphertexts of feedback f_1 and f_2 under secret keys k_1 and k_2, respectively. The homomorphic property of the encryption supports direct computation on the ciphertexts as follows:

$$\begin{aligned} c_1 + c_2 &= E_{k_1}(f_1) + E_{k_2}(f_2) \\ &= [(f_1 + f_2) + (k_1 + k_2)] \times MK \;\; mod \; \alpha \\ &= E_{k_1 + \; k_2}(f_1 + \; f_2). \end{aligned} \tag{3.5}$$

The result can be decrypted using the key MK^{-1} and the corresponding sum of secret keys $k_1 + k_2$ as follows:

$$f_1 + f_2 = D_{k_1+k_2}(c_1 + c_2). \tag{3.6}$$

The property of additive homomorphic encryption shown in Equation (3.6) allows the enquirer to compute the encrypted feedback as if it were computed based on its plaintexts:

$$\mathcal{C}_{sum} = \sum_{l=1}^{l=n} E_{k_l}(f_l) = E_{\sum_{l=1}^{l=n} k_l}(\sum_{l=1}^{l=n} f_l). \tag{3.7}$$

That is $\mathcal{C}_{sum} = E_{SK}(\mathcal{R}_{sum})$ where k_l is the secret key of r_l used to encrypt the feedback f_l. $\mathcal{C}_{sum}$ is the encrypted sum of all raters' feedback. $SK = \sum_{l=1}^{l=n} k_l$ is the sum of secret keys privately held by each rater. To decrypt $\mathcal{C}_{sum}$ and recover the sum of feedback in plaintext $\mathcal{R}_{sum}$, the enquirer needs to obtain SK.

As indicated in the Equation (3.7), to obtain SK, the enquirer needs to ask all concerned raters to provide their secret keys. However, this requirement is unrealistic. Moreover, if the secret key of each feedback is disclosed to the enquirer, feedback privacy cannot be protected. Hence, we adopt secret sharing in [106, 107], which allows a rater to share its secret key through n pieces and with only at least m out of n pieces, the secret key can be recovered. We consider that the raters can share their secret keys with one another. When a certain number of raters is available to share the secret keys, the sum of the secret key SK can be determined.

Table 3.1: Notations used in the trust evaluation protocol

Notation	Description
CSP	Cloud service provider
FBS	Feedback storage
u_l	The lth user of CSP
r_l	The lth rater who gives feedback to CSP
f_l	The feedback given by rater r_l to CSP
$pubK_l, privK_l$	The public/private keys of rater r_l
k_l	The secret key of rater r_l for feedback encryption
k_{li}	The secret key share prepared by rater r_l for rater r_i
n	The number of raters in a rater set $\mathbb{R}$ of CSP
m	The minimum number of raters for secret key sum reconstruction
$g^{a_{lm-1}}, ..., g^{k_l}$	The m commitments for verifying the secret key share of rater r_l
$E_{k_l}(f_l)$	The encrypted feedback of rater r_l
$\mathcal{E}_{pubK_i}(k_{li})$	The encrypted secret key share prepared by rater r_l for rater r_i

3.4 Trust Evaluation Protocol

In this section, we discuss the trust evaluation protocol in detail, based on the aforementioned system model and building blocks. There are three phases in the trust evaluation protocol: rater registration, feedback submission and feedback reconstruction. The notations used in this section to describe the protocol are summarized in Table 3.1.

3.4.1 Rater Registration Phase

1) $\{\widetilde{Sign}(pubK_l)\} \leftarrow$ CSP. **A user requests for certification by a CSP.** To register with the FBS as a rater, a user $u_l \in \mathbb{U} = \{u_1, ..., u_n\}$ needs

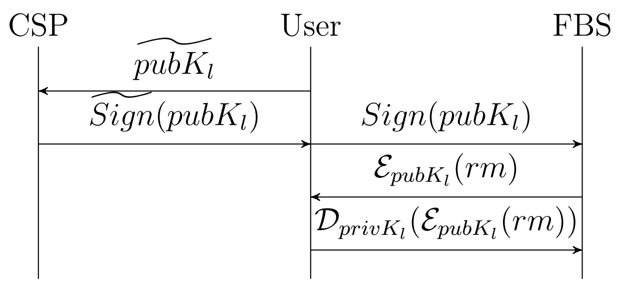

CSP
User
FBS
$\widetilde{pubK_l}$
$\widetilde{Sign}(pubK_l)$
$Sign(pubK_l)$
$\mathcal{E}_{pubK_l}(rm)$
$\mathcal{D}_{privK_l}(\mathcal{E}_{pubK_l}(rm))$

Fig. 3.2: Process of rater verification.

to provide the FBS with proof of identity. The user u_l first generates a pair of public/private keys $\{pubK_l, privK_l\}$ for the certification process. Then, the user u_l asks the CSP to sign the public key $\{pubK_l\}$ by means of a blind signature to prove the right to submit feedback. Basically, the user u_l first sends a "blinded" public key $\widetilde{pubK_l}$ to the CSP by combining it with a certain number of random factors. The CSP then returns the blinded signature $\widetilde{Sign}(pubK_l)$ to the user. After unblinding $\widetilde{Sign}(pubK_l)$, the user obtains the signature $Sign(pubK_l)$, which is proof of the rater registration. As previously mentioned, additional certification can be provided by means of a double blind signature (e.g., an additional blind signature by a payment system).

2) $\{pubK_l, Sign(pubK_l)\} \leftarrow u_l$. **The user verifies the identity with the FBS.** The user sends the signature $Sign(pubK_l)$ with the public key $pubK_l$ to the FBS for verification. The FBS verifies $Sign(pubK_l)$ with the public key of the CSP. Upon verification, the FBS continues to verify that user u_l has the private key of $pubK_l$ by asking u_l to decrypt a random message rm encrypted with $pubK_l$ (i.e., a challenge and response mechanism). If u_l can correctly decrypt the random message rm, it finally proves that u_l has the right to submit feedback so that the rater registration can be conducted. The verification process is shown in Fig. 3.2.

3) $\{\langle pubK_1, ..., pubK_n\rangle, \langle SSA_1, ..., SSA_n\rangle\} \leftarrow$FBS. **The user registers with the FBS as a rater.** Once a user is verified by the FBS, a new rater identity is created for the user. Due to the use of a blind signature (i.e., no linkage between $\widetilde{Sign}(pubK_l)$ and unblinded signature $Sign(pubK_l)$), the user's identity cannot be determined from the rater's identity. The rater also needs to provide a SSA for the secret sharing operation, which has no linkage with the CSP. After user registration, the FBS groups n registered raters together to form $\mathbb{R} = \{r_1, ..., r_n\}$ and publishes their public keys and SSAs. Note that there can be no linkage between an SSA and the corresponding rater, and the SSAs can be published in a group.

3.4.2 Feedback Submission Phase

In this phase, each rater generates a different secret key for the encryption of its feedback, and shares this secret key with all raters in the same random group/set $\mathbb{R}$ using the (m, n) Shamir's secret sharing method [106]. The secret key share process does not rely on the existing trust relationship between users, since their identities have been anonymized. When certain raters leak the privacy of honest raters, their own secret keys would leak at the same time. Thus, the privacy of raters' feedback is preserved. When a rater wants to update its feedback, it only uses the same secret key to encrypt its new feedback, without updating its secret key shares. Due to the homomorphic property of Shamir's secret sharing, given ith share of each secret key $F_1(i), ..., F_n(i)$, one can compute the ith share of the sum of n secret keys $F_1(x) + ... + F_n(x)$. With the distributed collaboration of m out of n raters, the sum of secrets can be reconstructed, where m is the threshold value.

1) $\{k_l, \langle k_{l1}, ..., k_{ln}\rangle, \langle g^{a_{lm-1}}, ..., g^{k_l}\rangle\} \leftarrow r_l$. **Each rater prepares secret key shares to protect the privacy of the feedback.** To implement

the above method, each rater in $\mathbb{R}$ privately constructs a different polynomial with the same degree of $m-1$, but different random coefficients, and a constant term, which represents the secret key of the rater. Each rater $r_l \in \mathbb{R}$ constructs a polynomial

$$F_l(x) = a_{lm-1}x^{m-1} + a_{lm-2}x^{m-2} + ... + a_{l1}x + k_l (\bmod\ \beta).$$

Rater r_l produces n points on $F_l(x)$ and one for each rater. That is, $k_{li} = F_l(i)$ is the secret key share prepared by rater r_l for rater r_i, $1 \leq i \leq n$. To allow each rater to verify that its secret key share is correctly prepared, we adopt a verifiable secret sharing scheme [27]. Each rater r_l also generates m commitments to the coefficients of $F_l(x)$, that is $\{g^{a_{lm-1}}, g^{a_{lm-2}}, ..., g^{a_{l1}}, g^{k_l}\}$. The public parameter g is the generator of a cyclic group. In this group, the discrete logarithm is difficult to compute. Rater r_l submits the m commitments $\{g^{a_{lm-1}}, ..., g^{k_l}\}$ to the FBS.

2) $\{\mathcal{E}_{pubK_1}(k_{l1}), ..., \mathcal{E}_{pubK_n}(k_{ln})\} \leftarrow r_l$. **A rater prepares the encrypted share of the secret key with the rest of the raters.** To prevent the FBS from directly observing the secret key shares, the rater encrypts each secret key share with the public key of the corresponding rater using Paillier encryption [26]. That is, $\mathcal{E}_{pubK_i}(k_{li})$ is the encrypted secret key share prepared by rater r_l for rater r_i. Rater r_l submits n encrypted shares of its secret key $\{\mathcal{E}_{pubK_1}(k_{l1}), ..., \mathcal{E}_{pubK_n}(k_{ln})\}$ to the FBS, as illustrated in Table 3.2.

3) $\{\prod_{l=1}^{l=n} \mathcal{E}_{pubK_1}(k_{l1}), ..., \prod_{l=1}^{l=n} \mathcal{E}_{pubK_n}(k_{ln})\} \leftarrow$ FBS. **The FBS computes the product of all encrypted key shares for each rater.** To facilitate the feedback reconstruction process, the FBS computes the product of Paillier-based encrypted key shares for each rater and publishes the product results, as shown in the gray parts of Table 3.2. Based on the additive homomorphic property of Paillier encryption, the encrypted key shares for the same rater can be multiplied together, which is equal to the

encrypted sum of key shares, that is $\prod_{l=1}^{l=n} \mathcal{E}_{pubK_i}(k_{li}) = \mathcal{E}_{pubK_i}(\sum_{l=1}^{l=n} k_{li})$.

4) $\{g^{\sum_{l=1}^{l=n} a_{lm-1}}, ..., g^{\sum_{l=1}^{l=n} k_l}\} \leftarrow$ FBS. **The FBS computes the product of all polynomial commitments for verification.** To allow any m of the n raters to verify the aggregated secret key shares, (i.e., $\sum_{l=1}^{l=n} k_{li}$ is correctly prepared), the FBS computes the product of all raters' polynomial commitments and publishes the product result i.e.,

$$\sum_{l=1}^{l=n} F_l(x) = \sum_{l=1}^{l=n} a_{lm-1}x^{m-1} + ... + \sum_{l=1}^{l=n} a_{l1}x + \sum_{l=1}^{l=n} k_l (\text{mod } \beta).$$

5) $E_{k_l}(f_l) \leftarrow r_l$. **A rater submits the encrypted feedback to the FBS.** To submit feedback, rater r_l encrypts the feedback to obtain $E_{k_l}(f_l)$ based on Equation (3.3). The rater then uploads the encrypted feedback to the FBS, as shown in the gray part of Table 3.2. The feedback submission process is shown in Fig. 3.3.

6) $\{E_{k_l}(f_l^{'}), \langle \mathcal{E}_{pubK_1}(k_{l1}^{'}), ..., \mathcal{E}_{pubK_n}(k_{ln}^{'})\rangle, \langle g^{d_{lm-1}}, ..., g^{d_{l1}}, g\rangle\} \leftarrow r_l$. **A rater updates the feedback and secret key shares.** A rater can update the feedback by resubmitting new feedback $f_l^{'}$ encrypted using the same secret key k_l to the FBS, that is, to use $E_{k_l}(f_l^{'})$ to replace the old one. To prevent malicious users from colluding to recover the secret keys of honest raters over time, each rater r_l can periodically update the secret key shares by generating a new polynomial $F_l^{'}(x)$ with random coefficients, but a constant term of zero. As a result, an attacker must collude with enough raters within a certain time frame to successfully recover the secret key.

$$F_l^{'}(x) = d_{lm-1}x^{m-1} + ... + d_{l1}x + 0 (\text{mod } \beta).$$

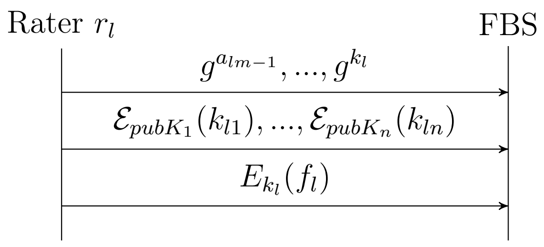

Rater r_l
FBS
$g^{a_{lm-1}}, ..., g^{k_l}$
$\mathcal{E}_{pubK_1}(k_{l1}), ..., \mathcal{E}_{pubK_n}(k_{ln})$
$E_{k_l}(f_l)$

Fig. 3.3: Process of feedback submission.

For each remaining rater, a new point on this new polynomial $F_l^{'}(x)$ is computed, that is, $k_{li}^{'} = F_l^{'}(i)$. Each rater then adds the old secret key share with the new point to obtain the new secret key share, that is, $k_{li}^{1} = k_{li} + k_{li}^{'} = F_l(i) + F_l^{'}(i)$. Thus, each rater r_l can update secret key shares by submitting n new encrypted secret key shares to the FBS, that is $\{\mathcal{E}_{pubK_1}(k_{l1}^{'}), ..., \mathcal{E}_{pubK_n}(k_{ln}^{'})\}$. At the same time, rater r_l updates its polynomial commitments by sending $\{g^{d_{lm-1}}, ..., g^{d_{l1}}, g\}$ to the FBS. Note that the FBS will verify the rater's identity in each submission. After a certain period of time, all submitted encrypted feedback and secret key shares need to be dismissed and can no longer be used. To avoid breaking anonymity and privacy, raters need to generate new secret keys for new feedback, and upload them to FBS.

3.4.3 Feedback Reconstruction Phase

1) $\{E_{k_1}(f_1), ..., E_{k_n}(f_n)\} \leftarrow$ FBS. **An enquirer gets the encrypted feedback from the FBS.** To evaluate the trustworthiness of a cloud service, an enquirer sends a request to the FBS for the feedback provided by raters. Upon receiving a reply from the FBS, the enquirer computes the encrypted sum of the raters' feedback by adding it together, that is $\mathcal{C}_{sum} = \sum_{l=1}^{l=n} E_{k_l}(f_l) = E_{\sum_{l=1}^{l=n} k_l}(\sum_{l=1}^{l=n} f_l)$.

2) Request for decryption $\leftarrow$ Enquirer. **The enquirer requests the raters to decrypt the assigned secret key shares.** To decrypt the

Table 3.2: Feedback submission and reconstruction

	Raters	Encrypted Feedbacks	Secret keys ($k_{li} = F_l(i)$)	r_1	...	r_n
Section 3.4.2	r_1	$E_{k_1}(f_1)$	$F_1(x) = a_{1m-1}x^{m-1} + ... + k_1$	$\mathcal{E}_{pubK_1}(k_{11})$	...	$\mathcal{E}_{pubK_n}(k_{1n})$
		$+$	$+$	$\times$		$\times$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	r_l	$E_{k_l}(f_l)$	$F_l(x) = a_{lm-1}x^{m-1} + ... + k_l$	$\mathcal{E}_{pubK_1}(k_{l1})$	...	$\mathcal{E}_{pubK_n}(k_{ln})$
		$+$	$+$	$\times$		$\times$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	r_n	$E_{k_n}(f_n)$	$F_n(x) = a_{nm-1}x^{m-1} + ... + k_n$	$= \prod_{l=1}^{l=n} \mathcal{E}_{pubK_1}(k_{l1})$	...	$= \prod_{l=1}^{l=n} \mathcal{E}_{pubK_n}(k_{ln})$
Section 3.4.3		$+ \Downarrow$	$+ \Downarrow$	$\Downarrow$		$\Downarrow$
		$= E_{\sum_{l=1}^{l=n} k_l}(\sum_{l=1}^{l=n} f_l)$	$= \sum_{l=1}^{l=n} a_{lm-1}x^{m-1} + ... + \sum_{l=1}^{l=n} k_l$	$= \mathcal{E}_{pubK_1} \sum_{l=1}^{l=n} k_{l1}$	...	$= \mathcal{E}_{pubK_n} \sum_{l=1}^{l=n} k_{ln}$
		Dec $\Downarrow$	Verify $\Downarrow g^{\sum_{l=1}^{l=n} a_{lm-1}}, ..., g^{\sum_{l=1}^{l=n} k_l}$	$\mathcal{D}_{privK_1}() \Downarrow$		$\mathcal{D}_{privK_n}() \Downarrow$
		$\boldsymbol{\mathcal{R}_{sum} = \sum_{l=1}^{l=n} f_l}$	$\Leftarrow \sum_{l=1}^{l=n} k_l \Leftarrow m$ out of n	$\sum_{l=1}^{l=n} k_{l1} = \sum_{l=1}^{l=n} F_l(1)$	...	$\sum_{l=1}^{l=n} k_{ln} = \sum_{l=1}^{l=n} F_l(n)$

encrypted sum of feedback, the enquirer needs to reconstruct the sum of the raters' secret keys $\sum_{l=1}^{l=n} k_l$. The enquirer progressively sends requests to the raters based on SSAs provided/published by the FBS. In each iteration, the minimum number of raters are chosen and contacted. For example, in the first iteration, m raters are contacted. If only x of them reply, $m-x$ raters will be contacted in the second iteration. Alternatively, the enquirer can broadcast requests to all raters but some replies may be wasted as only m of them are required.

3) $\mathcal{D}_{privK_i}(\prod_{l=1}^{l=n} \mathcal{E}_{pubK_i}(k_{li})) \leftarrow r_i$. **Each rater independently decrypts the product of respective secret shares, and forwards it to the enquirer.** Rater r_i who is willing to contribute to secret key reconstruction gets the product result of the encrypted secret key shares $\prod_{l=1}^{l=n} \mathcal{E}_{pubK_i}(k_{li})$ from the FBS. Due to the additive homomorphic property, $\prod_{l=1}^{l=n} \mathcal{E}_{pubK_i}(k_{li}) = \mathcal{E}_{pubK_i}(\sum_{l=1}^{l=n} k_{li})$. r_i decrypts it using the private key $privK_i$ to obtain $\sum_{l=1}^{l=n} k_{li}$ and sends it to the enquirer. The process of secret key share collection is illustrated in Fig. 3.4. Note that an honest rater will not decrypt any information other than the encrypted sum of secret key shares assigned to it.

4) Secret share verification $\leftarrow$ Enquirer. **The enquirer verifies the secret key shares with the polynomial commitments.** The enquirer obtains the product of all polynomial commitments from the FBS. The enquirer then computes the following value for each rater r_i

$$\begin{aligned}
&(g^{\sum_{l=1}^{l=n} a_{lm-1}})^{i^{m-1}} (g^{\sum_{l=1}^{l=n} a_{lm-2}})^{i^{m-2}} \ldots (g^{\sum_{l=1}^{l=n} k_l})^{i^0} \\
&= g^{\sum_{l=1}^{l=n} a_{lm-1} i^{m-1} + \sum_{l=1}^{l=n} a_{lm-2} i^{m-2} + \ldots + \sum_{l=1}^{l=n} k_l} \\
&= g^{\sum_{l=1}^{l=n} F_l(i)}.
\end{aligned}$$

Accordingly, using the public parameter g and received $\sum_{l=1}^{l=n} k_{li}$, the enquirer computes $g^{\sum_{l=1}^{l=n} k_{li}} (\text{mod } \beta)$. By comparing $g^{\sum_{l=1}^{l=n} F_l(i)} (\text{mod } \beta)$ and

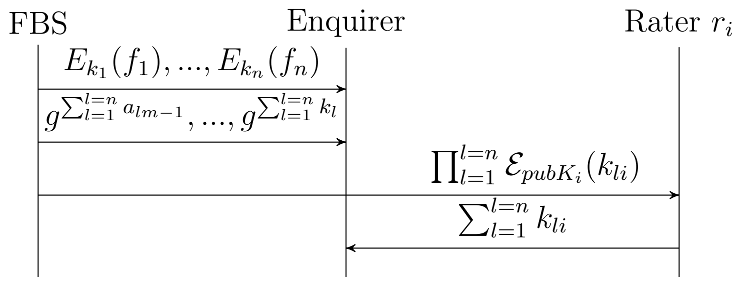

FBS
Enquirer
Rater r_i
$E_{k_1}(f_1), ..., E_{k_n}(f_n)$
$g^{\sum_{l=1}^{l=n} a_{lm-1}}, ..., g^{\sum_{l=1}^{l=n} k_l}$
$\prod_{l=1}^{l=n} \mathcal{E}_{pubK_i}(k_{li})$
$\sum_{l=1}^{l=n} k_{li}$

Fig. 3.4: Process of secret key share collection.

$g^{\sum_{l=1}^{l=n} k_{li}} (\text{mod } \beta)$, the enquirer can verify whether $\sum_{l=1}^{l=n} k_{li}$ can correctly reconstruct the sum of all raters' secret keys.

5) $\sum_{l=1}^{l=n} k_l \leftarrow$ Enquirer. **The enquirer reconstructs the raters' encrypted secret keys.** When there are at least m successful secret share verifications (i.e., from m raters), the sum of n raters' secret keys can be reconstructed using the Lagrange interpolation method, that is, $\sum_{l=1}^{l=n} k_l$.

6) $\sum_{l=1}^{l=n} f_l \leftarrow$ Enquirer. **The enquirer decrypts the encrypted sum of feedback.** Finally, using the secret keys, the enquirer can decrypt $\mathcal{C}_{sum}$ to obtain $\mathcal{R}_{sum} = \sum_{l=1}^{l=n} f_l$. By converting the respective binary digits of $\mathcal{R}_{sum}$ to the corresponding decimal numbers, the number of raters for each rating choice can be determined. The enquirer finally obtains the trust evaluation result using different evaluation weights: $\mathcal{T}r = (\sum_{j=1}^{j=z} w_j \times |b_j| \times j)/n$.

3.5 Security Evaluation

We analyze the security of our proposed Intercloud trust evaluation protocol based on the widely used simulation paradigm [108] in a manner similar to that seen in [109]. In this paradigm, a system is secure if its output distribution approximates an ideal system for all possible input distributions. More specifically, we first define an ideal system where all computations are conducted by a trusted party, $\mathcal{T}$. Furthermore, all participants communicate through $\mathcal{T}$ via a secure communication channel. As such, this hypothetical ideal system is secure, assuming $\mathcal{T}$ is honest. A real-world system is secure if the output distribution of the system is the same as that of an ideal system for all possible input sequences. A complete security analysis in this paradigm then consists of two parts, namely, a security model and security proof. The former gives the definition of an ideal system, and the latter asserts that the input-output distribution of a real-world system is the same as that of the ideal system.

3.5.1 Security Model

An Intercloud trust evaluation protocol consists of rater registration, feedback submission and feedback reconstruction phases with relevant parties being the FBS, CSP, users/raters and enquirers. We give the specifications of the ideal-world system as follows.

- **Rater registration** Each user $u \in \mathbb{U} = \{u_1, ..., u_n\}$ sends a request to register as a rater for a CSP to $\mathcal{T}$, who forwards the request to the CSP concerned. The response from the CSP is sent back through $\mathcal{T}$. Upon receiving approval, u sends another request to $\mathcal{T}$ for the registration as a rater with FBS. $\mathcal{T}$ first checks whether the request is legitimate, based on the CSP response. If the request is valid, $\mathcal{T}$ notifies FBS that one eligible user is requesting to give feedback for this particular CSP. Note that $\mathcal{T}$ will not reveal the identity of u to FBS. If FBS accepts this request, $\mathcal{T}$ creates a new rater identity r for u and informs u and FBS. FBS randomly groups raters together to obtain a set of raters $\mathbb{R} = \{r_1, ..., r_n\}$. The set, $\mathbb{R}$, is made known to all raters within the set through $\mathcal{T}$.

- **Feedback Submission** Each rater $r \in \mathbb{R} = \{r_1, ..., r_n\}$ sends a request to $\mathcal{T}$ for rater set $\mathbb{R}$ that includes r. $\mathcal{T}$ first checks whether the requester is rater r. Once it passes the check, $\mathcal{T}$ returns $\mathbb{R}$ to it. Rater acknowledges set $\mathbb{R}$ to T. $\mathcal{T}$ informs the FBS that rater r's $\mathbb{R}$. After a period of time, each rater r submits its feedback f to $\mathcal{T}$. Then, after $\mathcal{T}$ confirms the requester is rater r, it will store the feedback and inform FBS that rater r has submitted feedback, but without informing FBS about the content of the rater's rating choice.

- **Feedback Reconstruction** An enquirer asks $\mathcal{T}$ for the CSP feedback given by its past users. $\mathcal{T}$ forwards the enquirer's request to all raters in the $\mathbb{R}$ to ask whether they support the reconstruction of the feedback results. When there is a sufficient number of approval responses from raters, $\mathcal{T}$ calculates the sum of the feedback given by raters in the group $\mathbb{R}$ and returns this sum to the enquirer. Then $\mathcal{T}$ informs FBS that an enquirer has obtained the sum of the feedback in the group $\mathbb{R}$.

Upon completion of the three phases, all participants output the protocol outcome in each of the above three phases. Obviously, the system in the ideal world is secure, assuming $\mathcal{T}$ follows the specifications completely and that communication between $\mathcal{T}$ and the participants is secure. In particular, CSP and FBS will not be able to obtain individual feedback. CSP also cannot learn the raters' identity. To show that our proposed Intercloud trust evaluation protocol achieves its security goals, it suffices to prove that for all possible execution sequences and all possible inputs, the output distribution of our proposed system is the same as the above ideal system.

Following the above intuition, we define the security of an Intercloud trust evaluation protocol as follows. Let $\vec{x}$ denote the set of inputs for all participants. Let $\mathcal{M}$ denote the set of dishonest participants controlled by adversary $\mathcal{A}$. To be more specific, we use $\mathcal{A}_I$ and A_R to denote an adversary in the ideal and the real worlds respectively. Honest participants are denoted as $\mathcal{H}$. While $\mathcal{H}$ follows the protocol faithfully, $\mathcal{A}$ may act arbitrarily. Let $REAL_{\mathcal{A}_R}(\kappa, \vec{x})$ (*resp.* $IDEAL_{\mathcal{A}_I,\mathcal{S}}(\kappa, \vec{x})$) denote the probability distribution of joint outputs of honest clouds and adversary $\mathcal{A}_R$ (*resp.* adversary $\mathcal{A}_I$) on inputs $\vec{x}$ and security parameter κ in the real world (*resp.* in the ideal world).

Definition 1 *An Intercloud trust evaluation protocol is secure, if for all probabilistic polynomial time adversaries $\mathcal{A}_R$, there exists a corresponding $\mathcal{A}_I$ in the ideal world such that:*

$$\{REAL_{\mathcal{A}_R}(\kappa, \vec{x})\}_{\kappa \in N} \approx \{IDEAL_{\mathcal{A}_I}(\kappa, \vec{x})\}_{\kappa \in N} \tag{3.8}$$

where $\approx$ represents indistinguishability of two distributions.

3.5.2 Security Proof

The goal of the security proof is to show that for all $\mathcal{A}_R$, there exists a corresponding $\mathcal{A}_I$. To accomplish the goal, we construct a simulator, $\mathcal{S}$, who plays the role of all honest parties in the view of $\mathcal{A}_R$, and plays the role of $\mathcal{A}_I$ in the ideal world. That is, $\mathcal{S}$ represents the honest parties to interact with $\mathcal{A}_R$ in the real world, in order to learn about the attacks of $\mathcal{A}_R$. At the same time, $\mathcal{S}$ represents the dishonest users/raters, CSP, FBS and enquirer to interact with $\mathcal{T}$ in the ideal world. Then, we are going to show that the input-output distribution of the two worlds is indistinguishable. Based on the adversary model analyzed in Section 3.2.3, in the real world, adversary $\mathcal{A}_R$ might corrupt all participants (i.e., users/raters, CSP, FBS and enquirer) in all three phases of our protocol.

Below we discuss the various attacks presented in our adversary model (Section 3.2.3) and that if successful, would cause the outputs of the ideal and real world to be distinguishable. Then, we present arguments that these cases can only occur with negligible probability.

- **Rater registration**

  1) *(Attack 1) Malicious user in the real world who does not obtain the signature assigned by CSP can register as a rater and submit misleading feedback.* In the ideal world, $\mathcal{T}$ will check CSP's approval for a rater registration request, and thus this attack cannot happen. In our protocol, we assume that FBS is reliable for rater verification and needs to verify the signature with a CSP public key. Consequently, a malicious user only becomes a rater if it can forge a signature of the CSP. We would like to remark that it cannot become a rater by stealing signatures from valid users. The reason is that in our protocol, FBS will validate whether the signature holder is the real user by a challenge-response protocol in which the rater is required to decrypt the ciphertext of a random message. Since a malicious user does not have the private key of the real signature holder, it cannot decrypt the message and cannot pass the check on FBS. As a result, this scenario will not occur.

  2) *(Attack 3a) Malicious FBS maintains raters' information, which violates business privacy. FBS can help CSP identify raters that gave negative feedback.* In the ideal world, all users' user identity information is kept secret by $\mathcal{T}$ and thus, this attack cannot occur. In our protocol, we adopt a blind signature to prevent this from happening in the real world. Specifically, possession of an unblinded signature only demonstrates that the signature holder is a legitimate user of the CSP, but it cannot be linked to the specific blind signature generation.

- **Feedback Submission**

  1) *(Attack 2a) Selfish raters may prepare incorrect secret sharing information and submit it to the FBS.* In the ideal world, $\mathcal{T}$ stores a single feedback value and seeks approval from raters before releasing it. In other words, this attack cannot happen. In our protocol, we adopt verifiable secret sharing to prevent this from happening. Specifically, anyone can verify whether the secret key shares of each rater has been correctly prepared. As such, this attack also cannot happen in the real world.

  2) *(Attack 3b) Malicious FBS may help CSP to identify raters that gave negative feedbacks.* Malicious CSPs and FBS colludes to obtain actual value of individual feedback. In the ideal world, all feedback is maintained by $\mathcal{T}$. FBS only knows whether a rater has provided its rating or not, and thus this attack cannot happen. In our protocol, we adopt homomorphic encryption, so that all feedback and secret key shares are encrypted by the secret key and public key of honest raters. Therefore, only the honest raters themselves can decrypt it. The FBS cannot directly deduce the plaintext of these encrypted values and in the FBS's view, these values are indistinguishable from random values. Thus, this attack also cannot happen in the real world.

- **Feedback Reconstruction**

  1) *(Attack 2b) Selfish raters refuse the enquirer's request for secret key reconstruction, such that the enquirer cannot reconstruct the feedback sum.* This attack can occur in both the real world and the ideal world, and depends on the number of honest raters versus selfish raters. In the next section, we present an analysis to show

feedback reconstructions that fail with very low probability under reasonably chosen parameters.

2) *(Attack 4a) Malicious CSP pretends to be an enquirer to ask for honest raters to decrypt the individual secret key share of a specific rater so as to obtain its individual feedback.* In the ideal world, $\mathcal{T}$ checks the identity of the enquirer to prevent this from happening. In our protocol, this will also not happen, since an honest rater will refuse this kind of request from an enquirer.

3) *(Attack 4b) Malicious CSP colludes with FBS and n raters.* As long as n is smaller than the threshold, no information is leaked since secret reconstruction is impossible. In the case that n is greater than or equal to the threshold, the real world adversary $\mathcal{A}_R$ can reconstruct the secret key of honest raters and obtain individual feedback f without being noticed. In the ideal world, honest raters will be notified by $\mathcal{T}$ of the fact that someone attempted to reconstruct feedback. In this case, the two world will be distinguishable. We define event fail as the scenario where $\mathcal{A}_R$ obtains individual feedback without being noticed.

In conclusion, the only case in which the outputs of the real and ideal worlds are distinguishable is when event fail occurs in the feedback reconstruction phase. In the next section, we will analyze the probability of a successful attack of this kind, in which the adversary obtains individual feedback from our protocol.

3.6 Simulation Settings

In this section, we conduct simulations to evaluate our protocol, focusing on the security aspects. In particular, the aim is to evaluate how the protocol can resist collusion attack from malicious raters and maintain the availability of trust results with the selfish raters.

Choice of Parameters. We first use a weighted directed graph $G(N, p, q)$ to describe the possible connections between users and CSPs in the Intercloud scenario. A set of vertices N in the graph G represents the set of users/raters and CSPs in the Intercloud. Note that each vertex can be a user as well as a CSP. When a user interacts with a CSP, a directed edge will be added from the cloud vertex to the user vertex on the graph. We define parameter $p \in [0\%, 100\%]$ as the density of graph G, which is the ratio between the actual number of edges and the maximum number of edges in the graph G (i.e., $N(N-1)$). A larger p indicates there are more interactions between users and CSPs. And the weight of each edge in the graph is the feedback given by a rater to a CSP. We define the parameter $q \in [0, 100]$ as the average feedback given by a rater to a CSP, which is also the average weight of an edge in the graph. A larger q shows that the rater tends to give more positive feedback to the CSP. Fig. 3.5 shows a simple example of graph G. There are three vertices, each acting as CSP and user at the same time (i.e., $N = 3$). There are four edges. Thus, the graph density is $p = \frac{4}{6} = 67\%$. Each edge weight is feedback given by a rater to a CSP. In this graph, the average feedback is the average weight of all edges $q = \frac{(100+80+60+40)}{4} = 70$. We assume that all raters adopt our protocol to protect the privacy of their feedback. By labeling a portion of raters as malicious or selfish, we evaluate the privacy protection, trust result availability and efficiency of our protocol. Three graphs, denoted by Advogato, Robots, Random, are used in our simulation. Their characteristics are summarized in Table 3.3.

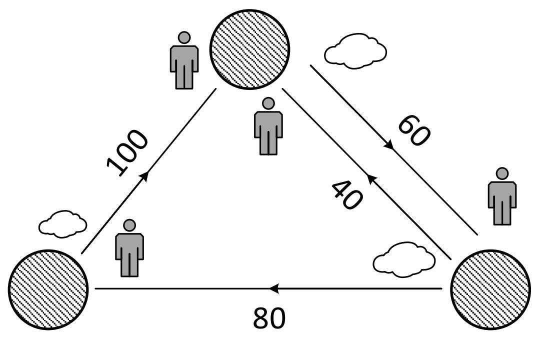


Fig. 3.5: Example of trust graph.

Table 3.3: Characteristics of trust graphs

	Advogato	**Robots**	**Random**	
			Random-a	**Random-b**
Vertices N	5,417	1,732	5,417	1,732
Density p	0.17%	0.12%	0.17%	[10%, 100%]
Avg. feedback q	80.12	70.46	[10, 100]	70.46

Trust graphs. There is more mutual co-operation in the Intercloud environment. However, current trust feedback datasets about cloud service only collect feedback given from users to CSPs, which is not a mutual evaluation or two-way trust/service relationship. Currently, no dataset about the two-way Intercloud relationship is available. Thus, we choose to evaluate our protocol based on two trust graphs Advogato and Robots which are commonly adopted in the analysis of any trust-related protocols [72]. These two datasets include a large number of mutual evaluations records, which is close to the two-way trust/service relationship between CSPs and users in the Intercloud environment. For instance, in the Advogato community, each member is a free software developer, and evaluates other developers with different rating levels. Robots community follows the same trust metric. The rating choices are *master*, *journeyer*, *apprentice*, and *observer*, in which *master* is the highest level and *observer* is the lowest or default level for a new account. We downloaded the latest dataset (Jul 07, 2014) of these communities from trustlet.org and used them to build graphs Advogato and Robots respectively, representing the possible relationships between users and CSPs in the Intercloud environment.

Random graph. As shown in Table 3.3, the graph density and average weight of the two trust graphs Advogato and Robots are similar. To supplement our simulation on the influences of graph density and average feedback, we also use another two graphs (denoted by Random-a and Random-b) with various parameters in our simulations. These two graphs are generated by randomly connecting between vertices (i.e., CSPs and users) and randomly assigning different weights (i.e., feedback) on edges. It is reasonable that several large CSPs would serve as ”hubs” of the Intercloud. They have many more users than the rest of the CSPs. Thus, we model the Intercloud as a scale-free network in the random graph. We generate a random variable with Pareto distribution, which represents the outdegree of the vertex (i.e., the number of users interacting with the same CSP). For each outgoing edge, we randomly pick a vertex as its destination. We consider that the majority of raters would choose the middle rating levels (e.g., *apprentice* and *journeyer*). And a small number of raters choose very low or very high rating levels (e.g., *observer* and *master*) as their feedback to the CSPs. Thus, the weights of edges in the random graph (i.e., feedback given by raters to CSPs) are assigned with a random variable following a normal distribution.

Random-a is a supplement to Advogato to evaluate protocol performance under various average feedback. Random-b supplements Robots to analyze the influence of density conditions. For Advogato and Robots, we build the graphs from the dataset after a pre-processing step that converts the rating level into the weight of the edges, by the following rule: *master* level = 100, *journeyer* level = 80, *apprentice* level = 60 and *observer* level = 40. Then each cloud has a different size of n users, m of which is the threshold number for feedback reconstruction. In subsequent simulations, we vary m/n ratio $\in [0.1,\ 1]$, with an increment of 0.1.

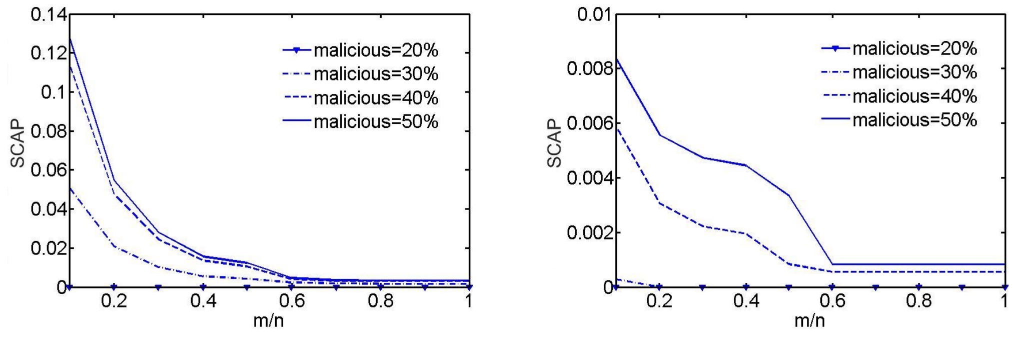
3.7 Simulation Results

3.7.1 Protection Against Collusion Attack

In this subsection, **we evaluate the relationship between m/n and successful collusion attack probability**, under different percentages of malicious raters. We calculate successful collusion attack probability as the percentage of honest raters whose feedback is leaked during the attack. A collusion attack is launched by malicious raters. They intentionally leak the secret key share information of honest raters. When a malicious CSP colludes with at least the threshold number of raters, it can illegally reconstruct the secret key to decrypt the individual feedback of honest raters. In the security analysis in Section 3.5.2, we have defined this as an event failure, and if this happens, the real world and the ideal world are distinguishable. Thus, the first simulation seeks to analyze the probability of this event failure in a practical setting. We use successful collusion attack probability to represent this event failure probability.

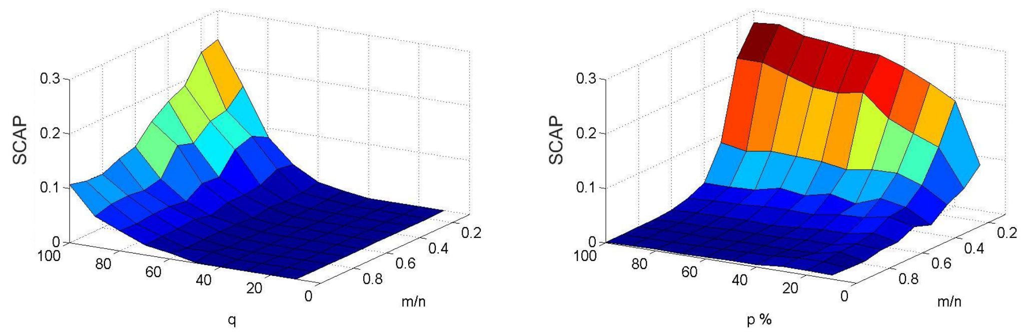
Raters in the network with a low reputation score are considered to be malicious. They have a higher possibility of colluding with each other to recover the feedback of honest raters. For the trust graphs, we set the malicious raters with reputations ranked in the lowest $\{20\%, 30\%, 40\%, 50\%\}$ of the entire network. To test the worst case scenario, we set 50% of raters as malicious in the random graphs. In Random-a, we fix $p = 0.17\%$ (same density of Advogato) and change q from 10 to 100, with an increment of 10. We use $q = 70.46$ for Random-b (same average feedback of Robots) and vary its p from 10% to 100%, with an increment of 10%.

The results for the trust and random graphs are shown in Fig. 3.6 and Fig. 3.7, respectively. It can be seen that even if the network contains 50% malicious raters, the successful collusion attack probability (i.e., the probability of event failure) will be less than 0.1 as long as $m/n > 20\%$ and $q < 90$ (i.e., the average feedback values in the entire network). In the design of our protocol, each



(a) Advogato, $q = 80.12$. (b) Robots, $p = 0.12\%$.

Fig. 3.6: Effect of increasing m/n ratio on the successful collusion attack probability (SCAP) in trust graphs.



(a) Random-a, $q \in [\ 10,\ 100\ ]$. (b) Random-b, $p \in [\ 10\%,\ 100\%\ ]$.

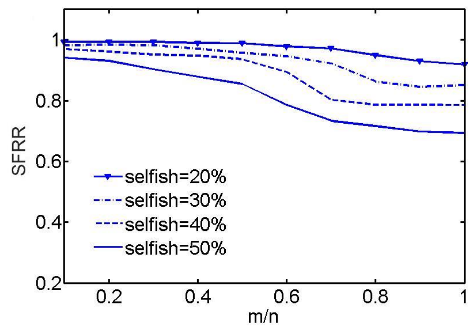
Fig. 3.7: Effect of increasing m/n ratio on the successful collusion attack probability (SCAP) in random graphs.

rater relies on other raters in the same set to share its secret key. When some of the raters collude to recover other raters' secret keys, their own secret keys are also disclosed. We consider that the majority of raters seek to protect their privacy. In addition, to recover raters' private feedback, CSPs need to inject a high enough number of malicious raters. Since these are real companies, many should have good corporate governance. The real-world collusion attack probability should be lower than the above results. Hence, the proposed protocol should be effective in handling collusion attacks.

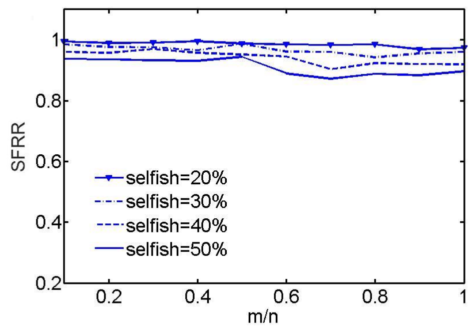
3.7.2 Feedback Recovery Rate of Our Protocol

In this subsection, **we discuss the relationship between m/n and the successful feedback recovery rate**, under different percentages of selfish raters. The decryption of the secret key sum relies on the support of raters. This simulation seeks to study whether the protocol can still accurately compute the trust result when there are selfish raters who are not willing to participate in the secret key sum reconstruction. Let $|\mathcal{T}r|$ $(w_j = 1)$ denote the number of CSPs' trust results reconstructed from all raters in the network. Let $|\mathcal{T}r'|$ $(w_j = 1)$ denote the number of CSPs' trust results recovered only from the raters who are not selfish. We define a successful feedback recovery rate as $\frac{|\mathcal{T}r'|}{|\mathcal{T}r|}$. In the following simulations, selfish raters are randomly assigned. For the trust graphs, we run with selfish raters $\in \{20\%, 30\%, 40\%, 50\%\}$. For the random graphs, we consider that 50% of raters in the network are selfish. Likewise, we evaluate the protocol recovery rate under various average feedback and network density by setting $q \in [10, 100]$ and $p \in [10\%, 100\%]$.

The feedback recovery rates of the protocol for the trust and random graphs are illustrated in Fig. 3.8 and Fig. 3.9, respectively. It can be seen that the higher the network density p, the higher average feedback q and the greater m/n all lead to a lower feedback recovery rate. Under the same percentage of selfish raters in the network, the network density p has more influence than parameter q in the successful feedback recovery rate, as shown in Fig. 3.9. In addition, when there are 50% of selfish raters in a network, the successful feedback recovery rate will be more than 0.77 as long as $m/n <= 0.5$.

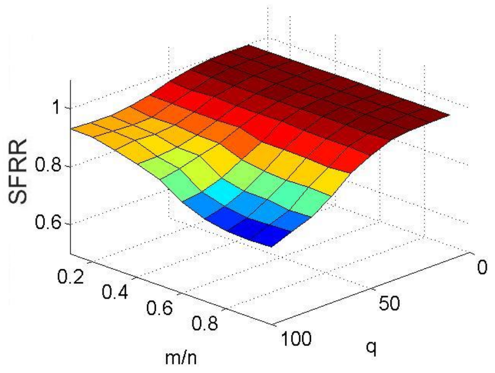


(a) Advogato, $q = 80.12$.

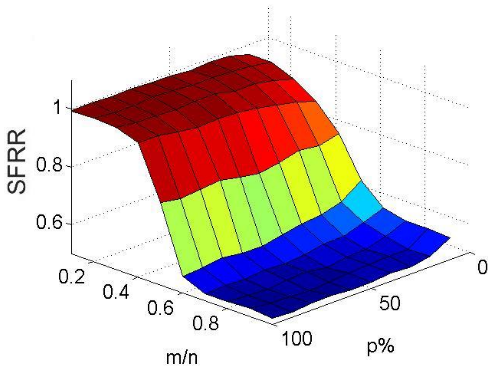


(b) Robots, $p = 0.12\%$.

Fig. 3.8: Effect of increasing m/n ratio on successful feedback recovery rate (SFRR) in trust graphs.



(a) Random-a, $q \in [\ 10,\ 100\ ]$.

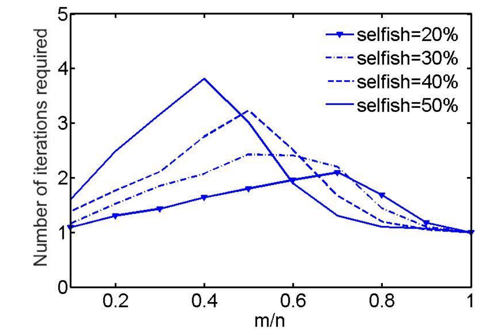


(b) Random-b, $p \in [\ 10\%,\ 100\%\ ]$.

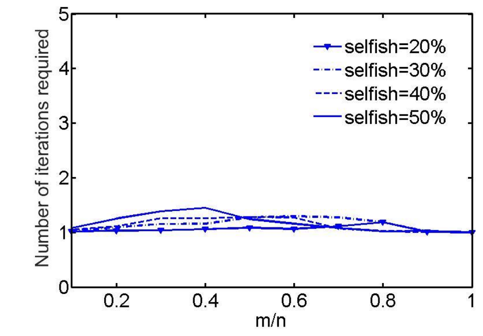
Fig. 3.9: Effect of increasing m/n ratio on successful feedback recovery rate (SFRR) in random graphs.

3.7.3 Feedback Recovery Efficiency of Our Protocol

In this subsection, **we analyze the relationship between m/n and the efficiency of feedback recovery**, under different percentages of selfish raters. We evaluate the efficiency of feedback recovery as the number of iterations required to obtain the secret key shares during the enquiry phase. An enquirer can progressively ask raters for the decrypted secret key shares. Ideally, only one iteration is required if m raters reply to the request. If some raters are selfish, more iterations are needed until m replies are collected to reconstruct the feedback sum. Therefore, the number of iterations required indicate the

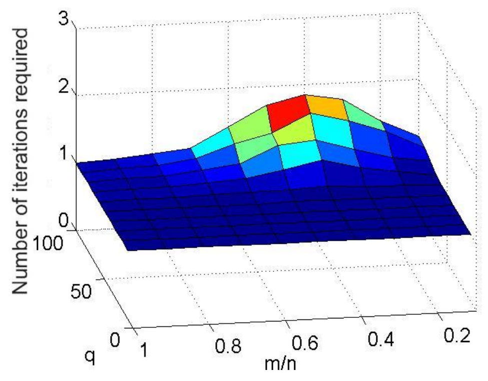


(a) Advogato, $q = 80.12$.

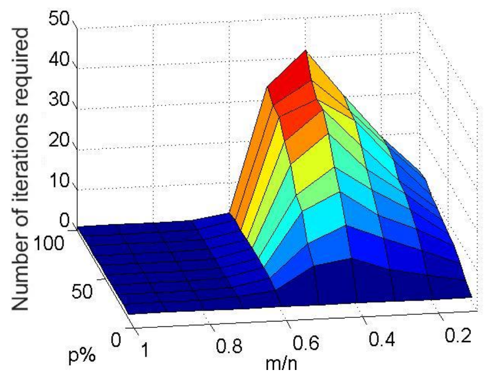


(b) Robots, $p = 0.12\%$.

Fig. 3.10: Effect of increasing m/n ratio on the average number of iterations required to obtain the secret key shares during the query phase in trust graphs.



(a) Random-a, $q \in [\ 10,\ 100\ ]$.



(b) Random-b, $p \in [\ 10\%,\ 100\%\ ]$.

Fig. 3.11: Effect of increasing m/n ratio on the average number of iterations required to obtain the secret key shares during the query phase in random graphs.

efficiency. In the simulations, the number of iterations required to recover the feedback sum of each CSP is determined so that the overall average can be computed. Similar to Section 3.7.2, we randomly choose selfish raters $\in \{20\%, 30\%, 40\%, 50\%\}$ of the entire network when running the simulation in the trust graph. For the random graphs, we assume 50% selfish raters.

We present the recovery efficiency of our protocol in Fig. 3.10 and Fig. 3.11. Advogato and Random-a show the similar trend in Fig. 3.10 (a) and Fig. 3.11 (a). Whereas, the peak number of iterations required in Random-b is much higher than that in the other three graphs as shown in Fig. 3.11 (b). Comparing the result of Random-a and Random-b, it can be seen that the network density p has more influence on the feedback recovery efficiency. When the network

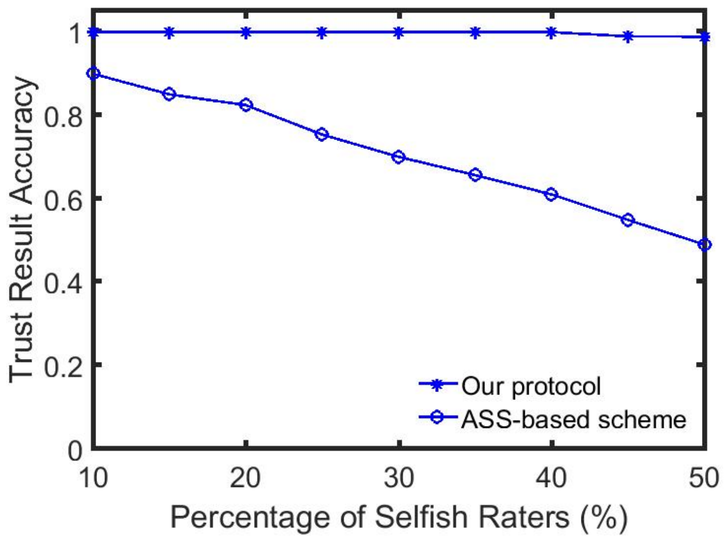


Fig. 3.12: Trust result accuracy comparison of our protocol and ASS-based scheme.

density $p \geq 10\%$, the enquirer needs more iterations to obtain the feedback sum (i.e., less efficient). The decline shown in the figures indicates that the trust result of some CSPs cannot be obtained when the number of selfish raters and m/n is large as discussed in Section 3.7.2. By considering the Intercloud configuration and security requirements, a suitable threshold value of m can be chosen.

3.7.4 Trust Result Accuracy Comparison

In this subsection, **we compare the trust result accuracy of our protocol with the scheme that uses additive secret sharing (ASS)**. Clark et al. [69] and Hasan et al. [72] both used ASS in their schemes to protect feedback privacy. Given a secret f, an ASS of f consists of n shares $\{f_1, ..., f_n\}$, where $f_1, ..., f_{n-1}$ are randomly chosen and $f_n = f - \sum_{i=1}^{i=n-1} f_i$. To accurately recover the secret, it simply adds all of the secret shares together.

ASS requires all raters to stay online to enable the computation of trust results. When any rater leaves the network or refuses to reply, all corresponding feedback shares will be lost in the trust result, causing inaccurate trust results.

To perform the analysis, let $\mathcal{T}r$ denote the trust result of each CSP computed from all of its raters. Let $\mathcal{T}r'$ denote the trust result of each CSP computed from the raters who are not selfish. We consider the trust result accuracy as the average value $\frac{\mathcal{T}r'}{\mathcal{T}r}$ of all CSPs whose trust results can be recovered. In the subsequent simulations, we randomly choose selfish raters $\in [5\%,\ 50\%]$, with an increment of 5% in the Advogato trust graph. We set $m/n = v/n = 0.4$ in our protocol and in ASS approach, such that the majority of feedback can be recovered (v is the number of feedback shares prepared by each rater in Hasan et al.'s scheme).

Fig. 3.12 shows the comparison of trust result accuracy on the Advogato trust graph. We observe that the accuracy of the ASS approach decreases with an increased percentage of selfish raters. However, the accuracy of our protocol can be maintained at a high level. This is because in our protocol, an enquirer only needs to request $m < n$ raters to reconstruct an accurate trust result. Therefore, our protocol is less affected by the percentage of selfish raters, unless the percentage is unreasonably high.

3.8 Conclusion

Intercloud seeks to facilitate resource sharing among clouds. To support Intercloud, a trust evaluation framework among clouds and users is required. For trust evaluation, conventional protocols are typically based on a centralized architecture focusing on a one-way relationship. For Intercloud, the environment is highly dynamic and distributed, and relationships can be one-way or two-way (i.e., clouds provide services to each other). This chapter presents a distributed trust evaluation protocol with privacy protection for Intercloud. The new contributions and innovative features are summarized below. First, feedback is protected by homomorphic encryption with verifiable secret sharing. Second, to cater to the dynamic nature of Intercloud, trust evaluation can be conducted in a distributed manner and is functional even when some of the parties are offline. Third, to facilitate customized trust evaluation, an innovative mechanism is used to store feedback, such that it can be processed flexibly while protecting feedback privacy. The protocol has been proved based on a formal security model. Simulations have been performed to demonstrate the effectiveness of the protocol. The results show that even when half of the clouds are malicious or offline, by choosing suitable operational parameters the protocol can still support effective trust evaluation with privacy protection.

4

Order-Hiding Range Query over Encrypted Data without Search Pattern Leakage

4.1 Introduction

For cloud data storage, data privacy and security are two key concerns. Although sensitive data can be encrypted before they are stored in the cloud, the encrypted data can hardly be processed efficiently. Hence, a lightweight solution is required to satisfy both high security and high efficiency requirements. In this chapter, we study the problem of range query over encrypted data which are required for some applications. For instance, social networks and computer networks can be modelled as large graphs. To protect the privacy of network connections in a graph, information on the vertices and edges is encrypted before uploading to the cloud server. To find the shortest path between two vertices (e.g., using Dijkstra's algorithm), it needs to compare the new path distance to the current vertex with the previous one. Hence, the cloud server needs to perform comparisons on the encrypted path information [110, 83]. In other scenarios, such as medical record reviewing or financial auditing, records are sensitive and queries are usually based on range values (e.g., certain time periods or a particular range of IP addresses) [111]. They also require to perform secure range queries over encrypted data. Compared with keyword queries, there are more technical challenges in designing an effective and secure scheme for range queries on encrypted data.

In general, there are two types of solutions to support secure range queries on encrypted data. The first type (e.g., [73, 74, 75, 76, 77, 78]) aims at providing faster search time while disclosing certain information (e.g., the ordering between unmatched records and trapdoors). The second type (e.g., [79, 80, 112]) is designed to achieve higher security at the expense of extra cost (e.g., longer search time, large index storage space, false positives in the query results). For instance, Fully Homomorphic Encryption (FHE), which allows arithmetic operations on ciphertexts can be applied to support privacy-preserving range queries [112]. Although FHE can achieve high security, its computational cost is high. In this chapter, we design and evaluate a privacy-preserving range query scheme to address the above-mentioned limitations. Our main contributions are summarized as follows:

- We propose an index generation algorithm that is secure against "inference attacks". Since the index generated in our scheme is not deterministic, the cloud server cannot obtain any distribution or relationship information (e.g., frequency and order) of records based on their indexes.

- We provide a range comparison method that does not disclose the different binary bit(s) between a record and a query. For unmatched records, the cloud server cannot learn their orderings from their comparison results with query range $[a, b]$ (i.e., if they are smaller than the lower bound "$\leq a$" or larger than the upper bound "$\geq b$", respectively).

- We develop a trapdoor generation algorithm that can hide query search patterns. Due to the non-deterministic nature of our trapdoor generation algorithm, the cloud server cannot determine 1) whether any two trapdoors are created from the same query, 2) whether the upper/lower bound of one query is larger/smaller than that of another query, and 3) for each query, the cloud server cannot distinguish its upper bound from its lower bound, based on the trapdoor value and query results.

- We prove that our scheme is secure under the best security model available, without the restriction of identical search patterns. This is better than existing schemes. Furthermore, our scheme does not produce false positives in the query results.

- We implement and compare our scheme with OPE [74] and ORE [78] schemes. Our scheme is on average over 16 times faster than the OPE scheme in index generation. For each range comparison, our scheme is on average 3.89 times faster than the ORE scheme when the processing unit is 2 bytes. This indicates that our scheme can achieve higher security than the ORE scheme, while maintaining good efficiency.

The rest of this chapter is organized as follows. Section 4.2 provides the scheme's general construction and security goal. Section 4.3 describes the building block utilised in the scheme. Section 4.4 presents the details of our proposed privacy-preserving range query scheme. Experimental setup and results are illustrated in Section 4.5. Section 4.6 analyses the security of the scheme and proves that it can achieve the defined security goals. Section 4.7 presents the conclusion.

4.2 General Construction and Security Definitions

We present the system model, algorithm construction, and security definitions of our scheme in this section.

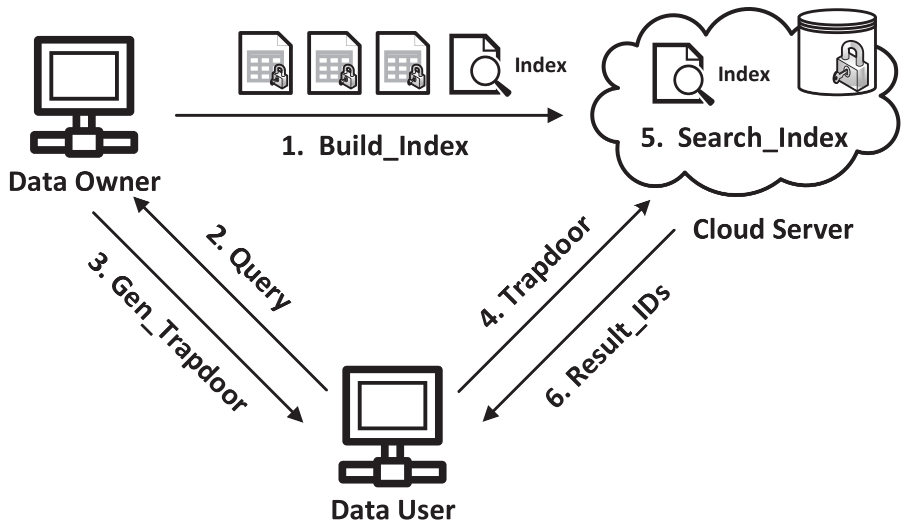


Fig. 4.1: Architecture of range query on encrypted data in cloud computing

4.2.1 System Model

The system model discussed in our scheme is shown in Fig. 4.1. There are three parties: a data owner, a data user, and a cloud server. The data owner would like to store a collection of sensitive records to the cloud. However, it does not fully trust the cloud provider. Thus, before uploading records to the cloud, it first encrypts each record. To provide the cloud with the ability to perform relational operations on encrypted records without decryption, the data owner associates each encrypted record with a secure searchable index generated using the attribute value of records to be queried. A valid data user submits a trapdoor - which is obtained from the data owner generated from plaintext query - to the server on the cloud. After obtaining the trapdoor, the server searches the matching records remotely via indexes, and returns the ID of satisfied records as the query results to the data user.

In our model, the data owner and authorized data user are regarded as fully trusted. They communicate through a secure channel. In practice, clouds are managed by well-established IT companies. In cases of attack, it is more likely for cloud providers to conduct passive attacks instead of active attacks on specific users. Therefore, in our scheme, we assume the cloud server as a semi-honest (honest-but-curious) adversary, which is trusted to correctly execute required communication protocols and algorithms. At the same time, the cloud server actively deduces the sensitive information of records and the content of received queries. The semi-honest adversary model has commonly been adopted in the existing privacy-preserving range query and keyword searchable symmetric encryption schemes [74, 79, 80, 87, 89, 90]. We make the same assumption as before.

4.2.2 Notation and Definition

Notations and functions in the rest of this chapter are defined as follows.

- $\mathcal{R} = (r_1, ..., r_N)$ is a collection of records, where r_u is the ID of the uth record.

- $\mathcal{D} = (d_1, ..., d_N)$ is a collection of attribute values from $\mathcal{R}$. Each attribute value $d_u (1 \leq u \leq N)$ contained in a record r_u, where $d_u \in \mathbb{Z}_{2^\ell}$ and ℓ is the bit length of attribute values.

- $\mathcal{I} = (I_{d_1}, ..., I_{d_N})$ is a collection of encrypted searchable indexes based on $\mathcal{D}$, where I_{d_u} is the index of d_u.

- $|I_{d_u}|$ is the index size, which is the number of row vectors and polynomial random nonce pairs used in generating I_{d_u}.

- $Q = [w_L, w_H]$ is a range query, where $w_L, w_H \in \mathbb{Z}_{2^\ell}$, w_L is the lower bound, and w_H is the upper bound of query Q, respectively.

- T_Q is the secure trapdoor of the query Q.

- $|T_Q|$ is the trapdoor size, which is the number of column vectors in T_Q.

- $\mathcal{D}(Q)$ is the set of query Q's result on the collection $\mathcal{D}$, where the attribute values of records in $\mathcal{D}(Q)$ belongs to the interval $[w_L, w_H]$.

Before going into the details, we first define the following main algorithms of our scheme.

Definition 2 *(*Privacy Preserving Range Query Scheme*): A searchable symmetric encryption range query scheme consists of the following four algorithms.*

- $(\mathsf{sk}, \mathsf{params}) \leftarrow \textbf{Gen\_Key}(\lambda)$: is executed by the data owner. Taking as input security parameter λ, the algorithm generates secret key sk and system parameter params.

- $\mathcal{I} \leftarrow \textbf{Bld\_Index}(\mathsf{sk}, \mathcal{D})$: is run by the data owner to support cloud server with the capability of searching on $\mathcal{R}$. It takes as input secret key sk and the attribute value collection $\mathcal{D}$ and outputs encrypted searchable indexes $\mathcal{I}$.

- $T_Q \leftarrow \textbf{Gen\_Trapdoor}(\mathsf{sk}, Q)$: is run by the data owner to create a trapdoor for a given query Q. It takes both secret key sk and range query Q: $[w_L, w_H]$ as the input. It outputs trapdoor T_Q for query Q.

- $\mathcal{D}(Q) \leftarrow \textbf{Rag\_Search}(\mathcal{I}, T_Q)$: is run by the cloud server to determine the query result. It takes as input encrypted searchable indexes $\mathcal{I}$ and trapdoor T_Q. It outputs record set $\mathcal{D}(Q)$ as the query result.

Correctness. We consider a privacy-preserving range query scheme over a collection of records is correct, if for all records in $\mathcal{D}(Q)$,

$$\Pr[r_u \in \mathsf{Rag\_Search}(\mathcal{I}, T_Q) | d_u \in [w_L, w_H]] = 1 - \mathsf{negl}(\lambda).$$

4.2.3 Security Goals

Since the records are encrypted by the data owner before outsourcing to the cloud server, our scheme aims to preserve the privacy of the searchable index and trapdoor during and after queries from the following aspects:

1) The attribute value used to generate the index is part of the record contents . As a result, the cloud server should not learn attribute value d_u from its index I_{d_u} or from trapdoor T_Q of any issued query.

2) The cloud server should not deduce whether the indexes of two different records are built from the same attribute value (hide the frequency), and whether the attribute value of one record is larger or smaller than that of another record (hide the order).

3) The privacy of a trapdoor is inherently linked to the privacy of the index. Hence, the cloud server should not distinguish the value of w_L and w_H from its trapdoor T_Q and from the received record indexes.

4) The cloud server is unable to determine whether two trapdoors are created from the same query range or not (hide the search pattern), and whether the upper or lower bound of one query is larger or smaller than that of another query.

5) For each unmatched record $r_u \notin \mathcal{D}(Q)$, the cloud server should not know if $d_u > w_H$ or $d_u < w_L$ and in which bits of d_u and in which bits of w_H/w_L that differs d_u from query Q.

Since we are unable to enumerate all possible attacks, we introduce the following security definitions.

4.2.4 Security Model

Let $\mathsf{SSE}^{\mathsf{RAG}} = (\mathsf{Gen\_Key}, \mathsf{Bld\_Index}, \mathsf{Gen\_Trapdoor}, \mathsf{Rag\_Search})$ be our privacy-preserving range query scheme. We analyse the security of $\mathsf{SSE}^{\mathsf{RAG}}$ under the game-based IND-CKA2 security model [90] with appropriate modifications. Since trapdoors are deterministically generated in Curtmola et al.'s scheme, adversaries in the game of IND-CKA2 model can only ask for the trapdoors of queries in pairs. And the IND-CKA2 model includes the search pattern of queries as one of its leakages. However, in our scheme $\mathsf{SSE}^{\mathsf{RAG}}$, the trapdoors are not deterministically created. The adversary in the game of our security model can make a request for the trapdoor of the single query.

To make the security definition of IND-CKA2 fit our stronger security guarantee, we relax the assumption of the same search pattern in IND-CKA2, and redefine two games in Section 4.2.4 and Section 4.2.4 to prove the security of indexes/ciphertext and the security of trapdoors separately. In each of the two games, challenger $\mathcal{C}$ executes the actual algorithms in $\mathsf{SSE}^{\mathsf{RAG}}$. An adversary $\mathcal{A}$ adaptively sends queries to challenger $\mathcal{C}$ based on all previously obtained indexes, trapdoors, and search results. Before we present our security model, we first formulate the information leakages that arise from our scheme.

Definition 3 *(**Information Leakage**) Function $\mathcal{L}(\mathcal{D}, |T_Q|, Q)$ describes the leakage from a range query Q over a collection of attribute values $\mathcal{D}$, that is*

$$\mathcal{L}(\mathcal{D}, Q) = \langle |I_{d_1}|, ..., |I_{d_N}|, |T_Q|, \mathcal{D}(Q) \rangle.$$

comprised of the index size of attribute values in $\mathcal{D}$, trapdoor size $|T_Q|$, and the access pattern $\mathcal{D}(Q)$ (i.e., the record set of range query Q's result).

Ciphertext Indistinguishability Security

We use the following game to formally define the requirement that the adversary learns nothing about the attribute values beyond their index sizes and access patterns.

Game 1

- **Setup**: Challenger $\mathcal{C}$ creates a large collection of attribute values $\mathbb{D}$, a sequence of range queries $\mathbb{Q}$ and gives them to the adversary $\mathcal{A}$. $\mathcal{C}$ runs $\mathsf{Gen\_Key}(\lambda)$ to generate secret key sk and system parameter params. $\mathcal{C}$ keeps secret key sk and sends params to $\mathcal{A}$.

- **Phase 1**: Adversary adaptively issues q_1 pairs of requests based on past received indexes and trapdoors, where the request $1 \leq i \leq q_1$ is shown as follows:

  - Index generation request for an attribute value collection $\mathcal{D}_i \in \mathbb{D}$: The challenger runs $\mathsf{Bld\_Index}(\mathsf{sk}, \mathcal{D}_i)$ on a collection of attribute values $\mathcal{D}_i$, and forwards the index $\mathcal{I}_i$ to the adversary.

  - Trapdoor generation request for a range query $Q_i \in \mathbb{Q}$: The challenger runs $\mathsf{Gen\_Trapdoor}(\mathsf{sk}, Q_i)$ on a range query Q_i, and forwards the index T_{Q_i} to the adversary.

- **Challenge**: The adversary submits two plaintext collections of attribute values $\boldsymbol{\mathcal{D}_0}$ and $\boldsymbol{\mathcal{D}_1} \in \mathbb{D}$. The challenger randomly flips a bit $b \in \{0,1\}$ and responds to the adversary with the index $\mathcal{I}_b \leftarrow \mathsf{Bld\_Index}(\mathsf{sk}, \mathcal{D}_b)$ as its challenge ciphertext.

- **Phase 2**: For request $q_1 + 1 \leq i \leq q$, adversary repeats the same process as in Phase 1 and finally obtains $\boldsymbol{\langle \mathcal{I}_1, ..., \mathcal{I}_q, \mathcal{I}_b \rangle}$ and $\boldsymbol{\langle T_{Q_1}, ..., T_{Q_q} \rangle}$.

- **Guess**: With the restriction that $\boldsymbol{\mathcal{D}_0}$ and $\boldsymbol{\mathcal{D}_1}$ cause the same leakage under all chosen queries

$$\mathcal{L}(\boldsymbol{\mathcal{D}_0}, Q_1) = \mathcal{L}(\boldsymbol{\mathcal{D}_1}, Q_1),$$
$$\ldots\ldots$$
$$\mathcal{L}(\boldsymbol{\mathcal{D}_0}, Q_q) = \mathcal{L}(\boldsymbol{\mathcal{D}_1}, Q_q),$$

  adversary guesses a bit $b^{'} \in \{0,1\}$. If $b = b^{'}$, we consider that the adversary wins the index security game.

Definition 4 *(**Ciphertext Indistinguishability Security**): We say that* $\mathsf{SSE}^{\mathsf{RAG}}$ *is ciphertext/index secure in terms of adaptive indistinguishability if for all polynomial-sized adversary $\mathcal{A}$, the advantage in winning Game 1 is less than a negligible function of λ.*

$$Pr[b^{'} = b] - \frac{1}{2} \leq \mathsf{negl}(\lambda).$$

Trapdoor Indistinguishability Security

We use the following game to formally define the requirement that the adversary learns nothing about the range query values beyond their trapdoor sizes and access patterns.

Game 2

- **Setup**: Challenger $\mathcal{C}$ creates a large collection of attribute values $\mathbb{D}$, a sequence of range queries $\mathbb{Q}$ and gives them to the adversary $\mathcal{A}$. $\mathcal{C}$ runs $\mathsf{Gen\_Key}(\lambda)$ to generate secret key sk and system parameter params. $\mathcal{C}$ keeps secret key sk and sends params to $\mathcal{A}$.

- **Phase 1**: Adversary adaptively issues q_1 pairs of requests based on past received indexes and trapdoors, where the request $1 \leq i \leq q_1$ is shown as follows:

  – Index generation request for an attribute value collection $\mathcal{D}_i \in \mathbb{D}$: The challenger runs $\mathsf{Bld\_Index}(\mathsf{sk}, \mathcal{D}_i)$ on a collection of attribute values $\mathcal{D}_i$, and forwards the index $\mathcal{I}_i$ to the adversary.

  – Trapdoor generation request for a range query $Q_i \in \mathbb{Q}$: The challenger runs $\mathsf{Gen\_Trapdoor}(\mathsf{sk}, Q_i)$ on a range query Q_i, and forwards the index T_{Q_i} to the adversary.

- **Challenge**: The adversary submits two range queries $\boldsymbol{Q_0}$ and $\boldsymbol{Q_1} \in \mathbb{Q}$. The challenger randomly flips another bit $c \in \{0, 1\}$ and replies to the adversary with the trapdoor $T_{Q_c} \leftarrow \mathsf{Gen\_Trapdoor}(\mathsf{sk}, Q_c)$ as its challenge trapdoor.

- **Phase 2**: For request $q_1+1 \leq i \leq q$, adversary repeats the same process as in Phase 1 and finally obtains $\langle \boldsymbol{\mathcal{I}_1, ..., \mathcal{I}_q} \rangle$ and $\langle \boldsymbol{T_{Q_1}, ..., T_{Q_q}, T_{Q_c}} \rangle$.

- **Guess**: With the restriction that $\boldsymbol{Q_0}$ and $\boldsymbol{Q_1}$ cause the same leakage under all chosen attribute value collections

$$\mathcal{L}(\mathcal{D}_1, \boldsymbol{Q_0}) = \mathcal{L}(\mathcal{D}_1, \boldsymbol{Q_1}),$$
$$......$$
$$\mathcal{L}(\mathcal{D}_q, \boldsymbol{Q_0}) = \mathcal{L}(\mathcal{D}_q, \boldsymbol{Q_1}),$$

  adversary guesses a bit $c' \in \{0,1\}$. If $c = c'$, we consider that the adversary wins the trapdoor security game.

Definition 5 *(**Trapdoor Indistinguishability Security**): We say that* $\mathsf{SSE}^{\mathsf{RAG}}$ *is trapdoor secure in terms of adaptive indistinguishability if for all polynomial-sized adversary $\mathcal{A}$, the advantage in winning Game 2 is less than a negligible function of λ.*

$$Pr[c' = c] - \frac{1}{2} \leq \mathsf{negl}(\lambda).$$

Definition 6 *(**Indistinguishability Security**): We say that scheme* $\mathsf{SSE}^{\mathsf{RAG}}$ *is both ciphertext and trapdoor secure in terms of adaptive indistinguishability if for all polynomial-sized adversary $\mathcal{A}$, the advantage in winning both Game 1 and Game 2 is less than a negligible function of λ.*

4.3 Building Block

The building block of our scheme is the 0/1 encoding first proposed by Lin and Tzeng to address the Millionaires' Problem [113]. The basic idea of 0/1 encoding is to turn data comparison to the problem of finding the intersection of two sets. For a comparison $a > b$, it needs to find a common element between the 1-encoding set of a and 0-encoding set of b. Let $s = s_1 s_2 ... s_\ell$ denote the binary string of s and $s_1 s_2 ... s_h$ represent the binary string of the first h digits of s (i.e., the h-length prefix string of s). The prefix string set P_s of s is defined as follows

$$P_s = \{s_1 s_2 ... s_h | 1 \leq h \leq \ell\}. \tag{4.1}$$

For each prefix string $s_1 s_2 ... s_h$ ending up with $s_h = 0$, we write the binary string $s_1 s_2 ... s_{h-1} 1$ as one of the elements in the 0-encoding set S_s^0. For each prefix string $s_1 s_2 ... s_h$ ending up with $s_h = 1$, we directly write its prefix string $s_1 s_2 ... s_h$ as one of the elements in the 1-encoding set S_s^1. Two binary string sets S_s^0 and S_s^1 are defined as the 0-encoding and 1-encoding sets of s, such that

$$S_s^0 = \{s_1 s_2 ... s_{h-1} 1 | s_h = 0, 1 \leq h \leq \ell\}, \tag{4.2}$$
$$S_s^1 = \{s_1 s_2 ... s_h | s_h = 1, 1 \leq h \leq \ell\}, \tag{4.3}$$

where ℓ is the number of binary digits in the binary string of s. For any two numbers a and b with bit-length of ℓ, their 0-encoding set and 1-encoding sets have the following properties [113].

$$\begin{cases} a > b \iff S_a^1 \cap S_b^0 \neq \emptyset, \\ a \leq b \iff S_a^1 \cap S_b^0 = \emptyset. \end{cases} \tag{4.4}$$

When $S_a^1 \cap S_b^0 \neq \emptyset$, their common element is denoted as

$$a_1 a_2 ... a_{h-1} 1 |_{a_h = 1} = b_1 b_2 ... b_{h-1} 1 |_{b_h = 0}$$

which is equivalent to

$$a_1 a_2 ... a_{h-1} 0 |_{a_h = 1} = b_1 b_2 ... b_{h-1} 0 |_{b_h = 0}.$$

Then, the right side element $b_1b_2...b_{h-1}0|_{b_h=0}$ belongs to P_b the prefix string set of number b. Based on the left side element $a_1a_2...a_{h-1}0|_{a_h=1}$, we define the following new 1-encoding set $\widetilde{S_s^1}$

$$\widetilde{S_s^1} = \{s_1s_2...s_{h-1}0|s_h = 1, 1 \leq h \leq \ell\}. \tag{4.5}$$

Since all elements of $\widetilde{S_s^1}$ end up with $a_h = 0$, we can convert the determination of $S_a^1 \cap S_b^0$ to the comparison between $\widetilde{S_a^1}$ and P_b, that is $S_a^1 \cap S_b^0 = \widetilde{S_a^1} \cap P_b$. Conversely, we can compare P_a with the 0-encoding set of b. Since each element of S_a^1 belongs to P_a and all elements of S_b^0 end up with $b_h = 1$, we can get $S_a^1 \cap S_b^0 = P_a \cap S_b^0$.

Therefore, $\widetilde{S_s^1}, S_s^0$ and P_s satisfy the same properties as the $0/1$ encoding when comparing two numbers a and b.

$$\begin{cases} a > b \iff \widetilde{S_a^1} \cap P_b \neq \emptyset \text{ or } P_a \cap S_b^0 \neq \emptyset, \\ a \leq b \iff \widetilde{S_a^1} \cap P_b = \emptyset \text{ or } P_a \cap S_b^0 = \emptyset. \end{cases} \tag{4.6}$$

Here is an example of how to compare two numerical values. Let $a = 9 = (1001)_2$ and $b = 14 = (1110)_2$ denote two binary strings with 4 bits. Based on definitions in Equation (4.1), (4.2) and (4.5), we obtain

$$P_9 = \{1, \underline{10}, 100, 1001\}, S_9^0 = \{\underline{\underline{11}}, 101\}, \widetilde{S_9^1} = \{0, 1000\},$$

$$P_{14} = \{1, \underline{\underline{11}}, 111, 1110\}, S_{14}^0 = \{1111\}, \widetilde{S_{14}^1} = \{0, \underline{10}, 110\}.$$

Since $\widetilde{S_{14}^1} \cap P_9 = \{10\} \neq \emptyset$ and $P_{14} \cap S_9^0 = \{11\} \neq \emptyset$, we learn that $14 > 9$. $\widetilde{S_9^1} \cap P_{14} = \emptyset$ and $P_9 \cap S_{14}^0 = \emptyset$, we learn that $9 \leq 14$.

Based on Equation (4.6), we obtain the following properties when comparing a value d with a closed interval $[w_L, w_H]$.

$$\begin{cases} d \notin [w_L, w_H] \\ \qquad\qquad \iff d < w_L \text{ and } d > w_H \\ \qquad\qquad \iff P_d \cap \widetilde{S^1_{w_L}} \neq \emptyset \text{ and } P_d \cap S^0_{w_H} \neq \emptyset. \end{cases} \tag{4.7}$$

This means that we can use the same prefix string set of d to compare with the different encoding sets of upper bound w_H and lower bound w_L of a range query.

As shown in the property of the 0/1 encoding scheme, if $a \leq b$ then $S^1_a \cap S^0_b = \emptyset$. Based on this property, we can easily obtain that when $a = b(a \leq b, b \leq a)$, then $S^1_a \cap S^0_b = \emptyset$ and $S^1_b \cap S^0_a = \emptyset$ at the same time. Accordingly, using our newly designed scheme, we can show that when a is equal to b, then $P_a \cap S^0_b = \emptyset$ and $P_a \cap \widetilde{S^1_b}$. Therefore, when users perform the keyword search over encrypted data, our proposed scheme can still support the keyword comparison by comparing the prefix string set of one value P_a with both the encoding set of another value (i.e., S^0_b and $\widetilde{S^1_b}$). Hence, the following designs in our scheme still can still prevent the search pattern leakage in the keyword search.

4.4 Our Privacy Preserving Range Query Scheme

We present our privacy-preserving range query scheme in this section. We assume that the database records have already been encrypted before being stored in the cloud server.

4.4.1 Scheme Overview

As introduced in the previous sections, the cloud server will match the index of records with the query trapdoor to target the satisfied database records. To provide a security guarantee, our scheme needs to achieve indistinguishability of both the index and trapdoor. The building block of our scheme is $0/1$ encoding introduced in Equation (4.4) and (4.6), which converts the data comparison into the calculation of the intersection between their corresponding $0/1$ encoding set $\widetilde{S_a^1}/S_a^0$ and prefix string set P_b.

Inspired by the idea of private set intersection (PSI) [114], our scheme represents the encoding elements in P_b as the roots of polynomial function. During the comparison phase, the encoding elements in $\widetilde{S_a^1}$ and S_a^0 are plugged into the polynomial function. This design can prevent the server from knowing which binary bit the two values differ. To hide the search pattern of different queries, we use an invertible matrix with random numbers, such that the cloud server is unable to distinguish between different queries. As the encoding elements are placed in shuffled order during the generation of the trapdoor, the cloud server cannot locate the intersection results from the upper or lower bound of the queries. The details of our scheme are indicated below.

4.4.2 Scheme Details

- **Gen_Key**(λ) : First, the Gen_Key algorithm is performed by the data owner to determine a modulus p and a secure keyed hash function $H : \{0,1\}^\lambda \times \{0,1\}^\ell \to \{0,1\}^{s(\lambda)}$, in which $s(\lambda)$ is a quantity polynomial of the security parameter λ. Then, Gen_Key generates a secret key K and a random invertible matrix $M_{(\ell+3)\times(\ell+3)}$, in which ℓ is the bit length of the attribute value to be indexed. The system parameter params is the pair (p, H) sent to the cloud server and the secret key sk is the pair (K, M) kept by the data owner.

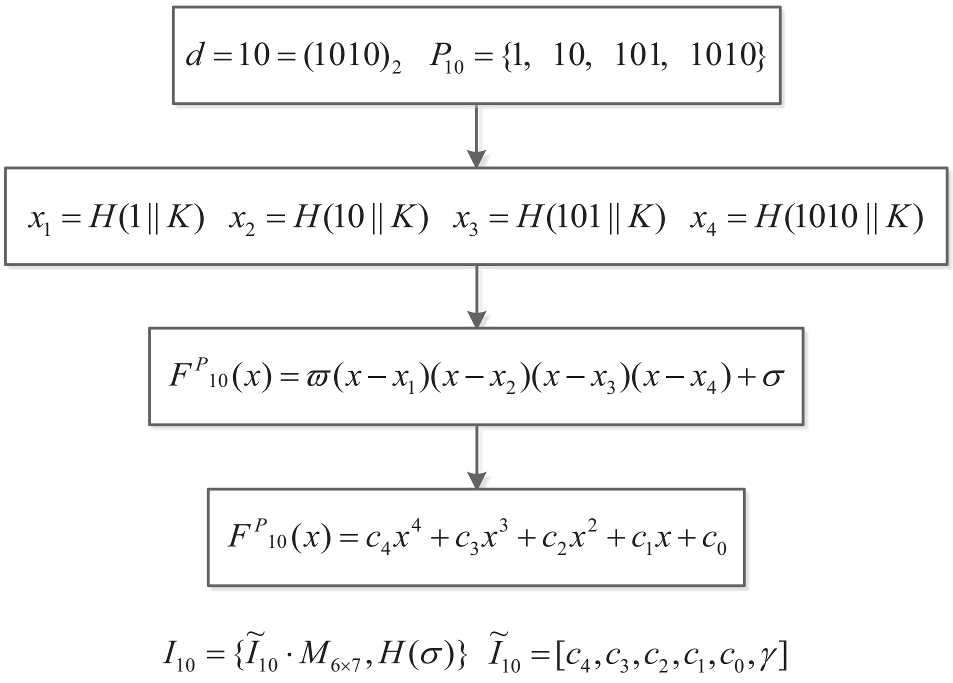

$d = 10 = (1010)_2 \quad P_{10} = \{1, \ 10, \ 101, \ 1010\}$
$x_1 = H(1 || K) \quad x_2 = H(10 || K) \quad x_3 = H(101 || K) \quad x_4 = H(1010 || K)$
$F^P{}_{10}(x) = \varpi(x - x_1)(x - x_2)(x - x_3)(x - x_4) + \sigma$
$F^P{}_{10}(x) = c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0$
$I_{10} = \{\widetilde{I}_{10} \cdot M_{6\times7}, H(\sigma)\} \quad \widetilde{I}_{10} = [c_4, c_3, c_2, c_1, c_0, \gamma]$

Fig. 4.2: Example of index generation.

- **Bld_Index**(sk, $\mathcal{D}$) : The Bld_Index algorithm is performed by the data owner to generate indexes for records. For each attribute value d_u in the collection of $\mathcal{D}$, Bld_Index algorithm executes the following steps. Fig. 4.2 illustrates an example of index generation.

1) First, the Bld_Index algorithm computes the ℓ length binary strings for d_u and calculates the prefix string set $P_{d_u} = \{e_1, e_2, ..., e_\ell\}$ of d_u. Then, for each element $e_i \in P_{d_u}(1 \leq i \leq \ell)$, it computes the corresponding hash value $x_i = H(e_i||K) \pmod p$ by the secret key K generated in Gen_Key.

2) To hide in which bit of d_u that differs d_u from a range query, the algorithm constructs the following polynomial function $F^P_{d_u}(x)$ for all hash values $\{x_1, ..., x_\ell\}$. A pair of polynomial random nonces ϖ_u and σ_u is embedded

in $F^P_{d_u}(x)$ in order to distinguish the indexes of different records with the same attribute value,

$$\begin{aligned} F^P_{d_u}(x) &= \varpi_u(x-x_1)(x-x_2)...(x-x_\ell)+\sigma_u \ (\mathrm{mod}\ p) \\ &= c_\ell x^\ell + c_{\ell-1}x^{\ell-1}...c_1+c_0, \end{aligned} \tag{4.8}$$

where $\{x_1,...,x_\ell\}$ are the hash values of elements in P_{d_u}.

3) First, the algorithm constructs a row vector $\tilde{I}_{d_u}$ with $\ell+2$ elements, where $c_\ell,...,c_0$ are the coefficients of $F^P_{d_u}(x)$, and γ_u is a random nonce:

$$\tilde{I}_{d_u} = [c_\ell, c_{\ell-1}..., c_1, c_0, \gamma_u]. \tag{4.9}$$

Then, the algorithm constructs two matrixes $[M]_{(\ell+2)\times(\ell+3)}$ and $[M^{-1}]_{(\ell+3)\times(\ell+2)}$ based on the matrix M in sk. $[M]_{(\ell+2)\times(\ell+3)}$ is constructed by removing the $(\ell+2)$th row of M and $[M^{-1}]_{(\ell+3)\times(\ell+2)}$ is obtained by deleting the $(\ell+3)$th column of M^{-1}.

4) $\tilde{I}_{d_u}$ is right multiplied by matrix $[M]_{(\ell+2)\times(\ell+3)}$ to obtain

$$\begin{aligned} I'_{d_u} &= \tilde{I}_{d_u}\cdot[M]_{(\ell+2)\times(\ell+3)} \\ &= [c_\ell,...,c_1,c_0,0,\gamma_u]\cdot[M]_{(\ell+3)\times(\ell+3)}. \end{aligned} \tag{4.10}$$

The Bld_Index algorithm outputs the encrypted searchable index of d_u as a 2-tuple of

$$I_{d_u} = \{I'_{d_u}, H(\sigma_u)\} \tag{4.11}$$

where $H(\sigma_u)$ is the hash value of σ_u. Finally, the data owner submits encrypted searchable index $\mathcal{I}=(I_{d_1},...,I_{d_N})$ to the cloud sever.

- **Gen_Trapdoor**(sk, Q) : The Gen_Trapdoor algorithm is performed by the data owner to generate the trapdoor for a query $Q : [w_L, w_H]$ requested by the data user. Fig. 4.3 illustrates an example of trapdoor generation.

  1) The algorithm first computes two ℓ length binary strings for w_L and w_H. Then, it calculates new 1-encoding set $\widetilde{S^1_{w_L}}$ for lower bound w_L based on Equation (4.5) and the 0-encoding set $S^0_{w_H}$ for upper bound w_H based on Equation (4.2).

  2) We assume that the total number of elements in $S^0_{w_H}$ and $\widetilde{S^1_{w_L}}$ is g. Then, for each element $e_j (1 \leq j \leq g)$ in $S^0_{w_H}$ and $\widetilde{S^1_{w_L}}$, the algorithm computes its hash value by the secret key K to obtain $y_j = H(e_j||K) \pmod{p}$. Note that $y_1, ..., y_g$ are placed in shuffled order, y_j does not correspond to the jth encoding element in $\widetilde{S^1_{w_L}}$ or $S^0_{w_H}$.

  3) Since a hash function has the property of one-wayness and collision resistance, the set $H(P_{d_u}||K) = \{x_1, ..., x_\ell\}$ and set $H(\widetilde{S^1_{w_L}}||K) \cup H(S^0_{w_H}||K) = \{y_1, ..., y_g\}$ still satisfy the same properties in Equation (4.6), shown as follows:

$$\begin{cases} a > b \iff H(\widetilde{S^1_a}||K) \cap H(P_b||K) \neq \emptyset \\ \qquad\qquad \text{or } H(P_a||K) \cap H(S^0_b||K) \neq \emptyset, \\ a \leq b \iff H(\widetilde{S^1_a}||K) \cap H(P_b||K) = \emptyset \\ \qquad\qquad \text{or } H(P_a||K) \cap H(S^0_b||K) = \emptyset. \end{cases} \tag{4.12}$$

  The union of intersection

$$[H(\widetilde{S^1_{w_L}}||K) \cap H(P_{d_u}||K)] \cup [H(P_{d_u}||K) \cap H(S^0_{w_H}||K)]$$

  is the intersection $\{x_1, ..., x_\ell\} \cap \{y_1, ..., y_g\}$.

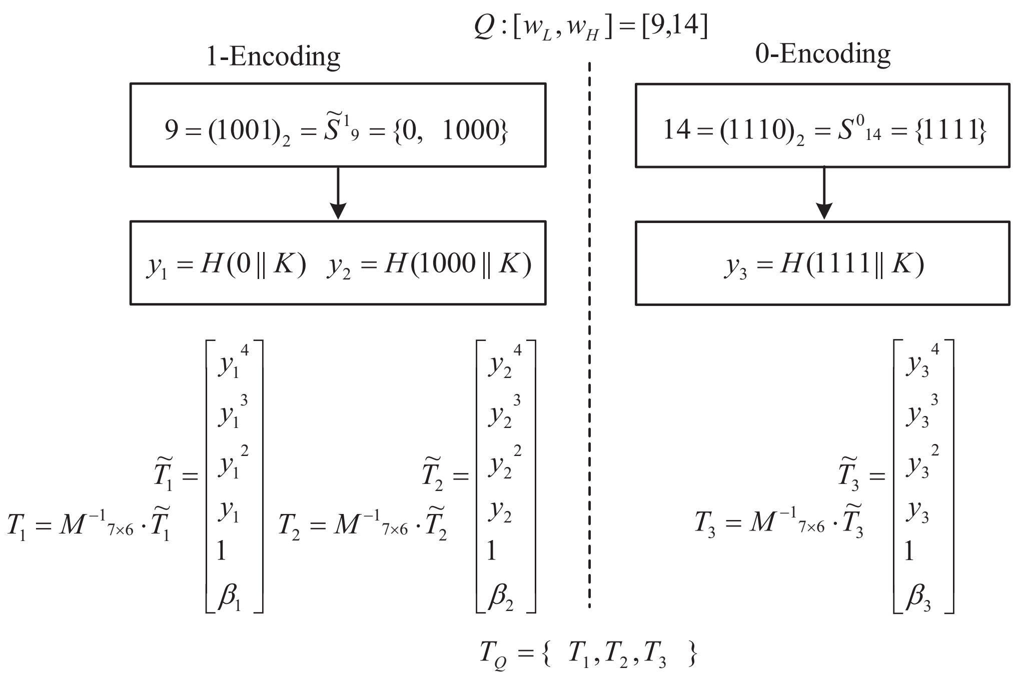

$Q:[w_L, w_H]=[9,14]$
1-Encoding
0-Encoding
$9=(1001)_2=\widetilde{S}^1{}_9=\{0,\ 1000\}$
$14=(1110)_2=S^0{}_{14}=\{1111\}$
$y_1=H(0\|K)\quad y_2=H(1000\|K)$
$y_3=H(1111\|K)$
$\widetilde{T}_1=[y_1^4, y_1^3, y_1^2, y_1, 1, \beta_1]^\top$
$\widetilde{T}_2=[y_2^4, y_2^3, y_2^2, y_2, 1, \beta_2]^\top$
$\widetilde{T}_3=[y_3^4, y_3^3, y_3^2, y_3, 1, \beta_3]^\top$
$T_1=M^{-1}{}_{7\times6}\cdot\widetilde{T}_1$
$T_2=M^{-1}{}_{7\times6}\cdot\widetilde{T}_2$
$T_3=M^{-1}{}_{7\times6}\cdot\widetilde{T}_3$
$T_Q=\{\ T_1, T_2, T_3\ \}$

Fig. 4.3: Example of trapdoor generation.

4) To distinguish the trapdoors of different queries with the same lower and upper bound values, the algorithm generates a random nonce β_j for each y_j and constructs a column vector $\tilde{T}_j$ with $\ell+2$ elements, as follows

$$\tilde{T}_j = [y_j^\ell, y_j^{\ell-1}, ..., y_j, 1, \beta_j]^\top \pmod p. \tag{4.13}$$

Then, $\tilde{T}_j$ is left multiplied by matrix $[M^{-1}]_{(\ell+3)\times(\ell+2)}$ to obtain the jth trapdoor element, as follows

$$\begin{aligned} T_j &= [M^{-1}]_{(\ell+3)\times(\ell+2)} \cdot \tilde{T}_j \\ &= [M^{-1}]_{(\ell+3)\times(\ell+3)} \cdot [y_j^\ell, ..., y_j, 1, \beta_j, 0]^\top. \end{aligned} \tag{4.14}$$

Finally, the data owner returns g vectors as the trapdoor of query Q to the data user

$$T_Q = \{T_1, ..., T_g\}. \tag{4.15}$$

$$\boxed{\begin{array}{l} d=10 \\ \text{Index } I_{10}=\{\widetilde{I}_{10}\cdot M_{6\times 7}, H(\sigma)\} \end{array}} \quad \boxed{\begin{array}{l} Q:[w_L,w_H]=[9,14] \\ \text{Trapdoor } \; T_Q=\{T_1,T_2,T_3\} \end{array}}$$

$$\begin{aligned} \sigma_1 &= I'_{10}\cdot T_1=\widetilde{I}_{10}\cdot M_{6\times 7}\cdot M^{-1}{}_{7\times 6}\cdot \widetilde{T}_1=[c_4,c_3,c_2,c_1,c_0,\underline{\underline{0}},\gamma]\cdot M\cdot M^{-1}\cdot \begin{bmatrix} y_1{}^4 \\ y_1{}^3 \\ y_1{}^2 \\ y_1 \\ 1 \\ \beta_1 \\ \underline{\underline{0}} \end{bmatrix} \\ &= c_4 y_1{}^4+c_3 y_1{}^3+c_2 y_1{}^2+c_1 y_1+c_0 \\ &= \varpi(y_1-x_1)(y_1-x_2)(y_1-x_3)(y_1-x_4)+\sigma \end{aligned}$$

$$\begin{aligned} \sigma_2 &= I'_{10}\cdot T_2=\widetilde{I}_{10}\cdot M_{6\times 7}\cdot M^{-1}{}_{7\times 6}\cdot \widetilde{T}_2 \\ &= c_4 y_2{}^4+c_3 y_2{}^3+c_2 y_2{}^2+c_1 y_2+c_0 \\ &= \varpi(y_2-x_1)(y_2-x_2)(y_2-x_3)(y_2-x_4)+\sigma \end{aligned}$$

$$\underline{H(\sigma_1)\neq H(\sigma), H(\sigma_2)\neq H(\sigma)\rightarrow \widetilde{S}^1{}_9\bigcap P_{10}=\phi\rightarrow 9\leq 10}$$

$$\begin{aligned} \sigma_3 &= I'_{10}\cdot T_3=\widetilde{I}_{10}\cdot M_{6\times 7}\cdot M^{-1}{}_{7\times 6}\cdot \widetilde{T}_3=[c_4,c_3,c_2,c_1,c_0,\underline{\underline{0}},\gamma]\cdot M\cdot M^{-1}\cdot \begin{bmatrix} y_3{}^4 \\ y_3{}^3 \\ y_3{}^2 \\ y_3 \\ 1 \\ \beta_3 \\ \underline{\underline{0}} \end{bmatrix} \\ &= c_4 y_3{}^4+c_3 y_3{}^3+c_2 y_3{}^2+c_1 y_3+c_0 \\ &= \varpi(y_3-x_1)(y_3-x_2)(y_3-x_3)(y_3-x_4)+\sigma \end{aligned}$$

$$\underline{H(\sigma_3)\neq H(\sigma)\rightarrow P_{10}\bigcap S^0{}_{14}=\phi\rightarrow 10\leq 14}$$

Fig. 4.4: Example of comparison between index and trapdoor

- **Rag_Search**$(\mathcal{I}, T_Q)$: The Rag_Search algorithm is conducted by the cloud server to determine records that satisfy the query Q. Fig. 4.4 illustrates an example of search algorithm between index $\mathcal{I}_{d_u}$ and trapdoor T_Q. For each record $r_u \in \mathcal{R}$, the cloud server executes the following steps to determine whether it satisfies the query or not.

  1) Based on the received index $I_{d_u} = \{I'_{d_u}, H(\sigma_u)\}$ and trapdoor T_Q, the cloud server calculates

$$\begin{aligned} \sigma_{uj} &= I'_{d_u}\cdot T_j \\ &= \tilde{I}_{d_u}\cdot [M]_{(\ell+2)\times(\ell+3)}\cdot [M^{-1}]_{(\ell+3)\times(\ell+2)}\cdot \tilde{T}_j \\ &= [c_\ell, ..., c_0, 0, \gamma_u]\cdot M\cdot M^{-1}\cdot [y_j^\ell, ..., y_j, 1, \beta_j, 0]^\top \\ &= c_\ell y_j^\ell + c_{\ell-1} y_j^{\ell-1} + ... + c_0 + 0\cdot\beta_j + \gamma_u\cdot 0 \\ &= \varpi_u(y_j-x_1)(y_j-x_2)...(y_j-x_\ell)+\sigma_u \pmod p. \end{aligned}$$

The idea of this step is to plug each trapdoor element y_j into function $F^P_{d_u}(x)$. Once there is a trapdoor element T_j to obtain $H(\sigma_{uj}) = H(\sigma_u)$, the server learns $\{x_1, ..., x_\ell\} \cap \{y_j\} \neq \emptyset$, which means either

$$d_u < w_L \longleftarrow H(\widetilde{S^1_{w_L}}||K) \cap H(P_{d_u}||K) \neq \emptyset$$
$$\text{or } w_H < d_u \longleftarrow H(P_{d_u}||K) \cap H(S^0_{w_H}||K) \neq \emptyset$$

The algorithm outputs $d_u \notin [w_L, w_H]$ that leads to the unsatisfied record d_u. Then, the algorithm directly move to determine the next record.

2) When all of $H(\sigma_{u1}) \neq H(\sigma_u), ..., H(\sigma_{ug}) \neq H(\sigma_u)$, it indicates that

$$w_L \leq d_u \longleftarrow H(\widetilde{S^1_{w_L}}||K) \cap H(P_{d_u}||K) = \emptyset$$
$$d_u \leq w_H \longleftarrow H(P_{d_u}||K) \cap H(S^0_{w_H}||K) = \emptyset.$$

The algorithm outputs $d_u \in [w_L, w_H]$ and inserts r_u into $\mathcal{D}(Q)$ as the query result.

After scanning all of the records in $\mathcal{R}$, the cloud server returns query result $\mathcal{D}(Q)$ to the data user.

4.4.3 Index Generation Optimization

The main time cost of Bld_Index and Rag_Search algorithm is in the calculation of polynomial function $F^P_{d_u}(x)$. To improve their execution speeds, we reduce the comparing units into smaller groups by following the grouping method. Instead of constructing a single polynomial function $F^P_{d_u}(x)$ based on the hash value of the entire elements $\{x_1, ..., x_\ell\}$, we first shuffle the prefix elements, then evenly partition them into several groups. Then, we construct several polynomial functions based on the x_i in each group, such that each polynomial function degree is smaller. During the search phase, trapdoor elements are checked with each group of x_i to find the first unsatisfied group. That is, each trapdoor element y_j is plugged into each group of polynomial function. The purpose of this grouping is to reduce the degree of $F^P_{d_u}(x)$ and the number of elements in I'_{d_u} and T_j. It accelerates the speed of Bld_Index (index building) and Rag_Search (trapdoor comparison).

The following shows an example of the grouping method. The prefix strings of an 8-bit attribute value d are partitioned into 2 groups in random order $\{x_1, x_8, x_2, x_4 \mid x_5, x_7, x_3, x_6\}$. It constructs a polynomial function for each group with the same random nonce σ. Finally, the index of d is the cascading of coefficients $\{[I'_d]_1 \mid\mid [I'_d]_2, H(\sigma)\}$ in each polynomial function. Since the elements are grouped in shuffled order, the comparison in grouping method provides the same security guarantee as for the original scheme. That is, the server is unable to learn which binary bit differs between the attribute value and the query.

$$\begin{gathered}
\varpi_1(x-x_1)(x-x_8)(x-x_2)(x-x_4)+\sigma \\
[I'_d]_1 = [c_{4,1}, ..., c_{0,1}, \gamma_1] \cdot M_{6\times 7}, \\
\varpi_2(x-x_5)(x-x_7)(x-x_3)(x-x_6)+\sigma \\
[I'_d]_2 = [c_{4,2}, ..., c_{0,2}, \gamma_2] \cdot M_{6\times 7}.
\end{gathered}$$

4.5 Implementation and Evaluation

Apart from the security improvements, in this section we implement our scheme and evaluate its practicality under different parameter settings. Specifically, we evaluate the performance of our scheme against the OPE scheme [74] and ORE [78] scheme to illustrate the benefits and costs of our scheme's security enhancements. As shown in Table 2.2, comparable encryption (CE) is a variant of the OPE and ORE schemes. However, it leaks the numerical difference of indexes during the search by default, which is less secure than the small-domain

ORE. Therefore, in this section we do not compare the search efficiency of our scheme with comparable encryption. The RSSE [80] scheme and PBtree [79] scheme rely on special tree structures built on the entire dataset to enhance query efficiency. Their searching speeds are highly related to the query choice and distribution of attribute domain. Both the RSSE and PBtree schemes produce false positives to the range query results. However, our scheme only addresses how to build secure indexes for attribute values and trapdoor for the range query condition. There is no false positive in the query results of our scheme. Hence, we do not compare the performance of our scheme with that of the RSSE and PBtree schemes. Certainly, all of the tree structures proposed in the RSSE and PBtree schemes for the entire dataset can be directly applied to our scheme, which can achieve the same efficiency with better security.

4.5.1 Experimental Settings

Implementation

Our scheme was implemented in C. For the cryptographic details, we chose the security parameter λ as 128 bits. For the sake of fairness, we used the PRF function implemented in the ORE scheme [78, 115] as the hash function H in our scheme, which is an AES-128 construction. Unlike the PRF function, we took the output of our AES-based hash function to be the domain of $\{0,1\}^{128}$. We then converted the outputs of H as mpz_t integers and used GMP-5.0.1 library [116] for all of the arithmetic operations. We chose the maximum value of 128 bits as the modulus $p = 2^{128} - 1$ and random invertible matrix M was generated with integer entries. All of the following experiments were conducted on a computer running macOS Sierra 10.12.5 with 4GB memory and a 1.3-GHz Intel Core i5 CPU. For the evaluation of the ORE scheme [78], we directly used the C implementation of FastORE [115]. For the evaluation

Table 4.1: Experimental settings for evaluations

Fig.	Attribute Value (bits)	Group Size (bits)
4.5a,4.5b	16, 24, 32, 48 ,64	2, 8, 16
4.6a,4.7a,4.8a		12, 16
4.6b,4.7b	64	4, 8, 12, 14, 16
4.8b	16	

of the OPE scheme [74], we used the C++ implementation from CryptDB [117, 118].

Range query is widely used in different scenarios, in which the query attribute is one of the important factors affecting performance. Specifically, date and time are often used range query attributes (i.e., search for the records during the time period from '01/07/2017 00:00:00' to '01/08/2017 00:00:00'). Since date and time are usually displayed in long format, the schemes use integers with longer bit length to represent this type of attribute value. For instance, Li et al. convert the check-in time attribute field in the Gowalla dataset (a geo-social network dataset [119]) to 32-bit integers [79]. Another type of range query attribute has a relatively small domain (i.e., product price, employee salary, and student scores), which can be represented as integers with shorter bit length. For example, the annual salary field in the USPS dataset (a dataset of employee records of US Postal Service [120]) can be represented as 24-bit integers.

To evaluate the performance of the scheme under different scenarios, we randomly choose the attribute value and the upper or lower bound of a query as the integers with different bit lengths. Our scheme used the grouping method introduced in Section 4.4.3. The group size is the number of elements involved in building each group of indexes. To discuss the performance of schemes under group size, we also varied the group size within the maximum bit length of attribute. Each measurement was found by taking the mean value over 50-10^7 iterations. Table 4.1 lists the parameter settings of each experimental result figure.

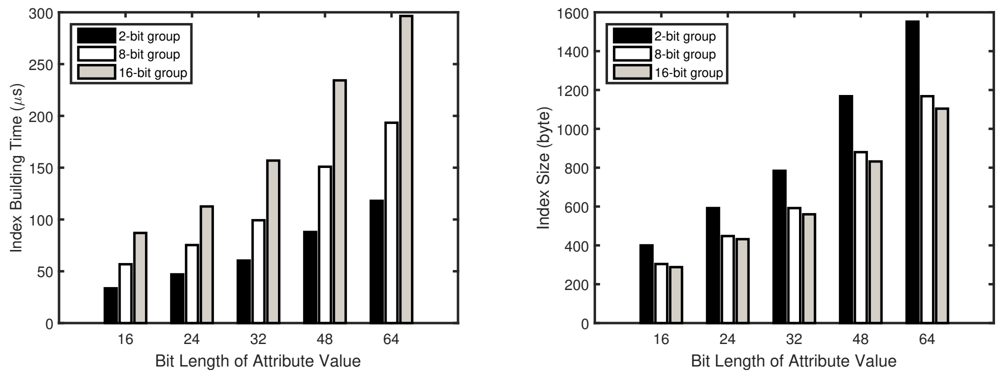
4.5.2 Experimental Result

Evaluation with different parameter settings

The index size and building time of our scheme for different-sized attribute values and groups are illustrated in Fig. 4.5a and Fig. 4.5b, respectively. The bars in both figures have the same trends. That is, the index building time and size increase with the increased bit length of attribute value. For the same bit length of attribute value, the index building time increases with increased group size, as shown in Fig. 4.5a. Whereas for the same bit length of the attribute value, the index size decreases with increased group size, shown in Fig. 4.5b. The opposite results are due to the larger group size, resulting in a longer time spent on constructing the index, but with a smaller number of groups. Each group brings one more set of polynomial coefficients to the entire index. The index size becomes smaller when there is a smaller number of groups. Hence, in our scheme there exists a trade-off between index building time and index size when choosing group size.

Evaluation of the index building time

The index building time comparison between OPE [74], ORE [78] schemes and our scheme for different-sized attribute values and groups is shown in Fig. 4.6a and Fig. 4.6b, respectively. The random oracle in ORE and our scheme use the same AES-based construction. The group size for the ORE scheme is the number of bits in each block. Each group or block is also the comparing unit



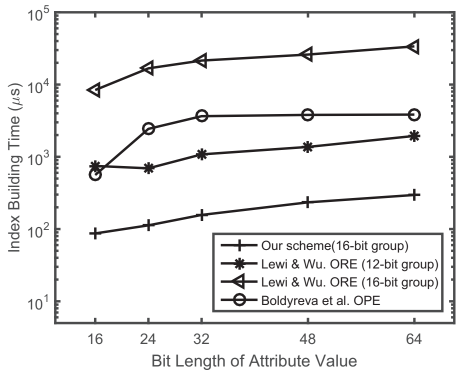
(a) Index building time of our scheme. (b) Index size of our scheme.

Fig. 4.5: Evaluation with different parameter settings

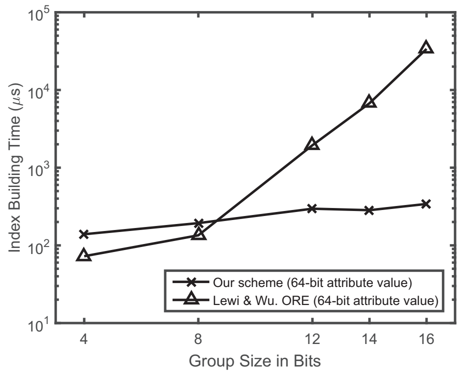
for ORE and our scheme. As shown in Fig. 4.6a, among the three schemes, our scheme has the minimum index building time, while the ORE scheme with the 16-bit group has the maximum index building time. The time cost of the OPE scheme is slower than the ORE scheme with 12-bit, but still much faster than the ORE scheme with 16-bit groups. Specifically, the index building time of our scheme is on average over 16 times faster compared to the OPE scheme, and on average over 6 times faster compared to the ORE scheme with 12-bit groups. We continue to discuss the influence of group size on both the ORE scheme and our scheme. The index building time of our scheme and the ORE scheme under different group sizes is shown in Fig. 4.6b. Note that the index building time of the ORE scheme grows more rapidly than our scheme when the group size is larger than 8 bits.

Evaluation of the index size

The index size comparison between OPE [74], ORE [78] schemes and our scheme for different-sized attribute values and groups is shown in Fig. 4.7a and Fig. 4.7b, respectively. In Fig. 4.7a, the index of the OPE scheme is the smallest due to its ciphertext still being a numerical value. The ORE scheme with the 16-bit group also has the largest index size. The index size of our scheme with the 16-bit group is on average 2.92 times smaller, compared to

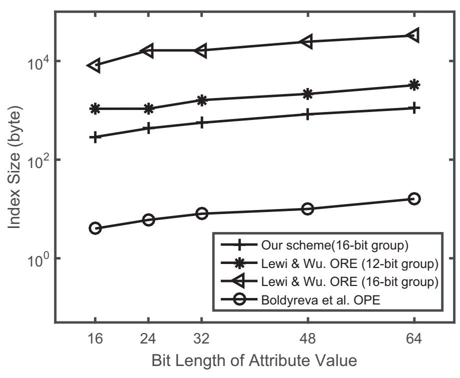


(a) Index building time of our scheme, OPE and ORE schemes with different-sized attribute values.

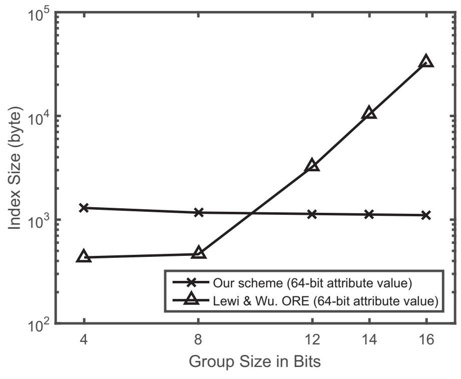


(b) Influence of group size on the index building time of our scheme and ORE scheme.

Fig. 4.6: Evaluation of the index building time



(a) Index size of our scheme, OPE and ORE schemes with different-sized attribute values.



(b) Influence of group size on the index size of our scheme and ORE scheme.

Fig. 4.7: Evaluation of the index size

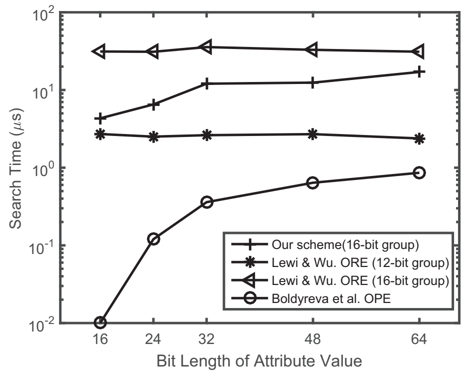
the ORE scheme with the 12-bit group. Fig. 4.7b illustrates the index size of ORE and our scheme under a different group size. The index size of the ORE scheme also grows quickly when the group size is larger than 8 bits. However, the index size of our scheme decreases with the increased group size.

As a similar trend reflected from Fig. 4.6b and Fig. 4.7b, the index generation of the ORE scheme is only efficient by setting a relatively small block size. The main reason is that the ORE scheme relies on a small-domain ORE construction in each group/block to achieve its best-possible security. That is, the indexes in each ORE group leak nothing but the ordering of their plaintexts. The cost of constructing the small-domain ORE indexes grows linearly in the size of group message space. For a b-bit group, the index size of each ORE group is linear with $2^b - 1$ [78]. Whereas in our scheme, the index of each group is linear with the bit length of the group b. Therefore, the index generation time and index size of our scheme are much faster and smaller than those of the ORE scheme when the group size is large.

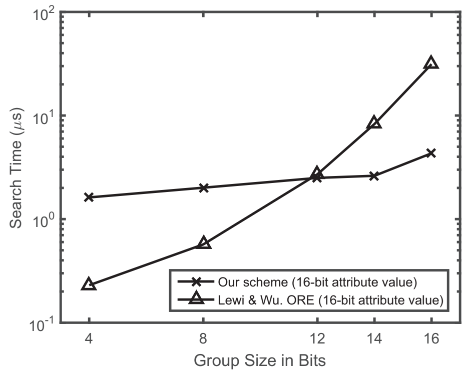
Evaluation of the search time

Our scheme leaks strictly less information than the OPE [74] and ORE [78] schemes. In this experiment, we evaluate how much search efficiency has been sacrificed due to the security improvements in our scheme. Fig. 4.8a compares the search times of the OPE and ORE schemes and ours with different-sized attribute values. The search time is the time used to compare an index with a trapdoor value of a range query's upper or lower bound. The three lower lines (i.e., including marker symbols $+$, $*$, and $\circ$) in Fig. 4.8a indicate that our scheme is slower than OPE and ORE (12-bit group) schemes. This is because our scheme is designed to hide more sensitive information than the other two schemes. Hence, it requires a longer time to complete a range comparison. The search time of the OPE scheme is the fastest, since it directly compares two numerical values. Nevertheless, it leaks much more information than the ORE scheme, and indeed, much more information than our scheme does, as shown in Table 2.2. Hence, OPE encrypted indexes are vulnerable to inference attack, which directly leaks the frequency and order relationship of attribute values.

To hide the bit of an attribute value that differs its index from the trapdoor, our scheme builds the index from a single function $F^P_{d_u}(x)$ using the entire prefix strings of the attribute value d_u. In the grouping method proposed in Section 4.4.3, we shuffle the prefix strings of each attribute value before



(a) Search time of our scheme, OPE and ORE schemes with different-sized attribute values.



(b) Influence of group size on the search time of our scheme and ORE scheme.

Fig. 4.8: Evaluation of the search time

partitioning them into different groups. Thus, the group of index elements that stops algorithm Rag_Search does not correspond to the same group of bits in the plaintext of d_u. As a result, our scheme does not leak the relative differences between attribute values to the same query in the range comparison process. Meanwhile, the server in our scheme has to compare the trapdoor with all or part of the attribute value's prefix strings, which takes a longer time. In the ORE scheme, its index elements keep the same binary order of the plaintext of the attribute value. Once the index differentiates trapdoor on a group/block of high bits, the ORE scheme will stop the comparison. Consequently, the ORE scheme is faster than our scheme when the group size is small (e.g., no more than 12 bits). However, the ORE scheme tells the server about the first bit or group of bits that differs between an index and a trapdoor. This leakage shows the relative distances between different attribute values to the same query. Even for the best secure setting of the ORE scheme (i.e., small-domain ORE), our scheme still provides stronger security. The ORE comparison result inevitably discloses whether an unmatched attribute value is larger than the upper bound or smaller than the lower bound of a range query, while the trapdoor generated in our scheme is not deterministic, which can prevent the search pattern leakage of the ORE scheme.

From the two upper lines (i.e., including marker symbols $\triangleleft$ and $+$) in Fig. 4.8a, interestingly, we can observe an opposite trend. The search time of our scheme is 3.89 times faster than the ORE scheme when the group size is 16 bits. To further explain the reason for this result, we discuss the influence of group size on both the ORE and our scheme in Fig. 4.8b. Since the OPE scheme does not process the range comparison in groups of bits, we cannot test the search time of the OPE scheme in Fig. 4.8b. It shows that our scheme is both more secure and faster, compared to the ORE scheme, when the group size is larger than 12 bits. This is because the ORE scheme has adopted the small-domain ORE construction in each group, such that it needs to compare the trapdoor with 2^b index elements in each b-bit group. When the group size increases, the ORE scheme provides higher security, but at the expense of ORE's search efficiency, which declines greatly. In our scheme, on the contrary, each b-bit group has exactly b index elements. The increasing group size has less impact on the search speed of our scheme. Additionally, our scheme provides the same security guarantees under different group sizes, as discussed before. Therefore, we can conclude that under the ORE scheme, it is difficult to achieve both security and search efficiency simultaneously. To achieve higher security, the ORE scheme must sacrifice more efficiency than our scheme.

4.6 Security Analysis

Inspired by the approach in [79], we analyse the security of our scheme in this section to prove that it achieves the defined security goals.

4.6.1 Ciphertext Indistinguishability Proof

Theorem 1 *The privacy-preserving range query scheme* $\mathsf{SSE}^{\mathsf{RAG}}$ *is ciphertext indistinguishability secure with the leakage function* $\mathcal{L}$ *from Definition 3, assuming that the keyed hash function* H *is a secure pseudo-random function.*

Proof: We use contradiction to prove the Theorem 1. Supposing that our scheme $\mathsf{SSE}^{\mathsf{RAG}}$ is not ciphertext indistinguishability secure, then there exists a polynomial-sized adversary $\mathcal{A}_1$ that can win Game 1 in Section 4.2.4 with an advantage greater than $\mathsf{negl}(\lambda)$. We construct a polynomial-sized adversary $\mathcal{B}_1$, which uses $\mathcal{A}_1$ as a subroutine to break the pseudo-randomness of function H.

Specifically, adversary $\mathcal{B}_1$ plays with a challenger $\mathcal{C}$ in a pseudo-randomness game. At the same time, $\mathcal{B}_1$ interacts with $\mathcal{A}_1$ by attempting to "fake" the challenger in Game 1. Before answering the queries of $\mathcal{A}_1$, the challenger gives $\mathcal{B}_1$ a function f and algorithm $\mathsf{Bld\_Index}$ and $\mathsf{Gen\_Trapdoor}$. However, the keyed hash functions H in $\mathsf{Bld\_Index}$ and $\mathsf{Gen\_Trapdoor}$ are replaced with the function f, which is either pseudo-random or truly random and takes as input $\{0,1\}^{\lambda}$ and outputs $\{0,1\}^{s(\lambda)}$.

Next, we describe how the adversary $\mathcal{B}_1$ provides the view for $\mathcal{A}_1$ and answers $\mathcal{A}_1$'s queries in the following phases.

- **Setup**: Based on the function f, adversary $\mathcal{B}_1$ generates a large collection of attribute values $\mathbb{D}$ and a sequence of range queries $\mathbb{Q}$, constructs a system parameter params and sends them to the adversary $\mathcal{A}_1$.

- **Phase 1**: Adversary $\mathcal{A}_1$ adaptively sends $\mathcal{B}_1$ an attribute value collection $\mathcal{D}_i$ and a range query Q_i as it did in Phase 1 of Game 1. $\mathcal{B}_1$ replies $\mathcal{A}_1$

with indexes $\mathcal{I}_i$ and trapdoor T_{Q_i} by running algorithm $\mathsf{Bld\_Index}$ and $\mathsf{Gen\_Trapdoor}$ given by the challenger, where $1 \leq i \leq q_1$.

- **Challenge**: Adversary $\mathcal{A}_1$ submits two collections of attribute values $\mathcal{D}_0$ and $\mathcal{D}_1$. $\mathcal{B}_1$ randomly chooses a bit $b \in \{0, 1\}$ and replies to $\mathcal{A}_1$ with index $\mathcal{I}_b \leftarrow \mathsf{Bld\_Index}(\mathsf{sk}, \mathcal{D}_b)$ as before.

- **Phase 2**: For request $q_1 + 1 \leq i \leq q$, adversary $\mathcal{A}_1$ repeats the same process as in Phase 1 and finally obtains $\langle \mathcal{I}_1, ..., \mathcal{I}_q, \mathcal{I}_b \rangle$ and $\langle T_{Q_1}, ..., T_{Q_q} \rangle$.

  All range queries are chosen under the restriction of $\mathcal{L}(\mathcal{D}_0, Q_i) = \mathcal{L}(\mathcal{D}_1, Q_i)$, where $1 \leq i \leq q$.

- **Guess**: After q requests, $\mathcal{A}_1$ outputs a bit $b^{'}$. If $b^{'} = b$, then $\mathcal{B}_1$ outputs 1 to the challenger in the pseudo-randomness game. This means that $\mathcal{B}_1$ guesses that function f is pseudo-random. If $b^{'} \neq b$, then $\mathcal{B}_1$ outputs 0 to the challenger. This means that $\mathcal{B}_1$ guesses that function f is truly random.

Next, we prove the following two claims to indicate that $\mathcal{B}_1$ can distinguish whether function f is pseudo-random or truly random with non-negligible probability over $1/2$.

Claim1. If f is a pseudo-random function, then

$$\Pr[\mathcal{B}_1^f = 1 | f : \{0,1\}^\lambda \times \{0,1\}^\ell \to \{0,1\}^{s(\lambda)}] > \frac{1}{2} + \mathsf{negl}(\lambda).$$

Claim2. If f is a truly random function, then

$$\Pr[\mathcal{B}_1^f = 0 | f : \{0,1\}^\ell \to \{0,1\}^{s(\lambda)}] = \frac{1}{2}.$$

Claim 1 Proof: If f is a pseudo-random function, then $\mathcal{A}_1$'s observation is identical to what is viewed during Game 1 defined in Section 4.2.4. Since we have assumed that $\mathcal{A}_1$ can win Game 1 with an advantage greater than $\mathsf{negl}(\lambda)$, then adversary $\mathcal{B}_1$ can also output 1 with an advantage greater than $\mathsf{negl}(\lambda)$. Thus, we prove Claim 1.

Claim 2 Proof

Claim 2 means that adversary $\mathcal{A}_1$ cannot distinguish $\mathcal{D}_0$ from $\mathcal{D}_1$ when the function f is truly random. Next, we prove Claim 2 from the following aspects.

- **Indexes of any attribute values $\langle \mathcal{I}_1, ..., \mathcal{I}_q \rangle$ and $\mathcal{I}_b$ reveal no difference between $\mathcal{D}_0$ and $\mathcal{D}_1$ to $\mathcal{A}_1$.**

In the index generation, algorithm Bld_Index uses the function f to map the prefix string of an attribute value into a string. If f is a truly random function, the output of f is a random string. Then, upon these random strings, algorithm Bld_Index assigns different records with different random nonces. Specifically, random nonces ϖ_u and σ_u are embedded in constructing the function $F^P_{d_u}(x)$. And random nonce γ_u is used in the matrix multiplication. The last element of index I_{d_u} is the $f(\sigma_u)$. Hence, each index is identical to a series of random strings. Adversary $\mathcal{A}_1$ is unable to detect that the secret key sk, queried attribute value d_u have not been used. Given two different indexes $I_{d_{u1}}$ and $I_{d_{u2}}$, it is infeasible for $\mathcal{A}_1$ to determine whether they are created from the same attribute value ($d_{u1} = d_{u2}$) or two different attribute values ($d_{u1} \neq d_{u2}$). Additionally, the index sizes of attribute values in $\mathcal{D}_0$ and $\mathcal{D}_1$ are required to be the same. $\mathcal{A}_1$ also cannot distinguish $\mathcal{D}_0$ from $\mathcal{D}_1$ based on the index sizes shown in the $\mathcal{I}_b$.

- **Trapdoors of any range queries $\langle T_{Q_1}, ..., T_{Q_q} \rangle$ reveal no difference between $\mathcal{D}_0$ and $\mathcal{D}_1$ to $\mathcal{A}_1$.**

In algorithm Gen_Trapdoor, each element of trapdoor T_{Q_i} ($Q_i = [w_{iL}, w_{iH}]$) is initially constructed using the new 1-encoding of w_{iL} or 0-encoding of w_{iH}. This encoding approach is different from that used in index generation. Then, algorithm Gen_Trapdoor uses function f to map each encoding into a string. As indicated before, when f is a truly random function, its output is a random string. Later, each trapdoor element T_{Q_i} is assigned with a different random nonce β_j used in the matrix multiplication. Hence, elements of any trapdoor to adversary $\mathcal{A}_1$ are identical to a series of random strings. Therefore, it is infeasible for $\mathcal{A}_1$ to correlate any trapdoor values T_{Q_i} with the attribute values in $\mathcal{D}_0$ and $\mathcal{D}_1$.

- **Matched records in $\mathcal{D}_b(Q_i) \leftarrow$ Rag_Search$(\mathcal{I}_b, T_{Q_i})$ reveal no difference between $\mathcal{D}_0$ and $\mathcal{D}_1$ to $\mathcal{A}_1$.**

$\mathcal{D}_0$ and $\mathcal{D}_1$ are required to have the same access pattern under all issued queries, that is, $\mathcal{D}_0(Q_i) = \mathcal{D}_1(Q_i)$. And every prefix string of a satisfied attribute value has no common elements with any encodings of a range query, that is $H(\sigma_{u1}) \neq H(\sigma_u), ..., H(\sigma_{ug}) \neq H(\sigma_u)$. Hence, $\mathcal{A}_1$ cannot trivially distinguish $\mathcal{D}_0$ from $\mathcal{D}_1$ based on their matched records under any range queries.

- **The case of how each unmatched record in $\mathcal{D}_b$ fails to be returned by a range query reveals no difference between $\mathcal{D}_0$ and $\mathcal{D}_1$ to $\mathcal{A}_1$.**

Based on the property shown in Equation (4.7), there is no difference when comparing the index with the trapdoor elements of upper bound or lower bound of the same range query. That is, $\mathcal{A}_1$ cannot distinguish $\mathcal{D}_0$ from $\mathcal{D}_1$ based on the difference of $d_u > w_{iH}$ or $d_u < w_{iL}$. In addition, function $F^P_{d_u}(x)$ is constructed from all hashed prefix strings of d_u. The $\mathcal{A}_1$ cannot detect in which bit of d_u that differs d_u from Q_i. In the grouping method proposed in Section 4.4.3, we shuffle the prefix elements of each attribute value before partitioning them into different groups. Thus, the group of index elements that stops algorithm Rag_Search does not correspond to the same group of bits of d_u and leaks no difference between $\mathcal{D}_0$ and $\mathcal{D}_1$.

If f is a truly random function, $\mathcal{B}_1$ can correctly guess f with a probability of $1/2$. Thus, we prove Claim 2.

The function f given by the challenger is either pseudo-random or truly random with the same probability of $1/2$. Combining Claim 1 with Claim 2, we obtain

$$\begin{aligned}&\frac{1}{2}\Pr[\mathcal{B}_1^f = 1 | f : \{0,1\}^\lambda \times \{0,1\}^\ell \rightarrow \{0,1\}^{s(\lambda)}]\\ &+\frac{1}{2}\Pr[\mathcal{B}_1^f = 0 | f : \{0,1\}^\ell \rightarrow \{0,1\}^{s(\lambda)}] > \frac{1}{2} + \frac{\mathsf{negl}(\lambda)}{2}.\end{aligned}$$

This result indicates that $\mathcal{B}_1$ can distinguish a pseudo-random function from a truly random function with an advantage greater than $\mathsf{negl}(\lambda)$. However, this result contradicts the property of a pseudo-random function, which is impossible. Thus, we prove that an adversary $\mathcal{A}_1$ that can win in Game 1 with non-negligible probability over $1/2$ does not exist. Therefore, our scheme $\mathsf{SSE}^{\mathsf{RAG}}$ is ciphertext indistinguishability secure.

4.6.2 Trapdoor Indistinguishability Proof

Theorem 2 *The privacy-preserving range query scheme* $\mathsf{SSE}^{\mathsf{RAG}}$ *is trapdoor indistinguishability secure with the leakage function* $\mathcal{L}$ *from Definition 3, assuming that the keyed hash function* H *is a secure pseudo-random function.*

Proof: We use contradiction to prove the Theorem 2. Supposing that our scheme $\mathsf{SSE}^{\mathsf{RAG}}$ is not trapdoor indistinguishability secure, then there exists a polynomial-sized adversary $\mathcal{A}_2$ that wins in Game 2, defined in Section 4.2.4 with non-negligible probability over $1/2$. We construct a polynomial-sized adversary $\mathcal{B}_2$, which uses $\mathcal{A}_2$ as a subroutine to break the pseudo-randomness of function H.

Specifically, adversary $\mathcal{B}_2$ plays with a challenger $\mathcal{C}$ in a pseudo-randomness game. At the same time, $\mathcal{B}_2$ interacts with $\mathcal{A}_2$ by attempting to "fake" the challenger in Game 2. Before answering the queries of $\mathcal{A}_2$, the challenger gives $\mathcal{B}_2$ a function z and algorithm $\mathsf{Bld\_Index}$ and $\mathsf{Gen\_Trapdoor}$. However, the keyed hash functions H in $\mathsf{Bld\_Index}$ and $\mathsf{Gen\_Trapdoor}$ are replaced with the function z, which is either pseudo-random or truly random, and takes as input $\{0,1\}^{\lambda}$ and outputs $\{0,1\}^{s(\lambda)}$.

Next, we describe how the adversary $\mathcal{B}_2$ provides the view for $\mathcal{A}_2$ and answers $\mathcal{A}_2$'s queries in the following phases.

- **Setup**: Based on the function z, adversary $\mathcal{B}_2$ generates a large collection of attribute values $\mathbb{D}$ and a sequence of range queries $\mathbb{Q}$, constructs a system parameter params and sends them to the adversary $\mathcal{A}_2$.

- **Phase 1**: Adversary $\mathcal{A}_2$ adaptively sends $\mathcal{B}_2$ an attribute value collection $\mathcal{D}_i$ and a range query Q_i as it did in Phase 1 of Game 2. $\mathcal{B}_2$ replies $\mathcal{A}_2$ with indexes $\mathcal{I}_i$ and trapdoor T_{Q_i} by running algorithm $\mathsf{Bld\_Index}$ and $\mathsf{Gen\_Trapdoor}$ given by the challenger, where $1 \leq i \leq q_1$.

- **Challenge**: Adversary $\mathcal{A}_2$ picks two range queries Q_0 and Q_1. $\mathcal{B}_2$ randomly samples a bit $c \in \{0,1\}$ and replies $\mathcal{A}_2$ with a trapdoor $T_{Q_c} \leftarrow \mathsf{Gen\_Trapdoor}(\mathsf{sk}, Q_c)$ as before.

- **Phase 2**: For request $q_1 + 1 \leq i \leq q$, adversary $\mathcal{A}_2$ repeats the same process as in Phase 1 and finally obtains $\langle \mathcal{I}_1, ..., \mathcal{I}_q \rangle$ and $\langle T_{Q_1}, ..., T_{Q_q}, T_{Q_c} \rangle$.

All collections of attribute values are chosen under the restriction of $\mathcal{L}(\mathcal{D}_i, Q_0) = \mathcal{L}(\mathcal{D}_i, Q_1)$, where $1 \leq i \leq q$.

- **Guess**: After q requests, $\mathcal{A}_2$ outputs a bit $c^{'}$. If $c^{'} = c$, then $\mathcal{B}_2$ outputs 1 to the challenger in the pseudo-randomness game. This means that $\mathcal{B}_2$ guesses that z is a pseudo-random function. If $c^{'} \neq c$, then $\mathcal{B}_2$ outputs 0 to the challenger. This means that $\mathcal{B}_2$ guesses that z is a truly random function.

Next, we prove the following two claims to indicate that $\mathcal{B}_2$ can distinguish whether function z is pseudo-random or truly random with an advantage greater than $\mathsf{negl}(\lambda)$.

Claim3. If z is a pseudo-random function, then

$$\Pr[\mathcal{B}_2^z = 1 | z : \{0,1\}^\lambda \times \{0,1\}^\ell \to \{0,1\}^{s(\lambda)}] > \frac{1}{2} + \mathsf{negl}(\lambda).$$

Claim4. If z is a truly random function, then

$$\Pr[\mathcal{B}_2^z = 0 | z : \{0,1\}^\ell \to \{0,1\}^{s(\lambda)}] = \frac{1}{2}.$$

Claim 3 Proof: If z is a pseudo-random function, then $\mathcal{A}_2$'s observation is identical to what is viewed during Game 2 defined in Section 4.2.4. Since we have assumed that $\mathcal{A}_2$ can win Game 2 with non-negligible probability over 1/2, then adversary $\mathcal{B}_2$ can also output 1 with non-negligible probability over 1/2. Thus, we prove Claim 3.

Claim 4 Proof

Claim 4 means that adversary $\mathcal{A}_2$ cannot distinguish Q_0 from Q_1 when the function z is truly random. Next, we prove Claim 4 from the following aspects.

- **Trapdoors of any range queries $\langle T_{Q_1}, ..., T_{Q_q} \rangle$ and T_{Q_c} reveal no difference between Q_0 and Q_1 to $\mathcal{A}_2$.**

As indicated in the second point of Section 4.6.1, algorithm $\mathsf{Gen\_Trapdoor}$ uses function z and different random nonce β_j in generating each trapdoor element T_{Q_i}. When the function z is truly random, the elements of any trapdoor are identical to a series of random strings. Adversary $\mathcal{A}_2$ is unable to detect that the secret key sk, the upper bound w_{iH} and lower bound w_{iL} have not been used. Given two different trapdoors $T_{Q_{i1}}$ and $T_{Q_{i2}}$, it is infeasible for $\mathcal{A}_2$ to determine whether they are created from the same query range or not, and whether the upper or lower bound of Q_{i1} is larger or smaller than that of Q_{i2}. Additionally, the trapdoor sizes of Q_0 and Q_1 are required to be the same. $\mathcal{A}_2$ cannot distinguish Q_0 from Q_1 based on the trapdoor size of T_{Q_c}.

- **Indexes of any attribute values $\langle \mathcal{I}_1, ..., \mathcal{I}_q \rangle$ reveal no difference between Q_0 and Q_1 to $\mathcal{A}_2$**

As mentioned in the first two points of Section 4.6.1, both algorithm $\mathsf{Bld\_Index}$ and $\mathsf{Gen\_Trapdoor}$ add different random nonces when generating each index and trapdoor element. Adversary $\mathcal{A}_2$ is unable to correlate any index value I_{d_u} with the trapdoor of Q_0 or Q_1.

- **Matched records in $\mathcal{D}_i(Q_c) \leftarrow \mathsf{Rag\_Search}(\mathcal{I}_i, T_{Q_c})$ reveal no difference between Q_0 and Q_1 to $\mathcal{A}_2$.**

Q_0 and Q_1 are required to have the same access pattern with all issued collections of attribute values, that is, $\mathcal{D}_i(Q_0) = \mathcal{D}_i(Q_1)$. And every trapdoor element has no common elements with any prefix string of a satisfied attribute value, that is $H(\sigma_{u1}) \neq H(\sigma_u), ..., H(\sigma_{ug}) \neq H(\sigma_u)$. Hence, $\mathcal{A}_2$ cannot trivially distinguish Q_0 from Q_1 based on their matched records in any attribute value collections.

- **The case of how each unmatched record fails to be returned by the range query Q_c reveals no difference between Q_0 and Q_1 to $\mathcal{A}_2$.**

To hide which trapdoor element that differs between the trapdoor and index, algorithm Gen_Trapdoor places the trapdoor elements of each issued range query in shuffled order. That is, trapdoor element T_j causing the $H(\sigma_{uj}) = H(\sigma_u)$ does not correspond to the same bit of lower/upper bound of a range query. Hence, the trapdoor element that differs in each unmatched attribute value from query Q_c does not leak any difference between Q_0 and Q_1 to $\mathcal{A}_2$.

If z is a truly random function, $\mathcal{B}_2$ can correctly guess z with a probability of $1/2$. Thus, we prove Claim 4.

The function z given by the challenger is either pseudo-random or truly random, with the same probability of $1/2$. Combining Claim 3 with Claim 4, we obtain

$$\begin{aligned}&\frac{1}{2}\Pr[\mathcal{B}_2^z = 1 | f : \{0,1\}^\lambda \times \{0,1\}^\ell \to \{0,1\}^{s(\lambda)}] \\ &+ \frac{1}{2}\Pr[\mathcal{B}_2^z = 0 | f : \{0,1\}^\ell \to \{0,1\}^{s(\lambda)}] > \frac{1}{2} + \frac{\mathsf{negl}(\lambda)}{2}.\end{aligned}$$

This result indicates that $\mathcal{B}_2$ can distinguish a pseudo-random function from a truly random function with non-negligible probability over $1/2$. However, this result contradicts the property of a pseudo-random function, which is impossible. Thus, we prove that there does not exist an adversary $\mathcal{A}_2$ that can win in Game 2 with an advantage greater than $\mathsf{negl}(\lambda)$. Thus, our scheme $\mathsf{SSE}^{\mathsf{RAG}}$ is trapdoor indistinguishability secure.

To sum up, based on the Definition in 6, we can conclude that our privacy-preserving range query scheme $\mathsf{SSE}^{\mathsf{RAG}}$ is both ciphertext and trapdoor indistinguishability secure with the leakage function $\mathcal{L}$ from Definition 3, assuming that the keyed hash function H is a secure pseudo-random function.

4.7 Conclusion

For cloud data storage, data privacy and security are two key concerns. Although sensitive data can be encrypted before they are stored in the cloud, the encrypted data can hardly be processed efficiently. Hence, a lightweight solution is required to satisfy both high security and high efficiency requirements. In this chapter, we study the problem of range query over encrypted data. The main idea is to transform the range comparison to a privacy-preserving set intersection operation. To protect record privacy, our scheme builds searchable encrypted indexes for records that are secure against inference attack. To ensure the privacy of range queries, non-deterministic encryption, which has not been achieved in range query before, is proposed to hide the search pattern of queries. During range comparison, our scheme neither leaks the order relationship between the upper/lower bound of a range query and the encrypted index, nor produces false positives in the query results. We have implemented our scheme and evaluated its performance in comparison with other schemes. The comparison results indicate that our scheme has a shorter index size and search time than the order-revealing encryption scheme when the processing unit is large. Meanwhile, our scheme only leaks the access pattern, and is proved to be more secure than existing schemes.

5

Access Pattern Hidden Query over Encrypted Data through Multi-clouds

5.1 Introduction

Searchable symmetric encryption (SSE) allows database searching to be conducted over encrypted data. Basically, using the secure indexes generated by the data owner, a cloud server matches the encrypted query keywords (called trapdoor) with the secure indexes to find the required data. As a result, the cloud server can learn the search pattern (i.e., if the query has been issued before) and access pattern (i.e., which encrypted documents satisfied which trapdoors). Based on these observations, the cloud server still can analyze and estimate the plaintext value of documents or query even without decryption keys. For instance, when the query results of any two trapdoors have large overlapping, it means that their corresponding query keywords must be closely related. The cloud server can reveal the trapdoor keywords, by calculating the co-occurrence frequency of trapdoors in the ciphertext dataset and finding the same frequency in the plaintext dataset [92]. In addition, some keywords have unique appearance frequencies in the datasets [94]. Attackers can identify the trapdoor keywords by mapping the same but unique result count between the ciphertext and plaintext dataset. All these two attacks take advantage of access pattern leakage to recover keywords of trapdoors.

The main reason behind access pattern leakage attacks is that the most searchable encryption schemes can only safeguard the confidentiality of document and query value at rest. The strongest security model available for Searchable Symmetric Encryption (SSE) schemes is formulated by Curtmola et al. in [91]. The schemes satisfying this model ensure the indexes and trapdoors themselves do not leak the document and query value even when the attackers adaptively issuing the queries. However, this model allows the attacker to learn the statistics information from the observation of searching results (i.e., access pattern of queries). Moreover, most of these schemes are designed to perform on the single server. Hence, the server is able to gain all the trapdoors and their query results. When the dataset to be searched is large enough to cover the entire domain knowledge, the server can statistically analyze the actual value of queries or documents. Oblivious Random Access Memory (ORAM) [22, 23] has been proposed to address the access pattern leakage problem by hiding the physical memory access patterns of executed programs from the server. However, the computational complexity of most ORAM schemes is high and they can only support limited types of queries. Similar to our approach, vertical fragmentation [102, 24] seeks to use multiple servers to make attackers unable to discover the sensitive association between document attributes, (e.g., separating documents of employee name and salary on two servers). But this technique cannot prevent the access pattern leakage attacks on a single attribute, and can lead to a long response time for multi-keyword search.

As shown in the Fig. 5.1, to tackle access pattern leakage attack, we adopt the servers on multiple clouds. Our contributions are summarized as follows:

- We distribute both database documents (rows) and queries among different cloud servers. Since none of the servers can access the entire database of documents and queries, they cannot identify the accurate statistical relationship of the query results.

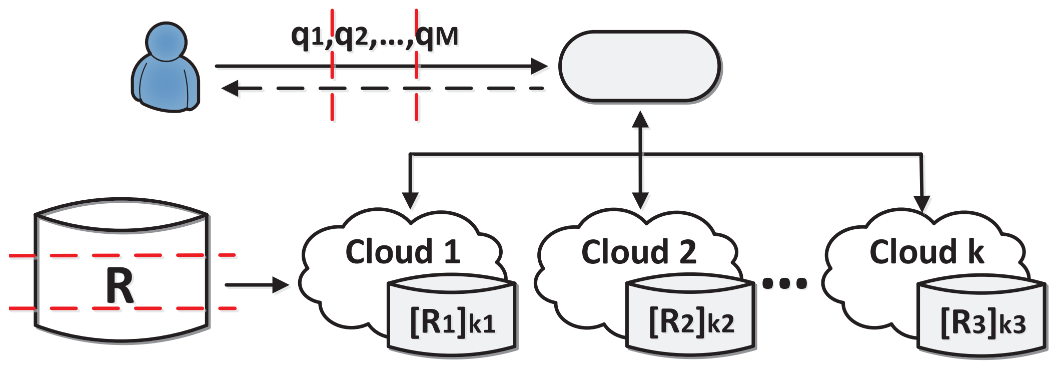


Fig. 5.1: Assigning records and queries among multiple clouds.

- We formulate the document and query assignment as an optimization problem to minimize both query response time and access pattern disclosure. We then determine an optimal assignment strategy using a minimum cut algorithm.

- Our distribution strategy supports any secure SSE schemes. Before storing documents to cloud servers, all documents are protected using the same SSE scheme, but each server is given different secret keys to encrypt and build the secure index for the documents assigned to it. Accordingly, different trapdoors are created based on the server that is executing the query.

- We adopt the servers on multiple clouds, but in the same region. This deployment requires less bandwidth and reduces the chance for a collusion attack among servers (i.e., compared with deployment for servers in different datacenters of the same cloud).

- We evaluate the assignment strategy output from our approach with a real world dataset. The numerical results indicate that on average 13% access pattern information can be saved by our assignment strategy, without sacrificing query response time. In addition, our algorithm is scalable under different realistic settings.

The remainder of this chapter is organized as follows. Section 5.2 presents and formulates our proposed record and query assignment model. Section 5.3 describes how to search for an optimal assignment strategy by solving the assignment optimization problem. The experimental results and conclusion are shown in Section 5.4 and 5.5, respectively.

5.2 Secure Assignment Strategy Formulation

In this section, we explain our security strategies to reduce the access pattern leakage by distributing both records and queries among multiple clouds. Both the IKK and count attacks rely heavily on extracting the information from a complete plaintext dataset [121]. Hence an attacker cannot launch these attacks if only a fraction of the plaintext dataset is known [93]. In other words, if a server only knows partial information about the encrypted dataset, the above attacks can be minimized, even with the complete knowledge of the plaintext dataset.

- **Record Fragmentation** Both matrix element $M_c(i, j)$ and $count(R_q)$ are computed from the trapdoor query result set R_q. When the server is unable to access the correct query results of the trapdoors, the attacker finds it difficult to deduce their keywords by matching their frequencies. Padding is one of the approaches to hide actual query results, by adding dummy records and identifiers before building the secure indexes [92]. However, padding unsatisfied records causes false positives in the query results, and brings extra communication overhead [93, 121]. In this chapter, we consider using multi-clouds to tackle the security problems. We will remove the records in the trapdoor query results to other cloud servers, in order to make the co-occurrence probability and keyword frequency observed by a single server incorrect. In other words, each server can only return part of the correct query results. The same query needs to be executed parallelly on multiple servers.

- **Query Distribution** Since IKK attack is based on the entire matrix M_c, the order of different query pairs co-occurrence probability can be used for keyword recovery attack. Apart from disturbing each query result set, hiding the differences between different $M_c(i,j)$ is also necessary. We adopt the same strategy proposed in [92]. Instead of using a padding approach, we will distribute queries with similar results on the same server, in order to make the query result of trapdoors on the same cloud server as similar as possible.

Our scheme can perform the keyword search in two phases. In the first phase, the cloud server only searches for the query result which provides the identities of records that contain the query keywords. To prevent the server from knowing the query result, the returned record identities are encrypted. After decrypting the record identities, the client retrieves the records in the second phase. In both phases, the client sends the request in batches without repeated keywords or records. For instance, a client sends a query request which includes $|X|$ different keywords. Basically, in the query result of the previous scheme, each record is linked to one keyword. By adopting the batch query method, each record is linked to multiple keywords. This method makes the access pattern of a keyword indistinguishable from the other $|X|-1$ keywords. As a result, a cloud server cannot associate a returned record with a particular query. Therefore, even if an attacker is able to collect the access pattern from multiple cloud servers at the same time, it is still difficult to accurately identify the keyword of the trapdoor based on statistical analysis.

As shown in Fig. 5.1, the fundamental question is: "How should the records and queries be distributed among the multiple cloud servers in order to minimize the response time while satisfying certain security requirements?" In this section, we will introduce the strategy of record and query assignment, to satisfy the above introduced security requirements. Let $\mathcal{R} = \{r_1, ..., r_N\}$ denote a set of database records with several attributes. Let $\mathcal{Q} = \{q_1, ..., q_M\}$ denote a sequence of queries written in the development process. Each of the queries has different combinations of keywords to be searched on $\mathcal{R}$. In addition, let $\mathcal{S} = \{S_1, ..., S_{|\mathcal{S}|}\}$ denote a group of cloud servers, which will be assigned both records and query tasks. Before distribution, the data owner builds a matrix $\mathcal{B}$ to describe the query results of all of the queries, in which the rows represent the record set and columns represent the query set. Each element $b_{r,q} \in \mathcal{B}$ is set to 1, if record $r \in \mathcal{R}$ satisfies query $q \in \mathcal{Q}$. The qth column vector of $\mathcal{B}$ denotes the query result of q, that is $\mathcal{B}_q$. Matrix $\mathcal{B}$ is safely kept by the data owner.

$$b_{r,q} = \begin{cases} 0, & ifr \notin Result(q) \\ 1, & ifr \in Result(q) \end{cases} \tag{5.1}$$

5.2.1 Record and Query Assignment

To prevent the aforementioned attacks, our scheme would assign both records and queries among cloud servers. We assume that the partial records and queries on a single cloud server cannot reveal the statistical property of the original access pattern.

Definition 7 ***Record and Query Assignment:*** *We define $\mathcal{A} = (\mathcal{A}_R, \mathcal{A}_Q)$ as an assignment of records $\mathcal{R}$ and queries $\mathcal{Q}$ to a set of cloud servers $\mathcal{S}$. The process of assignment is to horizontally and vertically partition the elements of matrix $\mathcal{B}$ into $|\mathcal{S}|$ blocks. Finally, each cloud server obtains a sub-matrix of $\mathcal{B}$ with a subset of records and queries. To avoid repetition of the returned results, each element $b_{r,q} \in \mathcal{B}$ will be assigned to one of the cloud servers.*

$$\mathcal{B} = \begin{array}{c} \begin{array}{cccc|c} q_1 & q_2 & \dots & q_M & \\ \end{array} \\ \left(\begin{array}{cccc} 0 & 1 & \dots & 0 \\ 1 & 0 & \dots & 1 \\ 1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & \dots & 1 \end{array}\right) \begin{array}{c} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_N \end{array} \end{array} \rightarrow \begin{array}{cccc} q_1 & q_2 & \dots & q_M \\ 0 & 1 & \dots & 0 \\ 1 & 0 & \dots & 1 \\ 1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & \dots & 1 \end{array} \tag{5.2}$$

Matrix $\mathcal{A}_R$ describes the placement of records on cloud servers. Each element $x_{r,s} \in \mathcal{A}_R$ is set to 1, if record $r \in \mathcal{R}$ is assigned to cloud server $s \in \mathcal{S}$. Since each record is assigned to at least one cloud server, $\mathcal{A}_R$ has no zero row, that is $\sum_{s=1}^{s=|\mathcal{S}|} x_{r,s} \geq 1$.

$$\mathcal{A}_R = \begin{array}{c} \begin{array}{ccccc} S_1 & \dots & \mathcal{A}_R^s & \dots & S_k \end{array} \\ \left(\begin{array}{ccccc} x_{1,1} & \dots & x_{1,s} & \dots & x_{1,|\mathcal{S}|} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{|\mathcal{R}|,1} & \dots & x_{|\mathcal{R}|,s} & \dots & x_{|\mathcal{R}|,|\mathcal{S}|} \end{array}\right) \begin{array}{c} r_1 \\ \vdots \\ r_N \end{array} \end{array} \tag{5.3}$$

Matrix $\mathcal{A}_Q$ indicates the placement of queries on cloud servers. Each element $y_{q,s} \in \mathcal{A}_Q$ is set to 1, if query $q \in \mathcal{Q}$ is distributed to cloud server $s \in \mathcal{S}$. Each row vector of $\mathcal{A}_Q$ represents an assignment of a query among the cloud servers, that is $\mathcal{A}_Q^q = [y_{q,1}, ..., y_{q,|\mathcal{S}|}]$. Since each query must be distributed to at least one cloud server, $\mathcal{A}_Q$ has no zero row, that is $\sum_{s=1}^{s=|\mathcal{S}|} y_{q,s} \geq 1$.

$$\mathcal{A}_Q = \begin{array}{c} \begin{array}{ccccc} S_1 & \cdots & \mathcal{A}_Q^s & \cdots & S_k \end{array} \\ \begin{pmatrix} y_{1,1} & \cdots & y_{1,s} & \cdots & y_{1,|\mathcal{S}|} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ y_{q,1} & \cdots & y_{q,s} & \cdots & y_{q,|\mathcal{S}|} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ y_{|\mathcal{Q}|,1} & \cdots & y_{|\mathcal{Q}|,s} & \cdots & y_{|\mathcal{Q}|,|\mathcal{S}|} \end{pmatrix} \end{array} \begin{array}{c} \\ q_1 \\ \vdots \\ \mathcal{A}_Q^q \\ \vdots \\ q_M \end{array} \tag{5.4}$$

We define the assignment on each cloud server s as $\mathcal{A}^s$, which is a tuple of two column vectors, that is $\mathcal{A}^s = (\mathcal{A}_R^s, \mathcal{A}_Q^s)$. Specifically, $\mathcal{A}_R^s = [x_{1,s}, ..., x_{|\mathcal{R}|,s}]^\intercal$ and $\mathcal{A}_Q^s = [y_{1,s}, ..., y_{|\mathcal{Q}|,s}]^\intercal$ correspond to the sth column in matrix $\mathcal{A}_R$ and $\mathcal{A}_Q$, respectively. It also means that each cloud server only obtains a subset of records and queries.

After applying the horizontal and vertical partition on matrix $\mathcal{B}$, we need to ensure records duplicated into multiple cloud servers cannot be associated. So, each cloud server reassigns different record IDs and uses different keys to encrypt records and their attribute names. Accordingly, different trapdoors are generated for the same query sent to different clouds. Finally, the independent searchable encryption scheme is applied within each cloud server.

5.2.2 Information Disclosure

In this subsection, we will clearly identify the information disclosure from the assignment $\mathcal{A}$ to the cloud server set $\mathcal{S}$. We assume that different cloud servers will not collude after applying different searchable encryption schemes. We firstly analyze the information disclosure on each single cloud server. By observing the response results of all of the queries, each cloud server s can rebuild another matrix $\mathcal{B}^s$ to describe the query results on its own. That is, for each $\forall x_{r,s} \in \mathcal{A}_R^s = 1$ and $\forall y_{q,s} \in \mathcal{A}_Q^s = 1$, the corresponding row and column vector in $\mathcal{B}$ is chosen to form $\mathcal{B}^s \subsetneq \mathcal{B}$ which is a sub-matrix of $\mathcal{B}$. The qth column vector of $\mathcal{B}^s$ denote the query result of q on the cloud s, that is $\mathcal{B}_q^s$.

$$\begin{array}{c c c c c c} \mathcal{A}_Q^s & y_{1,s} & y_{2,s} & \dots & y_{M,s} = 1 & \mathcal{A}_R^s \\ & 0 & 1 & \dots & 0 & x_{1,s} \\ & 1 & 0 & \dots & 1 & x_{2,s} \\ \mathcal{B}^s = & \vdots & \vdots & \ddots & \vdots & \vdots \\ & 0 & 1 & \dots & 1 & x_{N,s} = 1 \end{array} \tag{5.5}$$

Definition 8 ***Information Disclosure:*** *We define the information disclosure from the assignment $\mathcal{A}^s$ on each cloud server as a set of queries $D(\mathcal{A}^s)$ that does not satisfy the following constrains:*

- *Referring to the leakage definition proposed in [92], the result differences between any two queries on each cloud server s should be less than a threshold value $0 \leq \alpha \leq 1$. We use the Hamming distance between any two column vectors in $\mathcal{B}^s$ for evaluation. For any two queries $1 \leq \forall q_1, q_2 \leq cols(\mathcal{B}^s)$, if the*

$$\frac{Hamming(\mathcal{B}_{q_1}^s, \mathcal{B}_{q_2}^s)}{rows(\mathcal{B}^s)} > \alpha \tag{5.6}$$

  then q_1 and q_2 are regarded as insecure queries included to set $D(\mathcal{A}^s)$, that is $D(\mathcal{A}^s) \cup \{q_1, q_2\}$;

- *None of the queries on the cloud server s should include the entire query result, that is, for $1 \leq \forall q \leq cols(\mathcal{B}^s)$, if $\mathcal{B}_q^s = \mathcal{B}_q$, then $D(\mathcal{A}^s) \cup \{q\}$.*

We define the overall information disclosure of an assignment $\mathcal{A}$ as the ratio of insecure queries to the total queries.

$$D(\mathcal{A}) = \frac{\left| \bigcup_{s=1}^{s=k} D(\mathcal{A}^s) \right|}{|\mathcal{Q}|} \tag{5.7}$$

5.2.3 Query Response Time

We use the query response time observed by the data user to evaluate the performance of records and queries distribution across multiple cloud servers. We consider that all of the queries $q_1, ..., q_M$ have already been identified by the developers. And each query has a corresponding execute frequency, denoted as a vector $\mathcal{F} = [f_1, ..., f_M]$, $\sum_{q=1}^{q=M} f_q = 1$. The response time of each cloud server consists of two parts: local processing time and result transmission time. Since queries are sent to a cloud server in sequential order, the local processing of each cloud server can be modelled as a $M/M/1$ queue. The time cost of query result transmission is a constant calculated based on the result size and network bandwidth, that is $Tran(q, s) = Size(\mathcal{B}_q^s)/Net(s)$.

Obviously, the query arrival rate to each cloud server is related to the queries assigned to it. For any query q with executing frequency f_q, the cloud server will receive the requests with the rate of f_q, as long as it stores the records queried by q. We consider the query arrival rate to follow the Poisson distribution. Since the additive property of Poisson distribution, the arrival rate of any cloud server s is the summation of all of the execute frequencies of queries on them. Let θ_s denote the mean query arrival rate of server s, such that $\theta_s = \mathcal{F} \cdot \mathcal{A}_{\mathcal{Q}}^s = [f_1, ..., f_M] \cdot [y_{1,s}, ..., y_{M,s}]^\intercal$. We assume that the query processing time of each cloud's database follows the independent exponential distribution. Then we use μ_s to denote the mean query process rate of server s.

Since each query is forwarded to more than one cloud server and processed parallelly, based on the row vector $\mathcal{A}_Q^q$, the final mean response time of each query q equals to the maximum mean response time of all of the execute cloud servers, as follows.

$$T(q,\mathcal{A}) = \max_{y_{q,s}\in\mathcal{A}_Q^q, y_{q,s}=1} \left\{ \frac{\theta_s}{\mu_s - \theta_s} + Tran(q,s) \right\} \tag{5.8}$$

Thus, for an assignment $\mathcal{A}$ and a sequence of queries $\mathcal{Q} = \{q_1, ..., q_M\}$, the mean response time is shown as follows.

$$T(\mathcal{Q},\mathcal{A}) = \sum_{q\in\mathcal{Q}} T(q,\mathcal{A}) \cdot f_q \tag{5.9}$$

Definition 9 ***Optimal Assignment:*** *Given a matrix $\mathcal{B}$ built to represent the query result of a sequence of queries $\mathcal{Q}$ on a set of database record $\mathcal{R}$. We use the following optimization problem to find an optimal assignment $\mathcal{A}$ of all of the elements in $\mathcal{B}$ to the cloud servers in $\mathcal{S}$ that minimizes the tradeoff between the total query response time $T(\mathcal{Q},\mathcal{A})$ and information disclosure $D(\mathcal{A})$, where γ is a nonnegative weight and η is the disclosure constraint.*

$$\begin{aligned} &\underset{\mathcal{A}\in(\mathcal{B}\times\mathcal{S})}{minimum} && T(\mathcal{Q},\mathcal{A}) + \gamma D(\mathcal{A}) \\ &subject\ to && D(\mathcal{A}) \leq \eta \end{aligned} \tag{5.10}$$

5.3 Assignment Optimization

In this section, we describe how to find a record and query assignment with minimal query response time and information disclosure. The optimization problem formulated in equation (5.10) is NP-hard to solve. We present the following heuristic algorithm to find an approximate optimal assignment.

5.3.1 Algorithm Overview

The challenge of solving the optimizing problem (5.10) lies in the mutual affection between record and query placements. When any record in the result set of a query q is assigned to a cloud server s, then query q also needs to be sent to the same cloud in order to return the entire query result. The response time of query q is reduced, but the workload and information leakage on server s is increased. Hence, instead of determining the placement of record and query separately, we take each element $b_{r,q} \in \mathcal{B}$ as a decision unit. Thus, the optimization problem (5.10) equals finding an assignment of elements in $\mathcal{B} = \{b_{1,1}, ..., b_{1,M}; ...; b_{N,1}, ..., b_{N,M}\}$ to one of the cloud servers $S_1, ..., S_k$, such that the objective function $G(\cdot)$ is minimized.

```
Input: matrix $\mathcal{B}$; set of cloud servers $\mathcal{S}$; empty assignment $\mathcal{A}$; objective
  function $G(\cdot)$
Output: assignment $\mathcal{A}$: $G(\mathcal{A})$ is minimal
Initialization: $\mathcal{A} \leftarrow$ an arbitrary assignment ;
do
    $Stop \leftarrow 0$;
    for all pairs of cloud servers $\{s, t\} \in \mathcal{S}$ do
        $\mathcal{A}' = \text{EXCHANGE}(s, t, \mathcal{A})$;
        if $G(\mathcal{A}') < G(\mathcal{A})$ then
            $\mathcal{A} = \mathcal{A}'$;
            $Stop \leftarrow 1$;
        end
    end
while $Stop == 1$;
return $\mathcal{A}$;
```

Algorithm 1: Optimal assignment search algorithm.

5.3.2 Assignment Minimization via Minimum Cut

The structure of our algorithm is shown in Algorithm 1. Referring to the idea in [122], the algorithm repeatedly executes a for-loop until the first for-loop that cannot make any improvement to the current assignment, the algorithm outputs it. In the for-loop, the algorithm performs the same iteration (EXCHANGE)

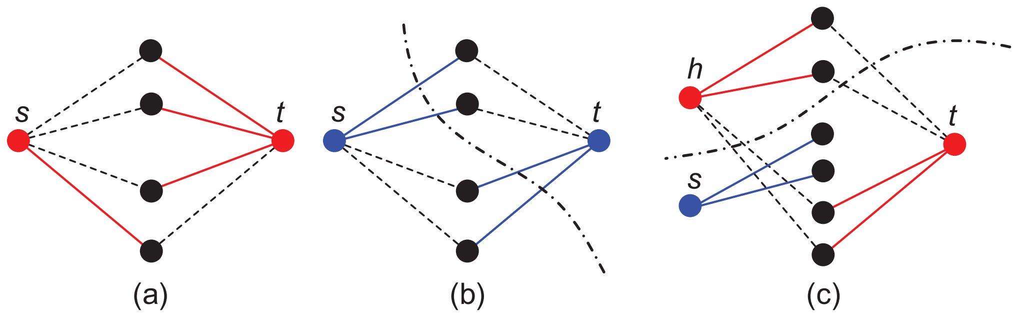


Fig. 5.2: Search for optimal assignment of elements to a pair of cloud servers via minimum cut.

for each pair of cloud servers. The output of each EXCHANGE is an optimal assignment of input elements to one pair of cloud servers. The variable “Stop” will be set to 1, if the output of any EXCHANGE is an assignment $\mathcal{A}'$ with lower objective function value.

Fig. 5.2 depicts how to solve EXCHANGE via the minimum $s-t$ cut. We first construct a graph by considering two cloud servers as terminal nodes (s,t) and all the elements in $\mathcal{B}$ as the middle nodes connecting s or t. The solid line in Fig. 5.2(a)) indicates an assignment of element to a cloud server. In the initial phase of the Algorithm 1, each element in $\mathcal{B}$ is arbitrarily linked to one of the cloud servers in $\mathcal{S}$. In each iteration, the inputs of EXCHANGE are a pair of cloud servers and an assignment $\mathcal{A}$ to be improved. The purpose of algorithm EXCHANGE is to improve the assignment $\mathcal{A}$ by connecting each element to one of the terminals. The principle of adjustment is to seek a new assignment that with lower objective function value under the constraint of information disclosure. In the initial phase, the algorithm adds edges from both servers to all the elements that are linked to one of the two servers, as shown by the dashed lines in Fig. 5.2(a)).

Since each element can only link to one of the terminal nodes, this adjustment can be achieved via the minimum $s-t$ cut on Fig. 5.2(a)). The output of a minimum $s-t$ cut is a set of edges (hit by the dash-dotted lines in Fig. 5.2(b)) with the minimal cost value to separate terminal node s and t, which is also a better assignment $\mathcal{A}'$ from elements to cloud servers. Some elements previously assigned to server s in $\mathcal{A}$ are now assigned to server t in $\mathcal{A}'$, and vice versa. The assignment of the rest of the cloud servers $h \neq s, t \in \mathcal{S}$ is the same $\mathcal{A}^h = \mathcal{A}'^h$. The algorithm repeatedly chooses another pair of cloud servers (s, h), links the elements of s and h with one another, and finds the minimum cut set, as shown in Fig. 5.2(c).

5.4 Performance Evaluation

In this section, we conduct experiments to analyze the performance of the assignment strategy output from Algorithm 1. In particular, we demonstrate that our assignment strategy resists access pattern attack, offers high query efficiency and is scalable under different realistic settings.

5.4.1 Experimental Settings

Dataset $\mathcal{R}$ and queries $\mathcal{Q}$

We adopt the Enron email dataset for evaluation, which is a set of email documents of 150 Enron corporation employees, sent from 2000 to 2002 [94]. This dataset has 30,109 email documents with 77,000 unique keywords after removing the stopwords [23]. We consider each document as a record, and generate queries by uniformly choosing them from the most frequent keywords. The documents that contain a certain keyword mean the document is the result of the query with that keyword. Since the query frequency does not influence information disclosure, we assume that the frequencies of all of the queries are equal and their sum is 1. So the query arrival rate θ_s of each cloud server is related to the total number of queries assigned to it.

Table 5.1: Parameter settings for performance evaluations

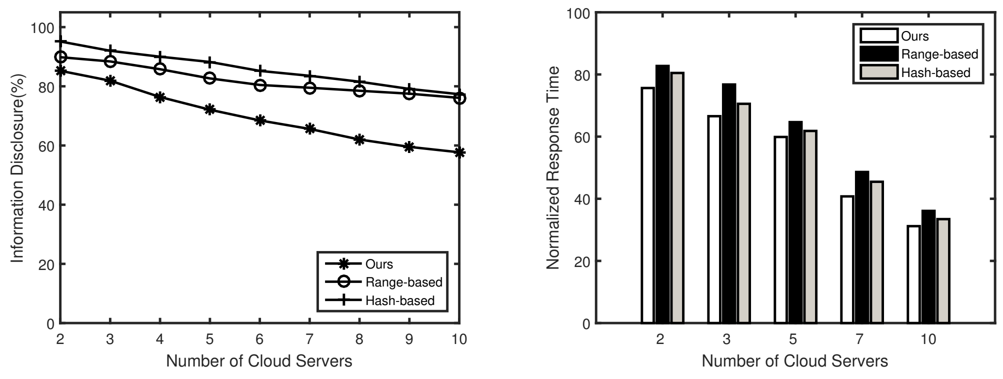
Fig.	No. of $\mathcal{Q}$	No. of $\mathcal{S}$	γ
5.3a,5.3b	500	[2,10]	1
5.4a,5.4b	[200,1000]	10	1
5.5a	500	10	1
5.5b	500	[2,10]	0, 1

Cloud Servers $\mathcal{S}$

Since the service rate of cloud database service is dynamic and difficult to control, we evaluate the performance of our assignment approach via the numerical experiments. We assume that the service rate μ_s of all of the cloud servers is equal to 1 in the following experiments. The time cost of transmitting back each single record is assumed to be 1 ms. Therefore, the bandwidth cost equals to the number of query results on each cloud server. In all of the experiments, we set $\alpha = 0.5$ and $\eta = 0.9$. We use the gco-v3.0 library [123] to implement the minimum cut algorithm and obtain our assignment policies based on the different parameter settings. Table 5.1 indicates the specific parameter settings of each figure.

Benefits of our assignment strategy

We compare the record and query assignment strategy designed by our approach with the record placement policies in the distributed databases. The usual sharding policies in the NoSQL database (*e.g.*, MongoDB) are Range-based and Hash-based [124]. In the Range-based sharding, records are divided into different chunks based on the range of shard key values. In the Hash-based sharding, records are evenly distributed among cloud servers based on the hash of shard key values.



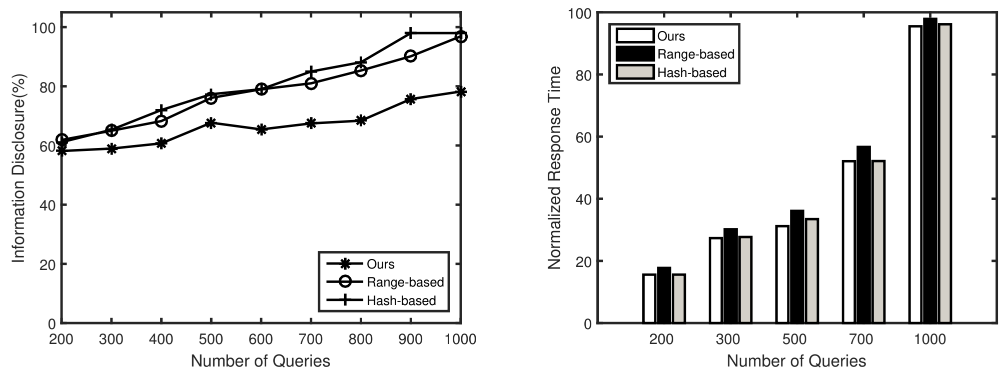
(a) Information disclosure. (b) Response time.

Fig. 5.3: Benefits of our assignment strategy on information disclosure and query response time.

5.4.2 Experimental Results

Fig. 5.3a shows the information disclosure of three data assignment policies with the growing number of cloud servers. The leakage always declines with a greater number of servers, since fewer access patterns are observed by each cloud server. Hash-based policy leaks more information than Range-based policy, because the records with "closer" key values are stored on the same server. So queries executed on the same server have similar results. Our assignment strategy discloses the lowest percentage of information over the other two. Under different numbers of cloud servers, our approach saves an average of 14.7% information disclosures compared to the Range-based policy. Compared to the Hash-based policy, our approach can save more. The benefits of our assignment over other policies increases with a greater number of cloud servers. This is because our approach achieves better assignment solutions with more cloud servers.

Fig. 5.3b depicts the response time of three placement strategies. The Hash-based policy has less query response time than the Range-based policy does, because records in Hash-based policy can be parallelly executed on different cloud servers. However, in these two partitioning policies, the query workload on each cloud server has not been considered. In our assignment strategy,



(a) Information disclosure. (b) Response time.

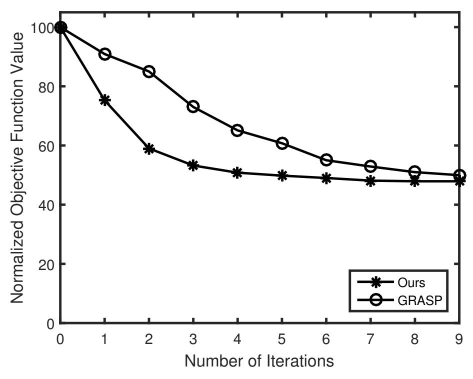
Fig. 5.4: Influence of the number of queries on the benefits of our assignment

using the access pattern as input leads to a faster query response time, which saves an average of 6% of the query response time compared to the Hash-based policy.

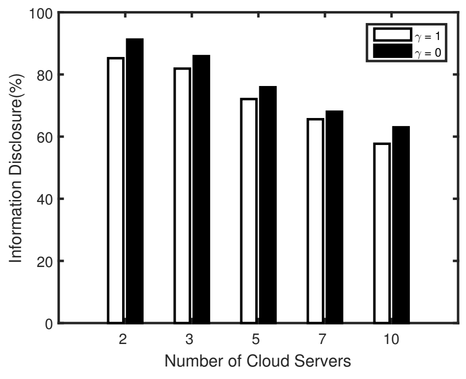
Influence of the number of queries

Fig. 5.4a shows the information disclosure of three data assignment policies with a different number of queries. The leakage increases with a greater number of queries, since more queries are gathered in the same cloud server, and more access patterns are disclosed. Our assignment strategy leaks the lowest percentage of information. Under different numbers of queries, our approach saves an average of 13% information disclosures compared with the Range-based policy. This is because our approach optimizes information disclosures. Fig. 5.4b illustrates the impact of the number of queries on query response time. The response time increase as a greater number of queries. On average, 1.7% response time can be saved by our approach, compared with the Hash-based policy. This result indicates that our approach can avoid information disclosures without sacrificing query efficiency.

Convergence speed of Algorithm 1 Fig. 5.5a demonstrates the comparison of convergence speed between Algorithm 1 (minimum cut) and the greedy randomized adaptive local search procedure (GRASP) proposed in [125]. In



(a) Convergence speed.



(b) Influence of optimization goals.

Fig. 5.5: Evaluation of convergence speed and influence of optimization goals

each iteration of Algorithm 1, function EXCHANGE is invoked to find an optimal assignment of elements to a pair of cloud servers. In each iteration of GRASP algorithm, function CONSTRUCTION is invoked to greedily find a new assignment with lower objective function value. Fig. 5.5a shows that the objective function value of our algorithm drops quickly in the first several iterations, which indicates that our approach converges faster than GRASP.

Performance with different optimization goals In the equation (5.10), γ is an important parameter. When $\gamma = 0$, the optimization goal is only the total query response time under the information disclosure constraint. When the $\gamma = 1$, the optimization goal is both the query response time and information disclosure. Fig. 5.5b compares the information disclosures under different optimization goals by varying the number of cloud servers. For both versions of the optimization problem, the information leakage decreases as the number of cloud servers increases. This result shows that tradeoff optimization goal ($\gamma = 1$) achieves lower information disclosure by using multiple cloud servers.

5.5 Conclusion

Searchable encryption seeks to support untrusted third parties to conduct direct searching over encrypted data. However, recent research has found that searchable encryption is vulnerable to attacks, which exploit the statistical relationship or pattern identified from encrypted query results. In this chapter, we study the problem of access pattern leakage attack on searchable encryption under a multi-cloud environment. Basically, both database records and queries are distributed among different cloud servers, so that each cloud server can only have partial information about queries and their results. To minimize the query response time while protecting information disclosure, we formulate the record and query assignment as an optimization problem, and solve the problem (i.e., finding the best possible solution) by the minimum $s - t$ cut algorithm. Numerical results show that on average 13% access pattern information can be saved by our assignment strategy while maintaining good query response time. Additionally, there are many queuing models which are able to represent the feature of the query process in the multi-cloud environment. In this thesis, to facilitate the analysis, we only consider the basic $M/M/1$ queuing model for the performance analysis. In the future work, other advanced queuing models can also be studied. For example, $M/G/m/m + r$ queuing system has been used to model the performance of cloud data centers with single task arrivals and a task buffer of finite capacity [127]. Furthermore, a network of queues can also be considered in future performance analysis [126, 127].

6 Conclusion and Future Work

In this thesis, we have studied two important security issues for supporting Intercloud, namely distributed trust evaluation and secure data query. To this end, we have the following conclusions:

1. A distributed trust evaluation protocol with privacy protection for Intercloud

In Chapter 3, we have presented a distributed trust evaluation protocol with privacy protection for Intercloud. Compared to other protocols, this distributed protocol provides some distinctive features, particularly for the Intercloud environment. First, it supports user anonymity by means of blind signature, facilitating users to provide honest feedback without fear of a retaliatory attack. Second, by means of an innovative mechanism for storing feedback, feedback privacy can be protected by using homomorphic encryption with verifiable secret sharing. Third, it allows customized processing of evaluation results while protecting feedback privacy. A security model has been employed to evaluate the protocol for its effectiveness. Unlike many other distributed protocols, which only support static configuration, the protocol can still be effective when some of the parties are offline (i.e., supporting a dynamic configuration). Simulation results indicate the protocol can still function well when half of the parties are malicious or offline. Future work is being planned to further analyze and enhance the protocol (e.g., using distributed ledger technology). For example, various blockchains can be formed (e.g., among Intercloud Exchanges, CSPs and users). It is of interest to study how the blockchains can interact to support trust evaluation and other advanced functions for Intercloud.

2. Order-Hiding Range Query over Encrypted Data without Search Pattern Leakage

In Chapter 4, we have designed an order-hiding range query scheme. Our scheme solves the security leakage problem in existing secure range query schemes. To hide the statistical relationships among indexes, our scheme adopts the 0/1 encoding technique and constructs the indexes as the coefficients of the randomized polynomial function. To avoid leaking the comparison operator and search pattern, our scheme introduces a random invertible matrix in the generation of query trapdoors. We formally analyse sensitive information leakage in our scheme, and have proved it is secure under an IND-CKA2 security definition without restriction of the same search pattern. We implemented and assessed the performance of our scheme. The comparison results show that although the ORE scheme has a shorter index size and search time with small processing units, it is slower and has a longer index size than our scheme when the processing unit is large. On average, the index building time of our scheme is more than 16 times faster than the OPE scheme. Meanwhile, our scheme only leaks the access pattern, and is proved to be more secure than existing schemes. Future work is being planned to further enhance the query efficiency of our scheme and evaluate the performance on large volume dataset in the databases range query.

3. Access pattern hidden query over encrypted data scheme through multi-clouds

In Chapter 5, we have investigated the problem of access pattern leakage attack to searchable encryption in a cloud database. We distribute both documents and queries among different cloud servers, so that each cloud server can only observe partial information about access patterns. To achieve a minimum query response time and information disclosures, we formulate this record and query assignment as an optimization problem and search for the optimal assignment by the minimum $s - t$ cut. The numerical results show that on average 13% access pattern information can be saved by our assignment strategy, without sacrificing query efficiency. Our future work is to further enhance the security of our scheme to resist other access pattern leakage attacks and evaluate the performance of our scheme by combing it with multiple servers searchable symmetric encryption schemes. We are also going to include more factors as the optimization goal or constraint condition (e.g. electricity cost for maintaining the data query on cloud server) when searching for the optimal assignment strategy.

References

[1] Kiranbir Kaur, Sandeep Sharma, and Karanjeet Singh Kahlon. “Interoperability and Portability Approaches in Inter-Connected Clouds: A Review”. In: *ACM Comput. Surv.* 50.4 (2017), 49:1–49:40 (cit. on p. 1).

[2] Ana Juan Ferrer. “Inter-cloud Research: Vision for 2020”. In: *2nd International Conference on Cloud Forward: From Distributed to Complete Computing, Madrid, Spain, 18-20 October, 2016.* 2016, pp. 140–143 (cit. on p. 1).

[3] Justice Opara-Martins, Reza Sahandi, and Feng Tian. “Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective”. In: *J. Cloud Computing* 5 (2016), p. 4 (cit. on p. 1).

[4] Adel Nadjaran Toosi, Rodrigo N Calheiros, and Rajkumar Buyya. “Interconnected cloud computing environments: Challenges, taxonomy, and survey”. In: *ACM Comput. Surv.* 47.1 (2014), p. 7 (cit. on pp. 1, 9).

[5] Tram Truong-Huu and Chen-Khong Tham. “A novel model for competition and cooperation among cloud providers”. In: *IEEE Transactions on Cloud Computing* 2.3 (2014), pp. 251–265 (cit. on pp. 1, 10).

[6] Li Liu, Shuxian Gu, Dongmei Fu, Miao Zhang, and Rajkumar Buyya. “A New Multi-objective Evolutionary Algorithm for Inter-Cloud Service Composition”. In: *TIIS* 12.1 (2018), pp. 1–20 (cit. on pp. 1, 10).

[7] Stelios Sotiriadis, Nik Bessis, Ashiq Anjum, and Rajkumar Buyya. “An Inter-Cloud Meta-Scheduling (ICMS) Simulation Framework: Architecture and Evaluation”. In: *IEEE Trans. Services Computing* 11.1 (2018), pp. 5–19 (cit. on pp. 1, 10).

[8] Lirim Osmani, Salman Toor, Miika Komu, et al. “Secure cloud connectivity for scientific applications”. In: *IEEE Transactions on Services Computing* PP.99 (2015), pp. 1–13 (cit. on pp. 1, 10).

[9] Stelios Sotiriadis, Nik Bessis, Euripides GM Petrakis, et al. “Virtual machine cluster mobility in inter-cloud platforms”. In: *Future Generation Comp. Syst.* 74 (2017), pp. 179–189 (cit. on pp. 1, 10).

[10] Esha Barlaskar, Peter Kilpatrick, Ivor T. A. Spence, and Dimitrios S. Nikolopoulos. “MyMinder: A User-centric Decision Making Framework for Intercloud Migration”. In: *CLOSER 2017 - Proceedings of the 7th International Conference on Cloud Computing and Services Science, Porto, Portugal, April 24-26, 2017.* 2017, pp. 560–567 (cit. on pp. 1, 10).

[11] Stelios Sotiriadis and Nik Bessis. “An inter-cloud bridge system for heterogeneous cloud platforms”. In: *Future Generation Comp. Syst.* 54 (2016), pp. 180–194 (cit. on p. 1).

[12] *Cisco Intercloud Fabric.* Last Accessed: April, 2018, `https://www.cisco.com/c/en/us/products/cloud-systems-management/intercloud-fabric/index.html`. 2018 (cit. on p. 1).

[13] Steven Tadelis. “The Economics of Reputation and Feedback Systems in E-Commerce Marketplaces”. In: *IEEE Internet Computing* 20.1 (2016), pp. 12–19 (cit. on pp. 2, 40).

[14] Oluwabunmi Adewoyin, Roberto Araya, and Julita Vassileva. “Peer Review in Mentorship: Perception of the Helpfulness of Review and Reciprocal Ratings”. In: *Intelligent Tutoring Systems - 13th International Conference, ITS 2016, Zagreb, Croatia, June 7-10, 2016. Proceedings.* 2016, pp. 286–293 (cit. on pp. 2, 40).

[15] Christoph Bösch, Pieter H. Hartel, Willem Jonker, and Andreas Peter. “A Survey of Provably Secure Searchable Encryption”. In: *ACM Comput. Surv.* 47.2 (2014), 18:1–18:51 (cit. on pp. 3, 27).

[16] Bing Wang, Shucheng Yu, Wenjing Lou, and Y. Thomas Hou. “Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud”. In: *2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014.* IEEE, 2014, pp. 2112–2120 (cit. on pp. 3, 27).

[17] Bing Wang, Wei Song, Wenjing Lou, and Y. Thomas Hou. “Inverted index based multi-keyword public-key searchable encryption with strong privacy guarantee”. In: *2015 IEEE Conference on Computer Communications, INFOCOM 2015, Kowloon, Hong Kong, April 26 - May 1, 2015.* IEEE, 2015, pp. 2092–2100 (cit. on pp. 3, 27).

[18] Wenhai Sun, Shucheng Yu, Wenjing Lou, Y. Thomas Hou, and Hui Li. “Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud”. In: *2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014.* IEEE, 2014, pp. 226–234 (cit. on pp. 3, 27).

[19] Ning Cao, Cong Wang, Ming Li, Kui Ren, and Wenjing Lou. "Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data". In: *IEEE Trans. Parallel Distrib. Syst.* 25.1 (2014), pp. 222–233 (cit. on pp. 3, 27).

[20] Q. Wang, M. He, M. Du, et al. "Searchable Encryption over Feature-Rich Data". In: *IEEE Trans. Dependable and Secure Computing* 15.3 (2018), pp. 496–510 (cit. on pp. 3, 27).

[21] M. Du, Q. Wang, M. He, and J. Weng. "Privacy-Preserving Indexing and Query Processing for Secure Dynamic Cloud Storage". In: *IEEE Trans. Information Forensics and Security* 13.9 (2018), pp. 2320–2332 (cit. on pp. 3, 27).

[22] Emil Stefanov and Elaine Shi. "Multi-cloud oblivious storage". In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM. 2013, pp. 247–258 (cit. on pp. 3, 36, 122).

[23] Alexander Degitz, Jens Köhler, and Hannes Hartenstein. "Access Pattern Confidentiality-Preserving Relational Databases: Deployment Concept and Efficiency Evaluation." In: *EDBT/ICDT Workshops*. 2016 (cit. on pp. 3, 37, 122, 134).

[24] Sabrina De Capitani di Vimercati, Sara Foresti, Sushil Jajodia, et al. "Fragmentation in presence of data dependencies". In: *IEEE Transactions on Dependable and Secure Computing* 11.6 (2014), pp. 510–523 (cit. on pp. 3, 37, 122).

[25] Claude Castelluccia, Aldar C.-F. Chan, Einar Mykletun, and Gene Tsudik. "Efficient and provably secure aggregation of encrypted data in wireless sensor networks". In: *TOSN* 5.3 (2009), 20:1–20:36 (cit. on pp. 40, 51).

[26] Pascal Paillier. "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes". In: *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*. 1999, pp. 223–238 (cit. on pp. 40, 56).

[27] Paul Feldman. "A Practical Scheme for Non-interactive Verifiable Secret Sharing". In: *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*. 1987, pp. 427–437 (cit. on pp. 40, 56).

[28] Chamikara Jayalath, Julian Stephen, and Patrick Eugster. "From the cloud to the atmosphere: running mapreduce across data centers". In: *IEEE Transactions on Computers* 63.1 (2014), pp. 74–87 (cit. on p. 9).

[29] *Amazon's cloud bests those of Microsoft and Google by this reliability test.* Last Accessed: Sep, 2016, `http://www.networkworld.com/article/3020235/cloud-computing/and-the-cloud-provider-with-the-best-uptime-in-2015-is.html`. 2016 (cit. on p. 9).

[30] David Bernstein, Erik Ludvigson, Krishna Sankar, Steve Diamond, and Monique Morrow. "Blueprint for the intercloud-protocols and formats for cloud computing interoperability". In: *Proc. 4th Int. Conf. Internet and Web Appl. and Services.* 2009, pp. 328–336 (cit. on pp. 9, 16).

[31] Nikolay Grozev and Rajkumar Buyya. "Inter-Cloud architectures and application brokering: taxonomy and survey". In: *Software: Practice and Experience* 44.3 (2014), pp. 369–390 (cit. on pp. 9, 13, 16).

[32] Adel Nadjaran Toosi, Ruppa K Thulasiram, and Rajkumar Buyya. "Financial option market model for federated cloud environments". In: *Proc. 5th Int. Conf. Utility and Cloud Comput.* 2012, pp. 3–12 (cit. on p. 10).

[33] Ioan Petri, Javier Diaz-Montes, Mengsong Zou, et al. "Market models for federated clouds". In: *IEEE Transactions on Cloud Computing* 3.3 (2015), pp. 398–410 (cit. on p. 10).

[34] *EnergyPlus.* Last Accessed: Sep, 2016, `http://apps1.eere.energy.gov/build-ings/energyplus`. 2016 (cit. on p. 10).

[35] *Octave.* Last Accessed: Sep, 2016, `http://www.gnu.org/software/octave`. 2016 (cit. on p. 10).

[36] *Compact Muon Solenoid experiment at CERN's LHC.* Last Accessed: Sep, 2016, `http://cms.web.cern.ch`. 2016 (cit. on p. 10).

[37] Ben-Jye Chang, Yu-Wei Lee, and Ying-Hsin Liang. "Reward-based Markov chain analysis adaptive global resource management for inter-cloud computing". In: *Future Generation Comp. Syst.* 79 (2018), pp. 588–603 (cit. on p. 10).

[38] Courtney Powell, Takehiro Aizawa, and Masaharu Munetomo. "Design of an SSO authentication infrastructure for heterogeneous inter-cloud environments". In: *Pro. 3rd Int. Conf. Cloud Netw.* 2014, pp. 102–107 (cit. on pp. 10, 22).

[39] Kevin Walsh and John Manferdelli. "Intra-Cloud and Inter-Cloud Authentication". In: *2017 IEEE 10th International Conference on Cloud Computing (CLOUD), Honolulu, HI, USA, June 25-30, 2017.* 2017, pp. 318–325 (cit. on p. 10).

[40] VMware Knowledge Base. *Troubleshooting a virtual machine that has stopped responding (1007819).* Last Accessed: April, 2016, `https://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=1007819`. 2016 (cit. on p. 11).

[41] Narahari Dogiparthi. *Why did my Azure VM restart?* Last Accessed: April, 2016, `https://blogs.msdn.microsoft.com/mast/2013/09/23/why-did-my-azure-vm-restart/`. 2016 (cit. on p. 12).

[42] Adel Nadjaran Toosi, Rodrigo N. Calheiros, and Rajkumar Buyya. "Interconnected Cloud Computing Environments: Challenges, Taxonomy, and Survey". In: 47.1 (2014), 7:1–7:47 (cit. on pp. 13, 14).

[43] Royal Philips. *Philips to introduce rapid cloud-based recovery service for healthcare data in collaboration with Amazon Web Services*. Last Accessed: April, 2016, `http://www.philips.com/a-w/about/news/archive/standard/news/press/2016/20160222-Philips-to-introduce-rapid-cloud-based-recovery-service-for-healthcare-data-in-collaboration-with-Amazon-Web-Services.html`. 2016 (cit. on p. 15).

[44] David Bernstein and Deepak Vij. *IEEE PROJECT 2302 - Standard for Intercloud Interoperability and Federation (SIIF)*. Piscataway, NJ, 2012. URL: `https://standards.ieee.org/develop/project/2302.html` (cit. on p. 16).

[45] IEEE Intercloud Testbed Project. *An Open, Global, Cloud Interoperability Project*. Last Accessed: April, 2016, `http://www.intercloudtestbed.org/`. 2016 (cit. on p. 16).

[46] Pramod S Pawar, Muttukrishnan Rajarajan, S Krishnan Nair, and Andrea Zisman. "Trust model for optimized cloud services". In: *Trust Management VI*. Springer, 2012, pp. 97–112 (cit. on p. 21).

[47] Ryan KL Ko, Peter Jagadpramana, Miranda Mowbray, et al. "TrustCloud: A framework for accountability and trust in cloud computing". In: *Services (SERVICES), 2011 IEEE World Congress on*. IEEE. 2011, pp. 584–588 (cit. on p. 21).

[48] Xiaoyong Li and Junping Du. "Adaptive and attribute-based trust model for service level agreement guarantee in cloud computing". In: *Information Security, IET* 7.1 (2013), pp. 39–50 (cit. on p. 21).

[49] Kai Hwang and Deyi Li. "Trusted cloud computing with secure resources and data coloring". In: *Internet Computing, IEEE* 14.5 (2010), pp. 14–22 (cit. on p. 21).

[50] Carl Ellison, Bill Frantz, Butler Lampson, et al. *SPKI certificate theory*. Tech. rep. 1999 (cit. on p. 21).

[51] Eve Maler et al. "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML)". In: *OASIS, September* (2003) (cit. on p. 21).

[52] Russell Housley, W Polk, Warwick Ford, and David Solo. *Internet X. 509 public key infrastructure certificate and certificate revocation list (CRL) profile*. 2002 (cit. on p. 21).

[53] F John Krautheim, Dhananjay S Phatak, and Alan T Sherman. “Introducing the trusted virtual environment module: a new mechanism for rooting trust in cloud computing”. In: *Trust and Trustworthy Computing.* Springer, 2010, pp. 211–227 (cit. on p. 21).

[54] SoftwareInsider. *SoftwareInsider: Business Software Reviews & Research.* Last Accessed: April, 2016, `http://cloud-computing.softwareinsider.com`. 2016 (cit. on p. 22).

[55] Sheikh Mahbub Habib, Sebastian Ries, and Max Mühlhäuser. “Towards a trust management system for cloud computing”. In: *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on.* IEEE. 2011, pp. 933–939 (cit. on p. 22).

[56] Talal H Noor, Quan Z Sheng, Lina Yao, Schahram Dustdar, and Anne HH Ngu. “CloudArmor: Supporting reputation-based trust management for cloud services”. In: *IEEE Trans. Parallel Distrib. Syst.* 27.2 (2016), pp. 367–380 (cit. on pp. 22, 39).

[57] David Bernstein and Yuri Demchenko. “The IEEE Intercloud Testbed–Creating the Global Cloud of Clouds”. In: *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on.* Vol. 2. IEEE. 2013, pp. 45–50 (cit. on p. 22).

[58] David Bernstein and Deepak Vij. “Intercloud federation using via semantic resource federation API and dynamic SDN provisioning”. In: *Network of the Future (NOF), 2014 International Conference and Workshop on the.* IEEE. 2014, pp. 1–8 (cit. on p. 22).

[59] B Di Martino, G Cretella, A Esposito, et al. “Towards an Ontology-Based Intercloud Resource Catalogue-The IEEE P2302 Intercloud Approach for a Semantic Resource Exchange.” In: *IC2E.* 2015, pp. 458–464 (cit. on p. 22).

[60] Demchenko Yuri, Dumitru Cosmin, Filiposka Sonja, et al. “Open Cloud eXchange (OCX): A Pivot for Intercloud Services Federation in Multi-provider Cloud Market Environment”. In: *2015 IEEE International Conference on Cloud Engineering, IC2E*, pp. 472–479 (cit. on p. 22).

[61] Jaime Lloret, Miguel Garcia, Jesus Tomas, and Joel JPC Rodrigues. “Architecture and protocol for intercloud communication”. In: *Information Sciences* 258 (2014), pp. 434–451 (cit. on p. 22).

[62] David Bernstein and Deepak Vij. “Intercloud security considerations”. In: *Pro. 2nd Int. Conf. Cloud Comput. Technol. and Sci.* 2010, pp. 537–544 (cit. on p. 22).

[63] Mukesh Singhal, Santosh Chandrasekhar, Tingjian Ge, et al. “Collaboration in Multicloud Computing Environments: Framework and Security Issues.” In: *Computer* 46.2 (2013), pp. 76–84 (cit. on p. 23).

[64] Yuri Demchenko, Canh Ngo, Cees De Laat, and Chi-Kwan Lee. "Federated Access Control in Heterogeneous Intercloud Environment: Basic Models and Architecture Patterns". In: *Proc. 2nd Int. Conf. Cloud Eng.* 2014, pp. 439–445 (cit. on p. 23).

[65] David Bernstein and Deepak Vij. "Intercloud exchanges and roots topology and trust blueprint". In: *Proc. 11th Int. Conf. Internet Comput.* 2011, pp. 135–141 (cit. on pp. 23, 25).

[66] Jemal Abawajy. "Determining service trustworthiness in intercloud computing environments". In: *Proc. 10th Int. Symp. Pervasive Syst., Algorithms, and Networks.* 2009, pp. 784–788 (cit. on pp. 23, 25).

[67] Canh Ngo, Yuri Demchenko, and Cees de Laat. "Toward a Dynamic Trust Establishment approach for multi-provider Intercloud environment." In: *Proc. 4th Int. Conf. Cloud Comput. Technol. and Sci.* 2012, pp. 532–538 (cit. on pp. 24, 25).

[68] Talal H. Noor, Quan Z. Sheng, Sherali Zeadally, and Jian Yu. "Trust Management of Services in Cloud Environments: Obstacles and Solutions". In: *ACM Comput. Surv.* 46.1 (July 2013), 12:1–12:30 (cit. on pp. 24, 39).

[69] Michael R. Clark, Kyle E. Stewart, and Kenneth M. Hopkinson. "Dynamic, Privacy-Preserving Decentralized Reputation Systems". In: *IEEE Trans. Mob. Comput.* 16.9 (2017), pp. 2506–2517 (cit. on pp. 25, 26, 76).

[70] Alexander Schaub, Rémi Bazin, Omar Hasan, and Lionel Brunie. "A trustless privacy-preserving reputation system". In: *IACR Cryptology ePrint Archive* 2016 (2016), p. 16 (cit. on p. 25).

[71] Ginés Dólera Tormo, Félix Gómez Mármol, and Gregorio Martínez Pérez. "Towards privacy-preserving reputation management for hybrid broadcast broadband applications". In: *Computers & Security* 49 (2015), pp. 220–238 (cit. on p. 25).

[72] Osman Hasan, Lionel Brunie, Elisa Bertino, and Ning Shang. "A decentralized privacy preserving reputation protocol for the malicious adversarial model". In: *IEEE Transactions on Information Forensics and Security* 8.6 (2013), pp. 949–962 (cit. on pp. 25, 26, 39, 69, 76).

[73] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. "Order-Preserving Encryption for Numeric Data". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004.* New York, NY, USA: ACM, 2004, pp. 563–574 (cit. on pp. 27, 80).

[74] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. "Order-Preserving Symmetric Encryption". In: *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings.* Berlin, Heidelberg: Springer, 2009, pp. 224–241 (cit. on pp. 27, 32, 80, 81, 83, 101, 103–105, 107).

[75] Raluca A. Popa, Frank H. Li, and Nickolai Zeldovich. "An Ideal-Security Protocol for Order-Preserving Encoding". In: *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013.* IEEE, 2013, pp. 463–477 (cit. on pp. 27, 80).

[76] Florian Kerschbaum and Axel Schröpfer. "Optimal Average-Complexity Ideal-Security Order-Preserving Encryption". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014.* New York, NY, USA: ACM, 2014, pp. 275–286 (cit. on pp. 27, 80).

[77] Dan Boneh, Kevin Lewi, Mariana Raykova, et al. "Semantically Secure Order-Revealing Encryption: Multi-input Functional Encryption Without Obfuscation". In: *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II.* Berlin, Heidelberg: Springer, 2015, pp. 563–594 (cit. on pp. 27, 80).

[78] Kevin Lewi and David J. Wu. "Order-Revealing Encryption: New Constructions, Applications, and Lower Bounds". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016.* New York, NY, USA: ACM, 2016, pp. 1167–1178 (cit. on pp. 27, 28, 32, 80, 81, 101, 102, 104, 105, 107).

[79] Rui Li, Alex X. Liu, Ann L. Wang, and Bezawada Bruhadeshwar. "Fast and Scalable Range Query Processing With Strong Privacy Protection for Cloud Computing". In: *IEEE/ACM Trans. Netw.* 24.4 (2016), pp. 2305–2318 (cit. on pp. 27, 30, 32, 80, 83, 102, 103, 109).

[80] Ioannis Demertzis, Stavros Papadopoulos, Odysseas Papapetrou, Antonios Deligiannakis, and Minos N. Garofalakis. "Practical Private Range Search Revisited". In: *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016.* New York, NY, USA: ACM, 2016, pp. 185–198 (cit. on pp. 27, 31, 32, 80, 83, 102).

[81] Muhammad Naveed, Seny Kamara, and Charles V. Wright. "Inference Attacks on Property-Preserving Encrypted Databases". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015.* New York, NY, USA: ACM, 2015, pp. 644–655 (cit. on p. 28).

[82] Xingliang Yuan, Yu Guo, Xinyu Wang, et al. “EncKV: An Encrypted Key-value Store with Rich Queries”. In: *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*. New York, NY, USA: ACM, 2017, pp. 423–435 (cit. on p. 29).

[83] Meng Shen, Baoli Ma, Liehuang Zhu, et al. “Cloud-Based Approximate Constrained Shortest Distance Queries Over Encrypted Graphs With Privacy Protection”. In: *IEEE Trans. Information Forensics and Security* 13.4 (2018), pp. 940–953 (cit. on pp. 29, 79).

[84] Jun Furukawa. “Request-Based Comparable Encryption”. In: *Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings*. Berlin, Heidelberg: Springer, 2013, pp. 129–146 (cit. on pp. 29, 32).

[85] Jun Furukawa. “Short Comparable Encryption”. In: *Cryptology and Network Security - 13th International Conference, CANS 2014, Heraklion, Crete, Greece, October 22-24, 2014. Proceedings*. Cham: Springer, 2014, pp. 337–352 (cit. on p. 29).

[86] Caleb Horst, Ryo Kikuchi, and Keita Xagawa. “Cryptanalysis of Comparable Encryption in SIGMOD’16”. In: *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*. New York, NY, USA: ACM, 2017, pp. 1069–1084 (cit. on p. 30).

[87] Jun Li and Edward Omiecinski. “Efficiency and Security Trade-Off in Supporting Range Queries on Encrypted Databases”. In: *Data and Applications Security XIX, 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, Storrs, CT, USA, August 7-10, 2005, Proceedings*. Berlin, Heidelberg: Springer, 2005, pp. 69–83 (cit. on pp. 30, 31, 83).

[88] Burton H. Bloom. “Space/Time Trade-offs in Hash Coding with Allowable Errors”. In: *Commun. ACM* 13.7 (1970), pp. 422–426 (cit. on p. 31).

[89] Eu-Jin Goh. “Secure Indexes”. In: *IACR Cryptology ePrint Archive* 2003 (2003), p. 216 (cit. on pp. 31, 83).

[90] Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. “Searchable symmetric encryption: improved definitions and efficient constructions”. In: *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, October 30 - November 3, 2006*. New York, NY, USA: ACM, 2006, pp. 79–88 (cit. on pp. 31, 83, 86).

[91] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. “Searchable symmetric encryption: improved definitions and efficient constructions”. In: *Journal of Computer Security* 19.5 (2011), pp. 895–934 (cit. on pp. 33, 122).

[92] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. “Access Pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation.” In: *NDSS*. Vol. 20. 2012, p. 12 (cit. on pp. 34, 35, 38, 121, 124, 125, 129).

[93] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. “Leakage-abuse attacks against searchable encryption”. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2015, pp. 668–679 (cit. on pp. 34, 36, 37, 124).

[94] *Enron Email Dataset*. Last accessed 20 December 2017, `http://www.cs.cmu.edu/~./enron/` (cit. on pp. 36, 121, 134).

[95] Oded Goldreich and Rafail Ostrovsky. “Software protection and simulation on oblivious RAMs”. In: *Journal of the ACM (JACM)* 43.3 (1996), pp. 431–473 (cit. on p. 36).

[96] Muhammad Naveed. “The Fallacy of Composition of Oblivious RAM and Searchable Encryption.” In: *IACR Cryptology ePrint Archive* 2015 (2015), p. 668 (cit. on p. 36).

[97] Peter Williams and Radu Sion. “Single round access privacy on outsourced storage”. In: *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM. 2012, pp. 293–304 (cit. on p. 36).

[98] Emil Stefanov, Marten Van Dijk, Elaine Shi, et al. “Path ORAM: an extremely simple oblivious RAM protocol”. In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM. 2013, pp. 299–310 (cit. on p. 37).

[99] Jun Tang, Yong Cui, Qi Li, et al. “Ensuring security and privacy preservation for cloud data services”. In: *ACM Computing Surveys (CSUR)* 49.1 (2016), p. 13 (cit. on p. 37).

[100] Valentina Ciriani, Sabrina De Capitani Di Vimercati, Sara Foresti, et al. “Fragmentation and encryption to enforce privacy in data storage”. In: *European Symposium on Research in Computer Security*. Springer. 2007, pp. 171–186 (cit. on p. 37).

[101] Valentina Ciriani, Sabrina De Capitani Di Vimercati, Sara Foresti, et al. “Combining fragmentation and encryption to protect privacy in data storage”. In: *ACM Transactions on Information and System Security (TISSEC)* 13.3 (2010), p. 22 (cit. on p. 37).

[102] Pierangela Samarati. “Data Security and Privacy in the Cloud.” In: *ISPEC*. 2014, pp. 28–41 (cit. on pp. 37, 122).

[103] Valentina Ciriani, Sabrina De Capitani Di Vimercati, Sara Foresti, et al. “Keep a Few: Outsourcing Data While Maintaining Confidentiality.” In: *ESORICS*. Vol. 9. Springer. 2009, pp. 440–455 (cit. on p. 37).

[104] Mohamed Ahmed Abdelraheem, Tobias Andersson, and Christian Gehrmann. “Searchable Encrypted Relational Databases: Risks and Countermeasures”. In: *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 2017, pp. 70–85 (cit. on p. 37).

[105] Tatsuaki Okamoto. “Efficient Blind and Partially Blind Signatures Without Random Oracles”. In: *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*. 2006, pp. 80–99 (cit. on p. 44).

[106] Adi Shamir. “How to share a secret”. In: *Communications of the ACM* 22.11 (1979), pp. 612–613 (cit. on pp. 52, 55).

[107] Josh Cohen Benaloh. “Secret sharing homomorphisms: Keeping shares of a secret secret”. In: *Conference on the Theory and Application of Cryptographic Techniques*. Springer. 1986, pp. 251–260 (cit. on p. 52).

[108] Yehuda Lindell. “How To Simulate It - A Tutorial on the Simulation Proof Technique”. In: *IACR Cryptology ePrint Archive* 2016 (2016), p. 46 (cit. on p. 61).

[109] M. Ho Au, P. P. Tsang, and A. Kapadia. “PEREA: Practical TTP-free Revocation of Repeatedly Misbehaving Anonymous Users”. In: *ACM Trans. Inf. Syst. Security* 14.4 (2008), 29:1–29:34 (cit. on p. 61).

[110] Qian Wang, Kui Ren, Minxin Du, Qi Li, and Aziz Mohaisen. “SecGDB: Graph Encryption for Exact Shortest Distance Queries with Efficient Updates”. In: *Financial Cryptography and Data Security - 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers*. Cham: Springer, 2017, pp. 79–97 (cit. on p. 79).

[111] Dan Boneh and Brent Waters. “Conjunctive, Subset, and Range Queries on Encrypted Data”. In: *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*. Berlin, Heidelberg: Springer, 2007, pp. 535–554 (cit. on p. 79).

[112] K. Gai and M. Qiu. “Blend Arithmetic Operations on Tensor-based Fully Homomorphic Encryption Over Real Numbers”. In: *IEEE Trans. Industrial Informatics* PP.99 (2017), pp. 1–1 (cit. on p. 80).

[113] Hsiao-Ying Lin and Wen-Guey Tzeng. “An Efficient Solution to the Millionaires’ Problem Based on Homomorphic Encryption”. In: *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings*. Berlin, Heidelberg: Springer, 2005, pp. 456–466 (cit. on pp. 90, 91).

[114] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. "Efficient Private Matching and Set Intersection". In: *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*. Berlin, Heidelberg: Springer, 2004, pp. 1–19 (cit. on p. 94).

[115] Kevin Lewi. *Fastore-An Implementation of Order-Revealing Encryption*. Last Accessed: May, 2018, `https://github.com/kevinlewi/fastore`. 2016 (cit. on p. 102).

[116] Torbjörn Granlund and the GMP development team. *GNU MP: The GNU Multiple Precision Arithmetic Library*. Last Accessed: May, 2018, `http://gmplib.org`. 2012 (cit. on p. 102).

[117] Raluca A. Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. "CryptDB: protecting confidentiality with encrypted query processing". In: *Proceedings of the 23rd ACM Symposium on Operating Systems Principles 2011, SOSP 2011, Cascais, Portugal, October 23-26, 2011*. New York, NY, USA: ACM, 2011, pp. 85–100 (cit. on p. 103).

[118] CryptDB Group. *CryptDB-A database system that can process SQL queries over encrypted data*. Last accessed 20 December 2017,`https://github.com/CryptDB/cryptdb`. 2014 (cit. on p. 103).

[119] Jure Leskovec and Andrej Krevl. *SNAP Datasets: Stanford Large Network Dataset Collection (Gowalla)*. Last accessed 20 December 2017, `https://snap.stanford.edu/data/loc-gowalla.html`. 2014 (cit. on p. 103).

[120] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines (usps)*. Last accessed 20 December 2017,`https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html#usps`. 2011 (cit. on p. 103).

[121] Mohamed Ahmed Abdelraheem, Tobias Andersson, and Christian Gehrmann. "Inference and Record-Injection Attacks on Searchable Encrypted Relational Databases". In: *Cryptology ePrint Archive, Report 2017/024* (2017) (cit. on p. 124).

[122] Lei Jiao, Jun Lit, Wei Du, and Xiaoming Fu. "Multi-objective data placement for multi-cloud socially aware services". In: *INFOCOM, 2014 Proceedings IEEE*. IEEE. 2014, pp. 28–36 (cit. on p. 132).

[123] *gco-v3.0 library*. Last accessed 20 December 2017, `http://vision.csd.uwo.ca/code/gco-v3.0.zip` (cit. on p. 135).

[124] *MongoDB Sharding*. Last accessed 20 December 2017, `https://docs.mongodb.com/manual/sharding/` (cit. on p. 135).

[125] Theodoros Rekatsinas, Amol Deshpande, and Ashwin Machanavajjhala. "SPARSI: partitioning sensitive data amongst multiple adversaries". In: *Proceedings of the VLDB Endowment* 6.13 (2013), pp. 1594–1605 (cit. on p. 137).

[126] Jordi Vilaplana, Francesc Solsona, Ivan Teixidó, et al. "A queuing theory model for cloud computing". In: *The Journal of Supercomputing* 69.1 (2014), pp. 492–507 (cit. on p. 139).

[127] Hamzeh Khazaei, Jelena Misic, and Vojislav B Misic. "Performance analysis of cloud computing centers using m/g/m/m+ r queuing systems". In: *IEEE Transactions on parallel and distributed systems* 23.5 (2012), pp. 936–943 (cit. on p. 139).

[128] Siva Theja Maguluri, R Srikant, and Lei Ying. "Stochastic models of load balancing and scheduling in cloud computing clusters". In: *INFOCOM, 2012 Proceedings IEEE*. IEEE. 2012, pp. 702–710.