SECURE PROTOCOLS FOR PRIVACY-PRESERVING DATA OUTSOURCING, INTEGRATION, AND AUDITING

Gaby Dagher

A thesis

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy

Concordia University

Montréal, Québec, Canada

December 2015

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By:	**Mr. Gaby Dagher**
Entitled:	**Secure Protocols for Privacy-preserving Data Outsourcing, Integration, and Auditing**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

______________________	Chair
Prof. Stan Matwin ______________________	External Examiner
Prof. Todd Eavis ______________________	Examiner
Prof. Wahab Hamou-Lhadj ______________________	Examiner
Prof. Lingyu Wang ______________________	Examiner
Prof. Benjamin Fung ______________________	Supervisor
Prof. Jeremy Clark ______________________	Supervisor

Approved by:

____

Chair of Department or Graduate Program Director

______________ 20 ________ ______________________________

Amir Asif, Ph.D., Dean

Faculty of Engineering and Computer Science

Abstract

Secure Protocols for Privacy-preserving Data Outsourcing, Integration, and Auditing

Gaby Dagher, Ph.D.

Concordia University, 2016

As the amount of data available from a wide range of domains has increased tremendously in recent years, the demand for data sharing and integration has also risen. The cloud computing paradigm provides great flexibility to data owners with respect to computation and storage capabilities, which makes it a suitable platform for them to share their data. Outsourcing person-specific data to the cloud, however, imposes serious concerns about the confidentiality of the outsourced data, the privacy of the individuals referenced in the data, as well as the confidentiality of the queries processed over the data. Data integration is another form of data sharing, where data owners jointly perform the integration process, and the resulting dataset is shared between them. Integrating related data from different sources enables individuals, businesses, organizations and government agencies to perform better data analysis, make better informed decisions, and provide better services. Designing distributed, secure, and privacy-preserving protocols for integrating person-specific data, however, poses several challenges, including how to prevent each party from inferring sensitive information about individuals during the execution of the protocol, how to guarantee an effective level of privacy on the released data while maintaining utility for data mining, and how to support public auditing such that anyone at any time can verify that the integration was executed correctly and no participants deviated from the protocol. In this thesis, we address the aforementioned concerns by presenting secure protocols for privacy-preserving data outsourcing, integration and auditing. First, we propose a secure cloud-based data outsourcing and query processing framework that simultaneously preserves the confidentiality of the data and the query requests, while providing differential privacy guarantees on the query results. Second, we propose a publicly verifiable protocol for integrating person-specific data from multiple data owners, while providing differential privacy guarantees and maintaining an effective level of utility on the released data for the purpose of data mining. Next, we propose a privacy-preserving multi-party protocol for high-dimensional data mashup with guaranteed *LKC*-privacy on the output data. Finally, we apply the theory to the real world problem of solvency in Bitcoin. More specifically, we propose a privacy-preserving and publicly verifiable cryptographic proof of solvency scheme for Bitcoin exchanges such that no information is revealed about the exchange's customer holdings, the value of the exchange's total holdings is kept secret, and multiple exchanges performing the same proof of solvency can contemporaneously prove they are not colluding.

Dedication

To my wife *Baha* and my children *George* and *Gabriel*,

with love.

"You can't connect the dots looking forward; you can only connect them looking backwards. So you have to trust that the dots will somehow connect in your future."

- Steve Jobs

Acknowledgments

I would like to express my sincere gratitude to my supervisors, Prof. Benjamin C. M. Fung and Prof. Jeremy Clark, for their resolute recommendations, guidance and assistance from the earliest to the final stages of my research. They have been a great source of encouragement throughout my Ph.D. study, and I am profoundly grateful to them.

Furthermore, I am endlessly grateful to all my family members for their unwavering support and motivation throughout this entire process.

Contents

List of Figures **xiv**

List of Tables **xvi**

1 Introduction **1**

1.1 Motivation 1

1.2 Contributions 3

1.2.1 SecDM: Secure and Privacy-preserving Data Outsourcing 3

1.2.2 SecSVD: Secure and Privacy-preserving Set-Valued Data Data Integration with Public Verifiability 4

1.2.3 Fusion: Secure and Privacy-preserving Relational Data Integration 5

1.2.4 Provisions: Secure and Privacy-preserving Data Auditing in Bitcoin 6

1.3 Organization of the Thesis 6

2 Background **8**

2.1 Privacy Models 8

2.1.1 Differential Privacy 9

2.1.2 LKC-Privacy 10

2.2 Cryptographic Primitives 10

2.2.1 Verifiable Mix Network 10

2.2.2 Exponential ElGamal 11

2.2.3 Distributed Exponential ElGamal Decryption 11
2.2.4 Mix and Match 12
2.2.5 Bulletin Board 12
2.2.6 Multiparty Coin Tossing 13
2.2.7 Non-Interactive Zero-Knowledge Proofs (NIZKP) 13
2.2.8 Cut-and-Choose 13
2.2.9 Pedersen Commitments 14

3 Literature Review **15**

3.1 Privacy-Preserving Data Publishing 15
3.1.1 Overview 15
3.1.2 Non-Interactive PPDP on Set-Valued Data 17
3.1.3 Interactive PPDP on Set-Valued Data 18
3.2 Confidentiality in Data Outsourcing 19
3.2.1 Centralized Environment 19
3.2.2 Distributed Environment 21
3.3 Searching on Encrypted Data 22
3.3.1 Searchable Encryption 22
3.3.2 Functional Encryption 26
3.3.3 Secure Function Evaluation 27
3.4 Public Verifiability 28

4 Secure and Privacy-preserving Data Outsourcing **29**

4.1 Introduction 29
4.2 Preliminaries 33
4.2.1 Anonymous Ciphertext-Policy Attribute Based Encryption 35
4.2.2 DiffGen Algorithm 36
4.3 Problem Formulation 37

4.3.1 Problem Overview 38

4.3.2 System Inputs 38

4.3.3 Adversarial Model 43

4.3.4 Problem Statement 44

4.4 Solution: SecDM Framework 44

4.4.1 Solution Overview 44

4.4.2 Secure Index Construction 46

4.4.3 Confidential Query Processing 51

4.4.4 Discussion 59

4.5 Protocol Analysis 61

4.5.1 Complexity Analysis 61

4.5.2 Correctness Analysis 61

4.5.3 Security Analysis 64

4.6 Performance Evaluation 71

4.6.1 Implementation and Setup 71

4.6.2 Experimental Results 72

4.7 Summary 76

5 Secure and Privacy-preserving Set-Valued Data Integration with Public Verifiability 78

5.1 Introduction 78

5.2 Preliminaries 84

5.2.1 Publishing Set-Valued Data via Differential Privacy 84

5.2.2 Encryption Scheme 85

5.2.3 Mix and Match Protocol 86

5.3 Solution: SecSVD Protocol 86

5.3.1 Solution Overview 86

5.3.2 Adversarial Model . . . 87
5.3.3 Data Model . . . 88
5.3.4 Auxiliary Functions . . . 88
5.3.5 Data Preparation . . . 89
5.3.6 Secure Distributed Differentially-private Set-valued Data Release Protocol . . 93
5.4 Protocol Analysis . . . 99
5.4.1 Complexity Analysis . . . 99
5.4.2 Security Analysis . . . 100
5.4.3 Correctness and Utility Analysis . . . 102
5.5 Performance Evaluation . . . 103
5.5.1 Experimental Results . . . 103
5.5.2 Cost Estimation . . . 104
5.6 Summary . . . 106

6 Secure and Privacy-preserving Relational Data Integration 108

6.1 Introduction . . . 108
6.2 Preliminaries . . . 111
6.2.1 Privacy Model . . . 111
6.2.2 Encryption Scheme . . . 112
6.2.3 Taxonomy Tree . . . 112
6.3 Problem Formulation . . . 113
6.3.1 Problem Overview . . . 113
6.3.2 Data Utility Measures . . . 114
6.3.3 Adversarial Model . . . 116
6.3.4 Problem Statement . . . 116
6.4 Solution: Fusion Protocol . . . 116
6.4.1 Solution Overview . . . 116

6.4.2 Multi-Party Protocol for Computing Specialization Score . . . 118
6.4.3 Multi-Party Protocol for LKC-private Data Integration Release . . . 122
6.5 Performance Evaluation . . . 124
6.5.1 Implementation and Setup . . . 125
6.5.2 Mashup Utility . . . 126
6.5.3 Scalability . . . 126
6.6 Summary . . . 127

7 Secure and Privacy-preserving Data Auditing in Bitcoin 128

7.1 Introduction . . . 128
7.2 Background . . . 131
7.2.1 Exchange Structure and Holdings . . . 132
7.2.2 Maxwell's Proof of Liabilities . . . 134
7.2.3 Proof of Assets . . . 135
7.3 Preliminaries . . . 137
7.3.1 Public Parameters . . . 137
7.3.2 Bitcoin Balance Lookups . . . 137
7.3.3 Commitments . . . 137
7.4 Solution: Provisions Protocol . . . 138
7.4.1 Protocol Overview . . . 138
7.4.2 Proof of Assets . . . 139
7.4.3 Proof of Liabilities . . . 142
7.4.4 Customer Verification . . . 145
7.4.5 Proof of Solvency . . . 145
7.4.6 Proof of Non-Collusion . . . 147
7.5 Security Discussion . . . 149
7.5.1 Security Definition . . . 149

7.5.2 Anonymity Sets 150

7.5.3 User Verification 152

7.6 Performance Evaluation 153

7.6.1 Asymptotic Performance 153

7.6.2 Incremental Updates 154

7.6.3 Implementation and Setup 155

7.6.4 Experimental Results 157

7.7 Summary 157

8 Conclusion **159**

8.1 Summary 159

8.2 Looking Ahead 160

Bibliography **162**

List of Figures

1 Security requirements for data outsourcing. 31

2 A raw data table D and its taxonomy trees. 39

3 Algorithm *DiffGen* for generating ε-differentially private data. 40

4 Taxonomy tree $\mathbb{T}^{Job}$ for attribute *Job*. 41

5 SecDM: privacy-preserving framework for data outsourcing. 45

a The setup phase. 45

b The query processing phase. 45

6 ADT query overlap. 55

7 Example of access structure and system count query. 57

8 C-SecDM: centralized SecDM framework. 59

9 Performance comparison w.r.t. # of queries. 60

10 Scalability of framework construction w.r.t. number of records. 73

11 Scalability of query processing. 74

a Number of specializations = 8. 74

b Number of specializations = 10. 74

c Number of specializations = 12. 74

12 Efficiency w.r.t. number of attributes per a query. 76

a Exact query 76

b Specific query 76

c Generic query 76

13 A context-free taxonomy tree for item universe $\{I_1, I_2, I_3, I_4\}$. 84

14 An encrypted context-free taxonomy tree TaxTree. 90

15 PartTree: ε-differentially private partitioning tree of MixDS. 95

16 Using Mix and Match protocol for record assignment. 98

17 Utility loss for frequent itemset mining. 104

a MSNBC dataset 104

b STM dataset 104

18 Privacy-preserving distributed data integration. 109

19 Taxonomy trees: $\mathbb{T}^{Age}$, $\mathbb{T}^{Education}$, $\mathbb{T}^{Sex}$ and $\mathbb{T}^{Salary}$. 112

20 Hierarchal high-dimensional data integration (HHDI). 123

21 Utility of mashup w.r.t. anonymity threshold K. 125

22 Scalability of Fusion protocol. 126

a Scalability w.r.t. number of attributes. 126

b Scalability w.r.t. number of records. 126

23 The Merkle tree from the Maxwell protocol [Wil14] for proof of solvency. 133

24 Performance of Protocol 7.1 (proof of assets). 156

25 Performance of Protocol 7.2 (proof of liabilities). 156

List of Tables

1 Summary of the thesis contributions . . . 3

2 Comparative evaluation of main features in related query processing approaches . . . 34

3 Differentially-private data table $\hat{D}$ . . . 40

4 Transformed data table $\hat{D}$ . . . 41

5 Key accessibility w.r.t. all parties in SecDM framework . . . 71

6 Sample set-valued datasets . . . 81

a Dataset owned by Alice: DS_1 . . . 81

b Dataset owned by Bob: DS_2 . . . 81

c Dataset owned by Carol: DS_3 . . . 81

7 Comparative evaluation of main features in related data integration approaches . . . 83

8 Integrated dataset of the sample datasets in Table 6 . . . 94

9 The utilized publicly verifiable primitives . . . 101

10 Average cost of PET on Mix and Match truth tables . . . 105

11 Estimated performance of Protocol 2.1 . . . 106

12 Estimated scalability of Protocol 2.1 . . . 106

13 Raw data owned by providers $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ . . . 110

14 LKC anonymous mashup data . . . 114

15 Data owned by $\mathcal{P}_1$ and $\mathcal{P}_3$ after receiving $[\![Class]\!]$ from $\mathcal{P}_2$ . . . 120

16 The distribution of attributes among parties in each multiparty setting . . . 124

Chapter 1

Introduction

1.1 Motivation

Sharing and integrating interrelated data from different sources has become critical in many contexts, such as sharing data between scientific researchers, identifying fraudulent insurance claims, epidemic monitoring and forecasting, and fighting terrorism. Data sharing and integration enables individuals, businesses, organizations and government agencies to perform better data analysis, make better informed decisions, and provide better services. In a recent report [1] that analyzes the services of the health and human services agencies, it is estimated that the U.S. government loses $342 billion every year in improper payments due to the lack of integration of patient information with other key data such as quality and cost.

One common approach for data sharing is data outsourcing to the cloud. The cloud computing paradigm is a new computing platform that enables data owners to have access to large-scale computation and storage at an affordable price. Data-as-a-service (DaaS) is one of the cloud computing services that facilitates the hosting and managing of large-scale databases on the cloud. DaaS is a

[1]http://www.businesswire.com/news/home/20150817005163/en/Health-Human-Services-Agencies-Squander-342B-Annually

compelling service for data owners, as they no longer need to invest in hardware, software and operational overhead. Despite all these benefits, data owners are reluctant to adopt the DaaS model, as it requires outsourcing their person-specific data to an untrusted cloud service provider, which raises several major concerns. The first concern is how to protect the confidentiality of the data while being stored on the cloud. More specifically, how to prevent the cloud service provider (and other tenants who share the cloud resources, services and physical infrastructure) from gaining access to the raw data. The second concern is how to enable the cloud to process queries on the stored data while revealing nothing about the queries to the cloud. The third concern is how to protect the privacy of the individuals whose personal information is stored in the outsourced data, and ensure that the query results cannot be used to either identify an individual or to reveal sensitive information about him. Typically, the data is de-identified before it is uploaded to the cloud such that direct identifying information about the individuals has been removed. However, the de-identification the data does not prevent record/attribute linkage attacks, as was shown in the cases of AOL [BZ06] and Netflix [NS08]. Hence the strong demands for advanced privacy protection techniques, such as *privacy-preserving data publishing (PPDP)*, a mechanism for publishing person-specific data such that useful information can be obtained from the published data while preserving the privacy of individuals.

On the other hand, integrating person-specific data in a distributed environment for the purpose of sharing the resulting dataset raises several concerns, some of which are analogous to the those in data outsourcing. The first concern is about protecting the privacy of the individual in the data being integrated such that the resulting integrated data cannot be used by any party to learn sensitive information about individuals. The second concern is how to make sure the integration process is secure and privacy-preserving. That is, how to ensure that all data owners adhere to their integration instructions while not being able to learn any useful information about other parties' data. The third concern is how to perform the integration such that the integrated data maintains certain level of useful information for data mining and analysis.

Table 1: Summary of the thesis contributions

Protocols	Data Sharing Model		Adversarial Model		Data Type		Privacy Model	
	Data Outsourcing	Data Integration	Semi-Honest	Malicious with Public Verifiability	Relational Data	Set-Valued Data	Differential Privacy	*LKC* Privacy
SecDM (Chapter 4)	✓		✓		✓		✓	
SecSVD (Chapter 5)		✓		✓		✓	✓	
Fusion (Chapter 6)		✓	✓		✓			✓
Provisions (Chapter 7)				✓	✓			

In the context of data sharing and integration, especially when dealing with person-specific data, it is often required to have an auditing mechanism in place to verify the correct execution of the data sharing and integration protocol. For example, the U.S. Health Information Technology for Economic and Clinical Health Act (HITECH) requires the Department of Health & Human Services (HHS) to periodically audit covered entities and business associates to ensure their compliance with the HIPAA Privacy, Security, and Breach Notification Rules [2]. During the execution of a protocol in untrusted environment, participants must prove they are following the protocol at each step. This auditing mechanism is called *private verifiability*, where the produced proofs may only convince the other participants in the protocol by being contingent on their interactions and/or individual secrets. On the other hand, *public verifiability*, a higher standard than private verifiability, enables *any internal or external party* at *any time* to verify that the protocol was executed correctly.

1.2 Contributions

To address the aforementioned challenges related to person-specific data sharing and integration, we study the problem of privacy-preserving data outsourcing, integration, and auditing and propose protocols for achieving that securely. Table 1 summarises different characteristics of the proposed protocols. The key contributions of this thesis are summarized as follows.

1.2.1 SecDM: Secure and Privacy-preserving Data Outsourcing

Recall that Data-as-a-service (DaaS) is a cloud computing service that emerged as a viable option to businesses and individuals for outsourcing and sharing their collected data with other parties.

[2]http://www.hhs.gov/ocr/privacy/hipaa/enforcement/audit/

Although the cloud computing paradigm provides great flexibility to consumers with respect to computation and storage capabilities, it imposes serious concerns about the confidentiality of the outsourced data as well as the privacy of the individuals referenced in the data.

In Chapter 4 of this thesis, we formulate and address the problem of querying encrypted data in a cloud environment such that query processing is confidential and the result is differentially private. Unlike other work in the literature [BAAD14][GLM$^+$13a][WAEA11][Yon14][TH13] for data outsourcing, our solution maintains the privacy and utility properties of the outsourced data while simultaneously ensuring *data confidentiality*, *query confidentiality*, and *privacy-preserving results*. More specifically, we propose a framework where the data provider uploads an encrypted index of her anonymized data to a DaaS service provider that is responsible for answering range count queries from authorized data miners for the purpose of data mining. To satisfy the confidentiality requirements, we leverage attribute based encryption to construct a secure *k*d-tree index over the differentially private data for fast access. We also utilize the exponential variant of the ElGamal cryptosystem to efficiently perform homomorphic operations on encrypted data. Experiments on real-life data demonstrate that our proposed framework can efficiently answer range queries and is scalable with increasing data size. Note that while comminution and network latency costs should be taken in accounts for practicality [DCFG$^+$14], we focus in this thesis on computational costs when generating our experimental results.

1.2.2 SecSVD: Secure and Privacy-preserving Set-Valued Data Data Integration with Public Verifiability

Privacy-preserving data integration is a mechanism that enables multiple data owners to securely integrate their data for the purpose of data mining. Applying such a mechanism on set-valued data in a malicious environment, however, involves several challenges, including how to handle the high-dimensional nature of set-valued data, how to prevent malicious parties from inferring sensitive information during the integration process, how to guarantee an effective level of privacy on the released data while maintaining utility, and how to enable independent public verifiability of the protocol.

In Chapter 5 of this thesis, we propose the first publicly verifiable protocol for integrating person-specific set-valued data from two or multiple data owners, while providing ε-differential privacy guarantee and maintaining an effective level of utility on the released data. Our proposed approach can handle both horizontally and vertically partitioned data, and is secure in the malicious adversarial model with dishonest majority.

1.2.3 Fusion: Secure and Privacy-preserving Relational Data Integration

In the last decade, several approaches concerning private data release for data mining have been proposed. Data integration, on the other hand, is a mechanism for merging data from several data providers. Fusing both techniques to generate mashup (integrated) data in a distributed environment while providing privacy and utility guarantees on the output involves several challenges. That is, how to ensure that no unnecessary information is leaked to the other parties during the integration process, how to ensure the mashup data is protected against certain privacy threats, and how to handle the high-dimensional nature of the data while guaranteeing high data utility.

In Chapter 6 of this thesis, we propose Fusion, a privacy-preserving multi-party protocol for high-dimensional data integration with guaranteed LKC-privacy for the purpose of data mining. Our contribution is twofold. First, we design a secure and distributed protocol for evaluating an information gain-based score function. Second, we propose a hieratical approach for high-dimensional data integration where the resulting dataset is LKC-private. Experiments on real-life data demonstrate that the anonymous mashup data provide better data utility, the approach can handle high dimensional data, and it is scalable with respect to the data size.

1.2.4 Provisions: Secure and Privacy-preserving Data Auditing in Bitcoin

Bitcoin exchanges function like banks, securely holding their customers' bitcoins on their behalf. Several exchanges have suffered catastrophic losses with customers permanently losing their savings. A proof of solvency demonstrates that the exchange controls sufficient reserves to settle each customer's account.

In Chapter 7 of this thesis, we propose Provisions, a privacy-preserving proof of solvency whereby an exchange does not have to disclose its Bitcoin addresses; total holdings or liabilities; or any information about its customers. Provisions is the first protocol to enable Bitcoin exchanges to provide a *proof of reserves* with a corresponding *proof of liabilities* and prove their solvency in a complete privacy-preserving manner. We also propose an extension which prevents exchanges from colluding to cover for each other's losses. We have implemented Provisions and it offers practical computation times and proof sizes even for a large Bitcoin exchange with millions of customers.

1.3 Organization of the Thesis

The rest of this thesis is organized as follows:

- Chapter 2 introduces two recent privacy models, and several cryptographic primitives involving public encryption schemes with homomorphic properties, commitments, verifiable mixing, and zero knowledge proofs.

- Chapter 3 provides an in-depth literature review of the state-of-the-art techniques for privacy-preserving data publishing, confidentiality in data outsourcing, searching on encrypted data, and interactive and non-interactive privacy-preserving data mining.

- Chapter 4 studies the problem of secure and privacy-preserving data outsourcing. We propose a secure cloud-based data outsourcing and query processing framework that simultaneously preserves the confidentiality of the data and the query requests, while providing differential privacy guarantee on query outputs. The results of this chapter are currently under review in [DFMC].

- Chapter 5 studies the problem of secure and privacy-preserving set-valued data integration with public verifiability. We propose a publicly verifiable protocol for integrating person-specific data from multiple data owners, while providing differential privacy guarantee and maintaining an effective level of utility on the released data for the purpose of data mining. The results of this chapter are currently under review in [DCF].

- Chapter 6 studies the problem of secure and privacy-preserving relational data mashup. We propose a multi-party protocol for high-dimensional data mashup with guaranteed *LKC*-privacy on the output data. The results of this chapter have been published in [DIAF15][ADFH14].

- Chapter 7 studies the problem of secure and privacy-preserving data auditing in Bitcoin. We propose a cryptographic proof of solvency scheme for Bitcoin exchanges such that no information is revealed about the exchanges customer holdings, the value of the exchanges total holdings is kept secret, and multiple exchanges performing the same proof of solvency can contemporaneously prove they are not colluding. The work in this chapter is a collaborative effort with a team from the computer science department at Stanford University. The results of this chapter have been published in [DBB^{+}15b].

- Chapter 8 concludes the thesis and provides a look ahead.

Chapter 2

Background

In this chapter, we present some background for the research presented in this thesis. We first introduce the data privacy models in Section 2.1, and then introduce the cryptographic primitives in Section 2.2.

2.1 Privacy Models

In [Dal77], Dalenius provided the following definition for protecting the privacy of individuals in published person-specific data.

Definition 1 ***Privacy Protection [Dal77].*** Access to the published data should not enable the attacker to learn anything extra about any target victim compared to no access to the database, even with the presence of any attacker's background knowledge obtained from other sources. □

As a result, different privacy models have been proposed in the literature, e.g., K-anonymity [Sam01a] [Swe02b], ℓ-diversity [MKGV07], and LKC-Privacy [MFHL09], to provide different levels of privacy protections based on the degree of the background knowledge of the attacker. However, Dwork *et al.* [DMNS06] later proved that with the presence of any attacker's background knowledge, absolute privacy protection cannot be achieved. In this section, we will present two of the main privacy protection models, namely, *differential privacy* [DMNS06] and *LKC-privacy* [MFHL09], where the former makes no assumptions on the attacker's prior knowledge, and the latter assumes a bound on the number of attributes L previously known to the attacker. Note that these privacy models will be later utilized in the remaining of the thesis to provide privacy guarantees on the output data of our proposed protocols.

2.1.1 Differential Privacy [DMNS06]

Differential privacy is a privacy model introduced by Dwork *et al.* for the purpose of preserving data confidentiality without making no assumptions about the attacker's background knowledge. Differential privacy provides a strong guarantee that the presence or the absence of an individual will not significantly affect the final output of any function.

Definition 2 ***ε-Differential Privacy.*** Given any two datasets D_1 and D_2 that differ on at most one record, a sanitizing mechanism M achieves ε-differential privacy if for any output $\hat{D} \in Range(M)$:

$$Pr[M(D_1) = \hat{D}] \leq e^{\varepsilon} \times Pr[M(D_2) = \hat{D}] \tag{1}$$

where the probabilities are taken over the randomness of M. □

Definition 3 ***Global Sensitivity [DMNS06].*** Given a query function $f : D \rightarrow \mathbb{R}^d$ that maps dataset D to a vector of d reals, the global sensitivity of f is:

$$S(f) = \max_{D_1, D_2} ||f(D_1) - f(D_2)||_1 \tag{2}$$

where D_1 and D_2 are any two neighbouring datasets that differ on at most one record. □

To achieve differential privacy when using query function f, the principal approach is to perturb the true output of f by adding to it random noise that is adjusted based on $S(f)$. In [DMNS06], the authors proposed generating the noise according to the Laplace distribution, $Lap(\lambda)$, where the probability distribution function is $Pr(x|\lambda) = \frac{1}{2\lambda} e^{\frac{|x|}{\lambda}}$, the mean is 0, and the standard deviation is λ which is determined based on the global sensitivity $S(f)$ and the privacy parameter ε.

Theorem 2.1.1 *[DMNS06]. For any function $f : D \to \mathbb{R}^d$ that maps datasets to reals, the privacy mechanism M:*

$$M(D) = f(D) + Lap(S(f)/\varepsilon) \tag{3}$$

satisfies ε-differential privacy. □

2.1.2 LKC-Privacy [MFHL09]

Let L be the maximum number of values of *quasi-identifier* attributes QID of the adversary's background knowledge on any individual in a data table T. Let S be a set of sensitive values. A data table T satisfies LKC*-privacy* if, and only if, for any values of quasi-identifier attributes of the adversary's background knowledge on an individual qid with $|qid| \leq L$,

1. $|T[qid]| \geq K$, where $T[qid]$ is the group of records in T containing qid and $K > 0$ is a minimum *anonymity threshold*, and

2. $\forall s \subseteq S$, $P(s|qid) \leq C$, where $P(s|qid)$ is the confidence with which the adversary can infer that the target victim has sensitive value s, and $0 < C \leq 1$ is a maximum *confidence threshold*.

LKC-privacy is a generalized privacy model of K-anonymity [Sam01a][Swe02b], confidence bounding [WFY07], and ℓ-diversity [MKGV07], which gives the flexibility to the data providers to employ these traditional privacy models.

2.2 Cryptographic Primitives

In this section, we introduce the fundamental building blocks for the cryptographic protocols that will be presented this thesis.

2.2.1 Verifiable Mix Network [JJR]

A mix network allows a list of encrypted messages to be jointly shuffled and re-randomized such that no participant knows the permutation mapping the inputs to outputs. A verifiable mix network produces a publicly verifiable transcript proving that the output list is correct (i.e., the messages were only reordered, not modified nor replaced).

2.2.2 Exponential ElGamal [CGS97]

ElGamal [EG85] is a public key encryption scheme based on the hardness of computing discrete logarithms (and a stronger assumption called decisional Diffie-Hellmen). Exponential ElGamal [CGS97] is a variant that is additively homomorphic and suitable for short messages. Exponential ElGamal consists of the following algorithms:

$\mathsf{KeyGen}()$. A 2048-bit safe prime p is chosen randomly such that $p = \alpha \cdot q + 1$ for a 256-bit prime q and some integer α (parameter sizes comply with current NIST recommendations [BR11]). Let g be a generator of the multiplicative subgroup $\mathbb{G}_q$. The private key x is chosen randomly from $\mathbb{Z}_q^*$ and the public key y is computed as $y = g^x \bmod p$ (henceforth, assume all operations are done $\bmod\, p$).

$\mathsf{Enc}(m, y, r)$. To encrypt a short message m with public key y, choose random integer r from $\mathbb{Z}_q^*$ and compute the ciphertext as $c = \langle c_1, c_2 \rangle = \langle g^r, g^m y^r \rangle$.

$\mathsf{Dec}(c, x)$. To decrypt ciphertext c with private key x, first compute $g^m = c_2 \cdot c_1^{-x}$ and solve for m (recall it is short) using a lookup table of pre-computed values or appropriate algorithm (such as Pollard's rho).

Exponential ElGamal is additively homomorphic, meaning a given encrypted message $\mathsf{Enc}(m, y, r) = \langle c_1, c_2 \rangle$ can be added to a second encrypted message $\mathsf{Enc}(m', y, r') = \langle c_1', c_2' \rangle$ without decryption: $\mathsf{Enc}(m + m', y, r + r') = \langle c_1 \cdot c_1', c_2 \cdot c_2' \rangle$. $\mathsf{Enc}(m, y, r)$ can also be multiplied by a constant a homomorphically: $\mathsf{Enc}(am, y, ar) = \langle c_1^a, c_2^a \rangle$.

2.2.3 Distributed Exponential ElGamal Decryption [Bra06]

Given ElGamal ciphertext (α, β), where the secret key $x \in \mathbb{Z}_p$ is shared between n parties, each participant $\mathcal{P}_i$ from $\mathcal{P}_1, \ldots, \mathcal{P}_n$ publishes β^{x_i}, where x_i is a private key share of $\mathcal{P}_i$. The plaintext can then be derived by computing: $\alpha/\prod_{i=1}^{n}\beta^{x_i}$. Note that Distributed Key Generation (DKG) protocol [Ped91] can be utilized to create the shared secret key, where each participant only knows its own share of the secret key.

2.2.4 Mix and Match [JJ00]

Mix and Match is a multi-party protocol for obliviously evaluating an encrypted input against a plaintext lookup table, where all values are encrypted (with an encryption scheme as above). The output is a ciphertext corresponding to the lookup table, and no participant knows the actual looked up value. In the *mix* phase, the participants use a verifiable mix network to blind and perform row-wise permutation of the lookup table in a distributed fashion. In the matching *phase*, the participants determine in a distributed fashion whether two given ciphertexts are equal using a primitive called *plaintext equality test (PET)*. More specifically, given two ElGamal ciphertexts (α_1, β_1) and (α_2, β_2), the participants jointly evaluate whether the underlying plaintexts for (α_1, β_1) and (α_2, β_2) [denoted by $(\alpha_1, \beta_1) \equiv (\alpha_2, \beta_2)$] are equal as follows:

1. Each participant $\mathcal{P}_i$ performs the following:

    (a) Select $z_i \in_U \mathbb{Z}_p$, where $\in_U$ denotes uniform and random selection.

    (b) Compute $(\alpha_i, \beta_i) = (\alpha^{z_i}, \beta^{z_i})$ and broadcast it.

2. All participants jointly apply the Distributed Exponential ElGamal Decryption protocol to decrypt $(\gamma, \delta) = (\Pi_{i=1}^{n}\alpha_i, \Pi_{i=1}^{n}\beta_i)$. If plaintext (γ, δ) is equal to 0, then $(\alpha_1, \beta_1) \equiv (\alpha_2, \beta_2)$; otherwise, $(\alpha_1, \beta_1) \not\equiv (\alpha_2, \beta_2)$.

2.2.5 Bulletin Board [Ben87]

A bulletin board is an append-only public broadcast channel where participants of a protocol coordinate their actions and publish their proofs. At the conclusion of the protocol, the contents of the bulletin board can be considered a transcript of an execution of the protocol.

2.2.6 Multiparty Coin Tossing [BOO10]

A coin tossing protocol allows a random bitstring to be jointly generated by a set of participants such that no participant knows the resulting bitstring, and a single honest participant is sufficient for ensuring the ensuing bitstring follows a uniform random distribution.

2.2.7 Non-Interactive Zero-Knowledge Proofs (NIZKP) [Sch91] [CP92]

Zero knowledge proofs are used for proving that a given statement is true without revealing *any* information about the statement itself beside that it is true. A protocol is zero-knowledge proof if it satisfies the following properties:

- Completeness: if the statement is *true*, the honest prover can convince the honest verifier of this fact.
- Soundness: if the statement is *false*, a malicious prover cannot convince an honest verifier that the statement is true, except with some small probability.
- Zero-knowledge: if the statement is *true*, a malicious verifier learns nothing but this fact.

Zero knowledge proofs can be adapted from basic Σ-protocols such as the Schnorr proof of knowledge of a discrete logarithm [Sch91] or the Chaum-Pedersen proof of representation of a Diffie-Hellman tuple [CP92], using Fiat-Shamir [FS90] to compile into a non-interactive zero-knowledge protocol (NIZKP). If one wishes to avoid the random oracle model, any alternative Σ-protocol to NIZKP compilation [HL10] is sufficient. Note that many protocols we use internally utilize NIZKPs, including Mix and Match, verifiable mix networks, and multiparty coin tossing.

2.2.8 Cut-and-Choose [Rab79]

When an efficient zero-knowledge proof is not admissible, we can utilize the more general technique of a cut-and-choose protocol. In a non-interactive setting, a participant wishing to demonstrate a value is well-constructed (without revealing how they were constructed) will construct a large number of candidate values and then utilize a public source of randomness, i.e., a beacon [CH10], to randomly select a subset for auditing purposes. The participant will show how each candidate in the subset was constructed, inferring belief that the remaining candidates were also constructed correctly.

2.2.9 Pedersen Commitments [Ped92]

A commitment scheme enables someone to commit to a chosen message while keeping it hidden to others, with the capability to reveal the message later. Pedersen commitment to a message $m \in \mathbb{Z}_q$ is defined as $\mathsf{com} = g^m \cdot h^r$ where g and h are fixed public elements of G and the quantity r is chosen at random in $\mathbb{Z}_q$. The generators g and h are chosen once in a way that ensures no one knows their relative discrete logarithm. Pedersen commitments are perfectly hiding and computationally binding.

Chapter 3

Literature Review

In this chapter, we first present an overview of privacy-preserving data publishing (PPDP) and survey various PPDP research proposals in the context of set-valued data in both *non-interactive* and *interactive* settings. Next, we survey confidentiality in data outsourcing for both *centralized* and *distributed* environments. Afterwards, we provide an in-depth literature review about searching on encrypted data that covers *searchable encryption*, *functional encryption* and *secure function evaluation*. Finally, we present the state of art in computations with public verifiability.

3.1 Privacy-Preserving Data Publishing

3.1.1 Overview

Privacy-preserving data publishing (PPDP) is a data release mechanism where useful information can be obtained from the released data while the privacy of individuals in the data is reserved. A common PPDP approach is *anonymization*. Several privacy models have been proposed in the literature for providing different types of privacy protection. For example, the (α, k)-anonymity model [WLFW06] applied generalization and suppression techniques to protect against record and attribute linkages. The ε-differential privacy model [Dwo06] aimed at protecting against table linkage and probabilistic attacks by ensuring that the probability distribution on the published data is the same regardless of whether or not an individual record exists in the data. Mohammed *et al.* [MCFY11] proposed a generalization-based anonymization algorithm in a non-interactive setting for releasing differentially private records for data mining. Cormode *et al.* [CPS+12] proposed a framework for using spatial data structures to provide a differentially private description of the data distribution. Xiao *et al.* [XXY10] proposed another framework that uses kd-tree based partitioning for differentially private histogram release. These frameworks support range queries while providing privacy guarantee.

Another PPDP approach for privacy protection is *anatomization.* Xiao and Tao [XT06] proposed the *anatomy* method that partitions the data vertically in order to disassociate the relationship between the quasi-identifier (QID) attributes and the sensitive attributes (ST), while satisfying the ℓ-diversity privacy requirement. This approach generates two tables: a QID table that contains all quasi-identifier attributes, and an ST table that contains all sensitive attributes. Both tables contain an anatomy group attribute (GID) such that all records belonging to the same anatomy group will have the same GID value in both tables. If an anatomy group has an ℓ distinct set of sensitive values where each value exists exactly once in the group, then the probability of linking a record to a sensitive value using GID value is $1/\ell$. Since the anatomy approach only considers the single association between QID attributes and ST attributes, Jiang *et al.* [JGWY10] proposed a new approach based on anatomy that considered the functional dependencies among the data attributes in a single table. This approach splits the table into several sub-tables according to the functional dependencies such that the decomposed sub-tables satisfy the privacy requirement of ℓ-diversity for each sensitive association. Nergiz and Clifton [NC11] proposed another decomposition approach using anatomy for query processing on outsourced data that consists of multi-relational tables.

In the rest of this chapter, we will survey privacy-preserving data publishing in the context of set-valued data in both *non-interactive* and *interactive* settings.

3.1.2 Non-Interactive Privacy-Preserving Data Publishing on Set-Valued Data

In this related line of work, the raw data is first anonymized while maintaining an effective level of data utility, and then released for the purpose of data mining and analysis.

Centralized Environment

Several approaches have been proposed to anonymize set-valued data [TMK08, HN09, TMK11, CMF$^+$11] where the data is hosted by one party. Terrovitis *et al.* [TMK08] proposed a k^m-anonymization model based on a bottom-up global generalization, where m is the maximum number of items an adversary might know in any given transaction. He and Naughton [HN09] removed the constraint on the background knowledge of the adversary to provide a k-anonymity privacy guarantee by proposing a greedy algorithm based on a top-down local generalization. Terrovitis *et al.* [TMK11] improved the efficiency of [TMK08] by proposing optimization methods, namely Local Recoding Anonymization (LRA) and Vertical Partitioning Anonymization (VPA), which are based on global recoding. Chen *et al.* [CMF$^+$11] argued that k-anonymity does not provide a sufficient privacy guarantee when applied on set-valued data, instead proposing a probabilistic top-down algorithm that utilizes context-free taxonomy tree to release set-valued data satisfying differential privacy. Although data anonymization has been studied extensively on relational data [FWCY10], set-valued data is typically high-dimensional so solutions for relational data are ineffective due to the curse of high dimensionality [Don00].

Distributed Environment

To the best of our knowledge, no work has been done on anonymizing set-valued data from different sources for the purpose of data release. However, there is a related line of work on relational data [JX09, JC06, MAFD14, AMFD12]. Jiang and Clifton [JC06] proposed an algorithm for secure integration of vertically partitioned data between two parties while satisfying k-anonymity. Jurczyk and Xiong [JX09] proposed a multi-party framework for horizontally partitioned data over n parties, where $n > 2$. The output is an integrated and anonymized data satisfying *l-site-diversity*, where l represents the minimal number of partitions to which records in each equivalence class belong. Al-hadidi *et al.* [AMFD12] and Mohammed *et al.* [MAFD14] proposed two-party protocols for securely generating integrated data satisfying ε-differential privacy from horizontally partitioned data and vertically partitioned data, respectively. Unlike our protocol that supports both horizontally and vertically partitioned data in the malicious adversarial model, the aforementioned approaches handle data that is either horizontally or vertically partitioned, in the semi-honest adversarial model. Mohammed *et al.* [MFD11] proposed a method for integrating vertically-partitioned relational data in the malicious setting; however, this method satisfies the k-anonymity privacy model and it is not publicly verifiable.

3.1.3 Interactive Privacy-Preserving Data Publishing on Set-Valued Data

This is another related line of work where the data owners jointly compute a data mining function on their private data, and learn the correct output and nothing else.

Centralized Environment

There have been serval works concerned with privacy-preserving data mining on a single set-valued data [BLST10, LQSC12, WCH+07, GLM+13b]. Bhaskar *et al.* [BLST10] and Li *et al.* [LQSC12] proposed two different approaches for mining frequent itemsets from set-valued data such that the output data satisfy differential privacy. Wong *et al.* [WCH+07] and Giannotti *et al.* [GLM+13b] focused on the problem of secure association rule mining on a set-valued data in an outsourcing environment. In [WCH+07], the authors proposed an encryption scheme based on substitution cipher techniques in order to non-deterministically map each item in a transaction to a set of n items, but no formal privacy guarantee was provided. The authors in [GLM+13b] proposed an encryption scheme for data transformation that satisfies *item k-anonymity*, i.e., each item in the outsourced set-valued data is indistinguishable from at least k-1 items.

Distributed Environment

Kantarcioglu and Clifton [KC04] proposed a multi-party approach for privacy-preserving mining of association rules on horizontally partitioned set-valued data. Using commutative encryption to design a secure two-party protocol for division computation, Zhang *et al.* [ZRZ+13] proposed a two-party approach for privacy-preserving association rule mining on set-valued data. These methods cannot be applied to our problem as they do not provide any formal privacy guarantee on the output data. Wahab *et al.* [WHZ+14] recently proposed a multi-party approach for mining association rules while satisfying the differential privacy model. However, this approach requires a trusted central authority and assumes that all parties operate is the semi-honest setting. There have been several works about privacy-preserving data mining on relational data that guarantee certain level of privacy on the output. For instance, Dwork *et al.* [DKM+06] proposed a method for answering count queries on a horizontally partitioned data in a malicious multi-party environment, by returning noisy counts that satisfy differential privacy. Narayan and Haeberlen [NH12] proposed a protocol for a semi-honest two and multiple party environment where the answers to SQL-style count queries are noisy counts satisfying differential privacy. These approaches are not applicable to the problem in Chapter 5 as they are neither designed to handle high dimensional data, nor support public verifiability.

3.2 Confidentiality in Data Outsourcing

Another area related to our work is *confidentiality in data outsourcing*, where data is stored and managed by one or more untrusted parties that are different from the data owner. Queries are executed on the data while keeping the data confidential and without revealing information about the queries.

3.2.1 Centralized Environment

A commonly used mechanism for ensuring data confidentiality is *encryption*. Some approaches proposed to process queries over encrypted data directly. However, such approaches do not provide a good balance between data confidentiality and query execution. For example, methods in [HILM02] [HMT04] attach range labels to the encrypted data, thus revealing the underlying distributions of the data. Other methods depend on order-preserving encryption [AKSX04][EAAG06]; however, these methods reveal the data order and are subject to inference and statistical attacks. Homomorphic encryption, on the other hand, is a promising public cryptosystem that allows query execution on encrypted data [GZ07a][Gen09]; however, its high computation cost makes it prohibitive in practice. In Chapter 4, we employ the exponential variation of ElGamal [CGS97] encryption scheme in one area of our solution by taking advantage of its additive homomorphism property. We show that this scheme is efficiently employed because the encrypted message is small enough for the scheme to remain practical.

Instead of processing queries directly over encrypted data, some approaches have proposed using indexing structures for fast data access and efficient query execution [WCL$^+$10][WL06][GZ07b]. Some indexing schemes have constraints on the type of queries they support. For example, hash-based indexing [DVJ$^+$03] and privacy homomorphism [HIM04] only support equality queries, whereas bucket-based indexing [HILM02] and character-oriented indexing [WDWS04][WWS05] support equality queries as well as partially supporting range queries. To support both equality queries and range queries, a category of approaches propose using disk-based indexes such as B-tree [BM70] and B$^+$-tree [Com79] and spatial access indexes such as kd-tree [Ben75] and R-tree [Gut84]. Our work in Chapter 4 fits in this category because we utilize an encrypted kd-tree index for efficient and secure traversal. Wang *et al.* [WAEA11] proposed a framework based on B$^+$-tree index for query processing on relational data in the cloud. However, in order to protect data confidentiality against the cloud, the proposed solution generates a superset of the result and requires the client (querying user) to perform predicates evaluation in order to compute the final result. Hu *et al.* [HXRC11] proposed a framework based on R-tree index for secure data access and processing of k-nearest-neighbor (kNN) similarity queries. However, the proposed approach partitions the R-tree index constructed over the outsourced data into two indexes, one is hosted by the cloud and the other is hosted by the client.

In addition, a high communication bandwidth is required to achieve access confidentiality. Recently, Wang and Ravishankar [WR13] proposed a framework for performing half-space range queries using an $\widehat{\text{R}}$-tree index that is encrypted using Asymmetric Scalar-product Preserving Encryption (ASPE) scheme [WCKM09]. Their method ensures data confidentiality and requires low communication and storage overhead on the client side. However, it does not provide a privacy guarantee, nor does it provide full confidential query processing because it leaks information on the ordering of the minimum bounding box of the leaf nodes and requires result postprocessing because it introduces false positives. Barouti *et al.* [BAAD14] proposed a protocol for secure storage of patient health records on the cloud, while allowing health organizations to securely query the data. The proposed protocol, however, does not provide privacy guarantees on the query results, while requiring high communication overhead on the client side. Blass *et al.* [BNVH12] proposed a confidential pattern counting protocol for clouds using MapReduce. The authors propose an efficient somewhat homomorphic encryption scheme that preserve the confidentiality of the data and the queries. However, similar to [BAAD14], the proposed approach does not provide privacy guarantees on the query results.

3.2.2 Distributed Environment

In a distributed data outsourcing environment, the data is partitioned and outsourced to a set of independent and non-colluding servers. To ensure data and query confidentiality, a distributed protocol is needed to securely process the query without disclosing the data in each of the outsourcing servers. Such protocols typically use secure multiparty computation (SMC) [Yao82][Gol04], a cryptographic technique that computes a secure function from multiple participants in a distributed network. For example, Shaneck *et al.* [SKK09] proposed an approach for computing kNN queries on horizontally partitioned data, Vaidya and Clifton [VC05] proposed an approach for secure answering of top k queries on vertically partitioned data while satisfying k-anonymity, whereas Rastogi and Nath [RN10] and Shi *et al.* [SCR$^+$11] addressed the problem of private aggregation of time-series data such that the outcome statistic is differentially private.

3.3 Searching on Encrypted Data

3.3.1 Searchable Encryption (SE)

Searchable encryption (SE) [SWP00] [DRD08] [BTHJ12] [JJK$^+$13] allows encrypted data hosted remotely by a third party to be searched by the host on behalf of the data owner while allowing the former to learn as little information as possible about the underlying plaintext data. Depending on whether it uses public symmetric encryption or public key encryption, SE schemes can be respectively split into either searchable symmetric encryption or public key encryption with keyword search. SE assumes that the encrypted data generated by the data owner and then stored on untrusted remote server (e.g. cloud). The user who is interested in obtaining some information from the data generates a trapdoor for the server to use for searching the data. Depending on the number of the data owners and users, SE can be divided into four categories: *single owner & single user*, *single owner & multi-user*, *multi-owner & single user*, and *multi-owner & multi-user*.

Single Owner & Single User (SOSU)

This setting is the most suitable for data outsourcing, where the data owner encrypts the data and creates trapdoors that will be sent to the host to conduct the search. Several SE schemes have been proposed to support searching on a *single* keyword [SWP00][Goh03][CM05][CGKO06][VLSD$^+$10] [CK10][KPR12]. Song *et al.* [SWP00] used sequential scan to propose the first usable SE scheme. The approach is to encrypt each word separately, and then enclose a hash value with specific pattern in the ciphertext. The search is performed by the server by extracting the hash value and verify that its pattern matches the pattern of the search keyword. In [Goh03][CM05], the authors constructed an index per each search document, where the former uses Bloom filters [Blo70] and the latter uses a set of bits each of which represents on search keyword stored in advance in a lookup table. Rather than indexing each document, the authors in [CGKO06][VLSD$^+$10] proposed different SE schemes that create an index per each search keyword, where inverted indexes and pseudo-random functions are used respectively. Chase and Kamara [CK10] also utilized inverted indexes to propose

SE schemes for searching on matrix-structured data and labeled data, while supporting the concept of *controlled disclosure* such that access is granted only to part of a data. Kamara [KPR12] later extended [CGKO06] by proposing a scheme that supports sub-linear search time, compact indexes and efficient data update.

On the other hand, several SE schemes have also been proposed to support searching on *multiple* keywords [GSW04][BKM05][BLL06][WWP08a][CJJ$^+$13][ABCK09][LWW$^+$10]. Assuming the existence of keyword fields in the search documents, Golle *et al.* [GSW04] was the first to propose a SE scheme for *conjunctive keyword search* with amortized linear cost. Ballard *et al.* [BKM05] used Shamir's threshold secret sharing to construct a scheme that is linear w.r.t. the number of search documents. Byun *et al.* [BLL06] later proposed a more efficient scheme with respect to communication and storage costs using bilinear mapping. Wang *et al.* [WWP08a] proposed the first SE scheme for conjunctive keyword search without assuming the existence of keyword fields using also bilinear mapping. Based on [CGKO06], Cash *et al.* [CJJ$^+$13] recently proposed an efficient (sub-linear) scheme for searching on unstructured and encrypted data. The proposed approach represents a trade-off between privacy and efficiency, where information about data access patterns is leaked to the server while the underlying data and queries remain confidential. To support more advanced search queries on encrypted data such as *fuzzy* search, Adjedj *et al.* [ABCK09] utilized *locality sensitive hashing* to reduce the dimensionality of biometric data and propose a symmetric searchable encryption scheme for biometric identification. The approach in [LWW$^+$10] proposed by Li *et al.* also supports fuzzy search, where edit distance is used to measure the similarity between keyword semantics and to consequently determine the closest files to the search query.

Single Owner & Multi-User (SOMU)

This setting is more suitable for data sharing, where the data owner encrypts the data and then share it with other parties who will be able to create trapdoors and perform their own search. A natural way to achieve that is to extend an existing SOSU approach and make it suitable for multi-users. In [CGKO06], the authors described how *broadcast encryption* can be used to extend their SOSU scheme (described earlier) to support a searchable symmetric encryption for multi-users in the SOMU setting. The search documents are first encrypted using the according to the SOSU scheme, and then the users receive the key for generating the trapdoors via broadcast encryption. The proposed construction does not require authentication, and supports revocation of users such that revoked user cannot anymore conduct searches on the encrypted documents. *Proxy re-encryption* [BBS98] allows a party to decrypt on behalf of another party by enabling a third party to convert a ciphertext for one party to another without obtaining any knowledge about their secret keys. In [RVBM09], Raykova *et al.* constructed a SOMU scheme by introducing *re-routable encryption*, a stronger notion of proxy encryption that protects the identities of the parties involve. The authors utilized Bloom filters to build the search structures, where for each search document a Bloom filter index is created. They also assumed the existence of a *query router* server, that performs user authentication, as well as queries and results transformation. Yang *et al.* [YLW11] utilized symmetric bilinear mapping to propose a scheme for data outsourcing with multi-users. Each user is issued a unique key for generating its own queries, while the cloud maintain a list of helper keys for all authorized users. Given a user's search query and his helper key, the cloud constructs a common key based on bilinear mapping to search the encrypted index. The proposed scheme supports *query unforgeability*, where neither the cloud nor other users can construct a valid query on behalf of a user. It also supports efficient user revocation, where the revocation process does not require key renewal for non-revoked users, nor requires update to the encrypted index.

Multi-Owner & Single User (MOSU)

In this setting, there exist several data owners, each of which encrypts its own data to target a certain recipient. Most of the schemes we will review next are based on public key encryption.

With respect to searching on a *single* keyword, some MOSU schemes in the literature [BDCOP04] [BSNS08][ABC^{+}08] were constructed based on *identity based encryption* [Coc01][BF03]. Boneh *et al.* [BDCOP04] proposed the first MOSU scheme by considering the scenario where an email sender encrypts its email with the public key of the receiver, while allowing the routing email server to search for a specific keyword in the encrypted email before it forwards it to the receiver. Using its private key, the receiver generates a trapdoor for a search keyword and sends it to the mail server via a secure channel, where the latter uses it to perform the search while learning nothing about the underlying text except of whether or not it contains the search keyword. This system is called *non-interactive public key encryption with keyword search*, as it allows for keyword search without requiring any interaction between the sender and the receiver. The server, however, can store all the trapdoors it receives from the receiver, and conduct search by itself on future emails. To address this issue, Baek *et al.* [BSNS08] proposed a MOSU scheme that enables search keywords that are used frequently to be *refreshed* by attaching attaching a time period to them. The shorter the time period is, the more secure can be obtained. This scheme also removes the need for a secure channel between the server and the receiver in [BDCOP04] by maintaining a private/public key pair, where the sender uses the public keys of the receiver and the server for encryption, while the the receiver encrypts its trapdoors using the public key of the server and sends them through public channels.

Several SE schemes have also been proposed in the literature to support searching on *multiple* keywords [BSNS08][PCL05][HL07][BCK09][SVLN$^+$10]. Baek *et al.* [BSNS08] extended [BDCOP04] (discussed above) to support conjunctive keyword search. In [PCL05], Park *et al.* enabled the hosting server (proxy) to decrypt the documents containing desired search keywords without having access to the user's private decryption key. The server, however, cannot decrypt other documents that do not contain the search keywords. In [HL07], Hwang *et al.* provided a conjunctive keyword search scheme that optimizes the storage and communication overhead of both the server and the user. To support more advanced search queries, Bringer *et al.* [BCK09] used *Bloom filters with storage* [BKOSI07] to propose a fuzzy search scheme for biometric identification over encrypted person-specific data. The approach separates the stored data on the server from its indexes, while using locality-sensitive hashing functions to allow for error-tolerance. Based on *hidden vector encryption* [BW07], Sedghi *et al.* [SVLN$^+$10] used symmetric bilinear pairings of prime order to propose an effluent MOSU scheme that supports wildcard search while accepting keywords over any alphabet.

Multi-Owner & Multi-User (MOMU)

To search on a single keyword, Bellare *et al.* [BBO07] proposed a MOMU scheme where the encryption algorithm is *deterministic* to achieve higher efficiency. To be able to conduct the search, a deterministic hash of the plaintext is attached to the ciphertext. The scheme is length-preserving, where each ciphertext and its corresponding plaintext both have the same length. In [BDDY08], Bao *et al.* constructed a scheme for the scenario where a group of data owners, each of which has its private key, can encrypt their documents separately and send them to the hosting server, while each data owner can generate their own trapdoors and send them to the server to search on all existing documents. The proposed scheme allows for dynamic data owner enrollment, provides transparent user revocation, and supports query unforgeability. Several schemes, including [HL07] [WWP09] [WWP08b], were proposed in the literature in the MOMU setting that support searching on *multiple* keywords. No MOMU schemes, however, have been proposed so far to support an advanced level of search queries such as fuzzy or symmetric search.

3.3.2 Functional Encryption (FE)

Functional encryption (FE) [BSW11] is a recent public-key encryption paradigm that enables both fine-grained access control and selective computation on encrypted data. In FE system, there exists a master secret key MSK that is held by a trusted authority. Let $x = (I, m)$ be the encryption of message m and its access control I, and let f be a function to be evaluated over x. The trusted authority uses MSK to derive a secret key SK_f that is associated with the function f. Given x, anyone can evaluate $f[x]$ using secret key SK_f.

Depending on whether or not the access control index is public, most of the functional encryption systems belong to one of the following two classes: ***predicate encryption***, which includes *anonymous identity-based encryption (A-IBE)*, *hidden vector encryption (HVE)*, and *inner product predicate (IPP)*, and ***predicate encryption with public index***, which includes *identity-based encryption (IBE)* [Coc01][BF03], *attribute-based encryption (ABE)* [BSW07], and *predicate encryption*

(PE) [BW07]. In this review, we focus on ABE since it supports fine-grained access control that can be utilized to handle not only keywords but also numerical ranges. Sahai and Waters [SW05] introduced attribute-based encryption as a new concept of public encryption schemes that allow data owners to encrypt their data while setting a policy indicating who can decrypt this data. There are two types of attribute-based encryption schemes: *key-policy attribute-based encryption (KP-ABE)*, and *cipher-policy attribute-based encryption (CP-ABE)*. In the key-policy attribute based encryption schemes [GPSW06][OSW07][KSW08][LDLW14], the message is encrypted and a ciphertext with a set of attributes is generated. The decryption of the message is achieved using a secret key with an access structure if the access structure is satisfied by the set of ciphertext attributes. On the other hand, with regard to cipher-policy attribute-based encryption schemes [BSW07][LS08][Wat11][HSM+14], a set of attributes is associated with a secret key, while an access structure (ciphertext policy) is associated with a ciphertext. The decryption of the message is achieved using a secret key with a set of attributes if the secret key's attribute set satisfies the access structure associated with the ciphertext.

3.3.3 Secure Function Evaluation (SFE)

Assume f is a function to be executed between a set of parties with individual inputs. Using *secure function evaluation (SFE)*, each party learns only the output of the function and is otherwise assured the confidentiality of their individual inputs. Yao [Yao86] introduced the first two-party protocol for SFE, which was later generalized to secure multi-party computation by Goldreich *et al.* [GMW87]. Our protocols follow the line of research utilizing threshold homomorphic cryptosystems to provide SFE. This paradigm, also known as computing on encrypted data, was introduced by Franklin and Harber in the semi-honest model for two/multi party protocols [FH94], and extended to the malicious model by Cramer *et al.* [CDN01] for the multiparty setting, and Schoenmakers and Tuyls [ST04] for the two-party setting. Jakobsson and Juels [JJ00] provided a general protocol in the malicious model that is adaptable to both settings.

3.4 Public Verifiability

To ensure a protocol executes correctly in the presence of malicious adversaries, participants must prove they are following the protocol at each step. Often such proofs are convincing to the other participants in the protocol and contingent on their interaction or individual secrets. A publicly (or universally) verifiable protocol produces a transcript proving correct execution that is verifiable by anyone at any time. In the database community, verifiability has been studied in the areas of data authentication [Tam03][DBP07], data streams [CCM09][CKLR11], and public auditing [WCW$^+$13][WLL12]. In *verifiable computation*, a single data owner delegates a computationally heavy task to the cloud while being able to verify the results [BGV11][CKV10][GGP10][GKR08]. Outside of databases, public verifiability is an important property for a number of cryptographic protocols, including *voting* schemes [Ben87][CGS97], *auction* schemes [LKM01][DLL13], and *mix network* schemes[PZ12][JJR].

Chapter 4

Secure and Privacy-preserving Data Outsourcing

4.1 Introduction

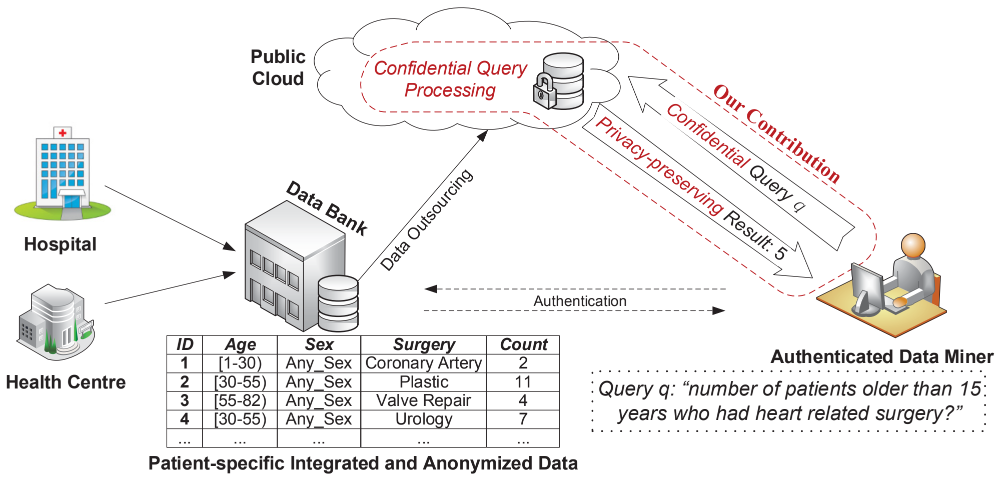
In recent years, there has been a considerable effort to ensure data confidentiality and integrity of outsourced databases on the cloud. Several research proposals suggest encrypting the data before moving it to the cloud [GZ07a, PRZB11]. While encryption can provide data confidentiality, it is less effective in deterring inference attacks. This reality demands new privacy-enhancing technologies that can simultaneously provide data confidentiality and prevent inference attacks due to aggregate query answering. Privacy-preserving data publishing (PPDP) is the process of anonymizing person-specific information for the purpose of protecting individuals' privacy while maintaining an effective level of data utility for data mining. Different PPDP privacy models provide different types of privacy protection [FWCY10]. Differential privacy [Dwo06] is a recently proposed privacy model that provides a provable privacy guarantee. Differential privacy is a rigorous privacy model that makes no assumption about an adversary's background knowledge. A differentially-private mechanism ensures that the probability of any output (released data) is equally likely from all nearly identical input data sets and thus guarantees that all outputs are insensitive to any individual's data.

In this chapter, we propose a cloud-based query processing framework that simultaneously preserves the confidentiality of the data and the query requests, while providing differential privacy guarantee on the query results to protect against inference attacks. Let us consider the following real-life scenario. Population Data BC (PopData) [1] is a non-profit organization (data bank) responsible (among other things) for storing and managing patient-specific health data received from several hospitals, health organizations and government agencies in the Province of British Colombia, Canada. PopData utilizes explicit identifiers to integrate the data, and then de-identifies the integrated data by separating the explicit identifiers from the rest of the data contents. Data miners interested in querying the data initially sign a non-identifiability agreement to prevent them from releasing research data that can be used to re-identify individuals. When PopData receives a data mining query, it first authenticates the data miner, verifies that she is working on an approved research project, and then executes the query on the de-identified data and returns the result back to the data miner. Similar organizations can be found in other countries, e.g., the National Statistical Service [2] in Australia.

A major concern in this scenario is *data privacy*. Although the data is de-identified, data miners can still perform (or accidentally release a research results that can leads to) record/attribute linkage attacks and re-identification of individuals, as was shown in the cases of AOL [BZ06] and Netflix [NS08]. On the other hand, to minimize the workload on PopData, cloud services can be used to store, manage, and answer queries on the integrated data. However, this rises two other concerns. One concern is *data confidentiality*, where the outsourced patient-specific data must be stored in a protected way to prevent the cloud from answering queries from unauthorized data miners, and to protect against potential multi-tenancy problems due to the sharing of services, resources, and physical infrastructure between multiple independent tenants on the cloud [DWC10]. Another concern is *query confidentiality*, where the cloud should be able to execute query requests from authorized

[1]PopData: https://www.popdata.bc.ca/

[2]Statistical Data Integration Involving Commonwealth Data: http://statistical-data-integration.govspace.gov.au/


Public Cloud
Confidential Query Processing
Our Contribution
Confidential Query q
Privacy-preserving Result: 5
Data Outsourcing
Data Bank
Hospital
Health Centre
Authentication
Authenticated Data Miner
Query q: "number of patients older than 15 years who had heart related surgery?"
Patient-specific Integrated and Anonymized Data

ID	Age	Sex	Surgery	Count
1	[1-30)	Any_Sex	Coronary Artery	2
2	[30-55)	Any_Sex	Plastic	11
3	[55-82)	Any_Sex	Valve Repair	4
4	[30-55)	Any_Sex	Urology	7
...	...	...	...	...

Figure 1: Security requirements for data outsourcing.

data miners without the ability to know what attributes and attribute values are specified in each query.

As shown by [BDMN05], count queries can be quite useful for data mining and statistical analysis applications where miners focus on extracting new trends and patterns from the overall data and are less interested in particular records.

Figure 1 illustrates the overall process of our proposed framework. Each data owner (e.g. hospital, health center) submits its raw data to the data bank (data provider). The data bank first integrates all data together, and then applies a PPDP privacy model on the integrated data such that explicit identifiers of record owners are removed, while other attributes (including sensitive attributes) are anonymized and retained for data analysis. Next, the data bank encrypts the anonymized data and upload it to the service provider (public cloud). Data miners authenticate themselves to the data bank and then submit their encrypted count queries to the cloud. The cloud securely processes each query, homomorphically computes the exact noisy count, and then sends the encrypted result back to the data miner. The proposed framework, named SecDM, achieves data privacy by supporting any privacy algorithm whose output is a contingency table data. Attribute-base Encryption (ABE) and ElGamal schemes are used to achieve data and query confidentiality. While our framework protects the confidentiality of individual query (data access), we provide a detailed security analysis in Section 4.5.3. We analyze in Section 4.4.4 the benefit of outsourcing the data to a service provider as compared to having the data bank handle the user queries directly and show that the processing overhead on the data bank is almost 10 times less than the overhead on the service provider.

The intuition of our solution is to generate a kd-tree index for efficient traversal and secure access on the anonymized data, where the index tree is encrypted using *attribute based encryption* and stored on the public cloud. When a data miner desires to query the outsourced data, she sends her proof of identity to the data provider with her query and receives an encrypted version of her query, namely, system query, which she sends to the cloud for processing. The cloud uses the system query to traverse the encrypted kd-tree index and securely compute the total count representing the privacy-preserving answer to the query. The cloud then sends the answer back to the data miner, who in turn decrypts the encrypted results using a decryption key provided originally by the data provider. Our framework protects the confidentiality of each individual query by its predicates hidden from the cloud. However, it does not hide the search pattern of the queries. We provide formal definition of framework properties as well as detailed security analysis in Section 4.5.3.

The contributions of this chapter can be summarized as follows:

Contribution 1. To the best of our knowledge, this is the first work that proposes a comprehensive privacy-preserving framework for query processing in a cloud computing environment. The proposed framework maintains the privacy and utility properties of the outsourced data while simultaneously ensuring *data confidentiality*, *query confidentiality*, and *privacy-preserving results*. Previous work [BAAD14][GLM$^+$13a][WAEA11][Yon14][TH13] satisfies only a subset of the aforementioned security features.

Contribution 2. To ensure efficient data access while maintaining data confidentiality, we present an algorithm for constructing an encrypted kd-tree index that hides the data from the cloud while allowing for confidential query processing. We utilize *attribute based encryption* in a unique way to handle range predicates on numerical attributes.

Contribution 3. Most existing work on the problem of data outsourcing in cloud computing environments either requires the query issuer to have prior knowledge about the data and subsequently requires storage and communication overhead [WAEA11], or yields results that require postprocessing on the query issuer's side [HILM02], or both [HXRC11]. In contrast, data miners in our proposed framework are considered "lightweight clients" as they are not required to have or store any information about the data, nor are they required to perform post-processing on the results (except for decrypting the results). The communication complexity with the cloud is constant with respect to the size of the dataset and the query type.

Contribution 4. SecDM has two major steps, namely index construction and query processing. It has linear time complexity on both steps w.r.t. the number of attributes, and it is sub-linear w.r.t. the data size on query processing. Extensive experiments on real-life data further confirm these properties.

In Table 2, we summarize the features of the representative approaches in the areas related to our work, including our proposed solutions. Unlike the other approaches, our proposed solution ensures data and query confidentiality and privacy-preserving results while assuming that the client has no prior knowledge about the data being queried and its structure. No further interaction is required between the cloud and the client once the latter has submitted her query to the cloud, and no local refinement is required by the client on the final result.

The results of this chapter are currently under review in TCC [DFMC].

4.2 Preliminaries

The framework presented in this chapter utilizes two main components: *anonymous ciphertext-policy attribute based encryption* (ACP-ABE) scheme [NYO08], and *DiffGen* Algorithm [MCFY11] for data anonymization. In this section, we first introduce the ACP-ABE scheme and define its properties and main functions, and then we introduce *DiffGen* and explain how it works.

Table 2: Comparative evaluation of main features in related query processing approaches (properties in columns are positioned as beneficial with fulfilment denoted by ● and partial fulfilment by ○)

	Supported Queries			Security			Client			
Approach	**Exact Query**	**Range Query**	**Similarity Query**	**Data Confidentiality**	**Query Confidentiality**	**Privacy-preserving Result**	**Query Reuse**	**Low Storage Overhead**	**Low Communication Overhead**	**Low Post-processing Overhead**
Private Spatial Decomposition (PSD) [CPS+12]	●	●				●		●	●	
SQL over Encrypted Data (SED) [HILM02]	●	●		●	●		●		●	
OPESS [WL06]	●	●		●	●				●	
Salted IDA [WAEA11]	●	●			●					
ASM-PH [HXRC11]	●		●		●					
NEST [HXL13]			●					●	●	●
PPNNS [SKK09]	●		●		●	●		●		●
$\widehat{\text{R}}$-tree [WR13]	●	●		●	○			●	●	
kd-PHR [BAAD14]	●	●		●	●			●		○
Our proposed solution SecDM [Section 4.4.1-4.4.3]	●	●		●	●	●	●	●	●	○
Our proposed solution C-SecDM [Section 4.4.4]	●	●		●	●	●		●	●	●

4.2.1 Anonymous Ciphertext-Policy Attribute Based Encryption (ACP-ABE)

In this chapter, we utilize CP-ABE to preserve the confidentiality of the data mining queries and the outsourced data (i.e. the data index hosted on the cloud). Our proposed framework requires the CP-ABE scheme to support attributes with multiple values, including the wildcard functionality (to indicate that certain attributes are not relevant to the ciphertext policy), while hiding the details of the access structures associated with ciphertexts. A good candidate satisfying the aforementioned properties is the CP-ABE scheme proposed in [NYO08]. This scheme is secure under the Decisional Bilinear Diffie-Hellman (DBDH) assumption [Jou00] and the Decision Linear (D-Linear) assumption [BBS04], constructed in the multi-valued attribute setting, supports wildcards, and allows the access structure to be expressed in conjunctive normal form (i.e. conjunction - AND of disjunctions - OR). It also prevents the decryptor from obtaining information about the access structure by hiding what values for each attribute is specified in the conjunction of all the attributes.

Given a set of attributes $\{A_1, \ldots, A_i, \ldots, A_n\}$, where each attribute A_i has a domain $\Omega(A_i) = \{v_{i,1}, \ldots, v_{i,j}, \ldots, v_{i,|\Omega(A_i)|}\}$, the scheme can be constructed according to the following four algorithms:

$\mathsf{Setup}(1^\lambda)$**.** A trusted authority first runs $\mathsf{Gen}(1^\lambda)$ to generate a tuple $[p, \mathbb{G}, \mathbb{G}_T, g \in \mathbb{G}, e]$, where $\mathbb{G}, \mathbb{G}_T$ are cyclic groups, $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map, and $\omega {\in_R} \mathbb{Z}_p^*$. For each attribute $A_i : 1 \leq i \leq n$, generate values $\{a_{i,j}, b_{i,j} \in_R \mathbb{Z}_p^*\}$ and point $\mathcal{A}_{i,j} \in_R \mathbb{G}$ for each attribute value $v_{i,j} \in \Omega(A_i) : 1 \leq j \leq |\Omega(A_i)|$. The algorithm outputs public key $PK = \langle Y, p, \mathbb{G}, \mathbb{G}_T, g, e, \{(\mathcal{A}_{i,j})^{a_{i,j}}, (\mathcal{A}_{i,j})^{b_{i,j}}\}_{1 \leq j \leq |\Omega(A_i)|}^{1 \leq i \leq n} \rangle$, where $Y = e(g, g)^\omega$. It also outputs master secret key $MSK = \langle \omega, \{a_{i,j}, b_{i,j}\}_{1 \leq j \leq |\Omega(A_i)|}^{1 \leq i \leq n} \rangle$.

$\mathsf{KeyGen}(MSK, L)$**.** This algorithm takes MSK and a set of attribute values $L = \{v_{1,t_1}, \ldots, v_{i,t_i}, \ldots, v_{n,t_n}\}$, where $v_{i,t_i} \in \Omega(A_i)$, and outputs a user's secret key $SK_L = \langle D_0, \{D_{i,0}, D_{i,1}, D_{i,2}\}_{1 \leq i \leq n} \rangle$, where $D_0 = g^{\omega - s}$, $D_{i,0} = g^{s_i} (\mathcal{A}_{i,j})^{a_{i,t_i} \cdot b_{i,t_i} \cdot \lambda_i}$, $D_{i,1} = g^{a_{i,t_i} \cdot \lambda_i}$, $D_{i,2} = g^{b_{i,t_i} \cdot \lambda_i}$, and $s_i, \lambda_i \in_R \mathbb{Z}_p^*$ for $1 \leq i \leq n$ such that $s = \sum_{i=1}^n s_i$. This algorithm will be used to encrypt user queries and generate corresponding system queries that will be executed against the encrypted data on the cloud.

$\mathsf{Enc}(PK, M, W)$**.** This algorithm takes public key PK and access structure $W = \{W_1 \wedge \ldots \wedge W_i \wedge \ldots \wedge, W_n\}$, and encrypts a message $M \in \mathbb{G}_T$. The result is a ciphertext: $CT = \langle \tilde{C}, C_0, \{C_{i,j,1}, C_{i,j,2}\}_{1 \leq j \leq \Omega(A_i)}^{1 \leq i \leq n} \rangle$, where $r \in_R \mathbb{Z}_p^*$, $\tilde{C} = MY^r$, and $C_0 = g^r$. If $v_{i,j} \in W_i$, then $[C_{i,j,1}, C_{i,j,2}] = [(A_{i,j})^{b_{i,j} \cdot r_{i,j}}, (A_{i,j})^{a_{i,j} \cdot (r - r_{i,j})}]$ (well-formed group elements), where $r_{i,j} \in_R \mathbb{Z}_p^*$. If $v_{i,j} \notin W_i$, then $C_{i,j,1}, C_{i,j,2} \in_R \mathbb{Z}_p^*$ (mal-formed group elements).

$\mathsf{Dec}(CT, SK_L)$**.** This algorithm decrypts ciphertext CT using user's secret key SK_L as follows: $M = \frac{\tilde{C} \cdot \prod_{i=1}^{n} e(C'_{i,1}, D_{i,1}) e(C'_{i,2}, D_{i,2})}{e(C_0, D_0) \cdot \prod_{i=1}^{n} e(C_0, D_{i,0})}$, where $[C'_{i,1}, C'_{i,2}] := [C_{i,j,1}, C_{i,j,2}]$ if $L_i = v_{i,j_i}$.

In the rest of the chapter we will refer to the ACP-ABE scheme as $\mathbb{A}$, and to the Exponential ElGamal scheme as $\mathbb{G}$.

4.2.2 DiffGen Algorithm

Mohammed *et al.* [MCFY11] proposed data anonymization algorithm for acheiving differential privacy in the non-interactive setting based on the generalization technique. The general idea is to anonymize the raw data D by a sequence of specializations, starting from the topmost general state. A specialization, written $v \to \mathit{child}(v)$, where $\mathit{child}(v)$ denotes the set of child values of v, replaces the parent value v with child values. The specialization process can be viewed as pushing the cut of each taxonomy tree downwards. A *cut* of the taxonomy tree for an attribute $A_i \in A^{pr}$, denoted by $Cut(\mathbb{T}^{A_i})$, contains exactly one value on each root-to-leaf path. The specialization starts from the topmost cut and pushes down the cut iteratively by specializing a value in the current cut.

DiffGen presented in Algorithm 1 first generalizes the raw data and then adds noise to achieve ε-differential privacy. Initially, all values in $\mathcal{A}^{pr}$ are generalized to the topmost value in their taxonomy trees (Line 1), and Cut_i contains the topmost value for each attribute A_i^{pr} (Line 2). At each iteration DiffGen uses exponential mechanism to select a candidate $v \in \cup Cut_i$ for specialization, where $\cup Cut_i$ is the set of all candidate values for specialization (Line 7). Candidates are selected based on their

ALGORITHM 1: (DiffGen)

```
Input: Raw data set $D$, privacy budget $\varepsilon$, and number of specializations $h$.
Output: Anonymized data set $\hat{D}$
1: Initialize every value in $D$ to the topmost value;
2: Initialize $Cut_i$ to include the topmost value;
3: $\varepsilon' \leftarrow \frac{\varepsilon}{2(|A_n^{pr}|+2h)}$;
4: Determine the split value for each $v^n \in \cup Cut_i$ with probability $\propto \mathtt{exp}(\frac{\varepsilon'}{2\Delta\mathtt{u}}\mathtt{u}(D, v^n))$;
5: Compute the score $\forall v \in \cup Cut_i$;
6: for $l = 1$ to $h$ do
7:     Select $v \in \cup Cut_i$ with probability $\propto \mathtt{exp}(\frac{\varepsilon'}{2\Delta\mathtt{u}}\mathtt{u}(D, v))$;
8:     Specialize $v$ on $D$ and update $\cup Cut_i$;
9:     Determine the split value for each new $v^n \in \cup Cut_i$ with probability $\propto \mathtt{exp}(\frac{\varepsilon'}{2\Delta\mathtt{u}}\mathtt{u}(D, v^n))$;
10:    Compute the score for each new $v \in \cup Cut_i$;
11: end for
12: return each leaf node with count $(C + \mathtt{Lap}(2/\varepsilon))$
```

score values, and different utility functions can be used to determine the scores of the candidates. Once a candidate is determined, DiffGen splits the records into child partitions. The split value of a categorical attribute is determined according to the taxonomy tree of the attribute. Since the taxonomy tree is fixed, the sensitivity of the split value is 0. Therefore, splitting the records according to the taxonomy tree does not violate ε-differential privacy. For numerical attributes, a split value cannot be directly chosen from the attribute values that appear in the data set D because the probability of selecting the same split value from a different data set D' not containing this value is 0. In this context, DiffGen uses the exponential mechanism again to determine the split value for each numerical candidate $v^n \in \cup Cut_i$ (Lines 4 and 9). Then, the algorithm specializes v and updates $\cup Cut_i$ (Line 8). It also calculates the scores of the new candidates due to the specialization (Line 10). Finally, the algorithm outputs each leaf partition along with their noisy counts (Line 12) (see [MCFY11] for more details).

4.3 Problem Formulation

In this section we formally define the research problem. First, we present an overview of the problem of confidential query processing, with privacy guarantee on outsourced data in the cloud in Section 4.3.1. Next, we define the input components in Section 4.3.2. We then describe the trust and adversarial model in Section 4.3.3. Finally, we present the problem statement in Section 4.3.4.

4.3.1 Problem Overview

In this chapter we examine a cloud computing model consisting of three parties: *data provider*, *data miner*, and *service provider*. The data provider, for example, represents a data bank that owns an integrated patient-specific database. The data miner represents a user who is interested in querying the data for the purpose of performing analytical data mining activities such as classification analysis. The service provider is a public (untrusted) party that facilitates access to IT resources, i.e., storage and computational services.

The data provider desires to make its data available to authorized data miners. Due to its limited resources, the data provider outsources the database to a service provider capable of handling the responsibility of answering count queries from data miners. To prevent the disclosure of patients' sensitive information, the data provider anonymizes its data and generates a set of records that satisfy ε-differential privacy. Even though the outsourced data is anonymized, the data provider wants to protect the data against the service provider so it cannot answer queries on the data from untrusted (unauthorized) data miners. The service provider, however, should be able to process count queries from authorized data miners confidentially and return results that provide a certain privacy guarantee.

4.3.2 System Inputs

In this section we give a formal definition of the input components, namely, *differentially private data* and *user count queries*. Without loss of generality, we assume that the input data is anonymized using an ε-differential privacy model [Dwo06], although our approach supports other privacy models that produce contingency-like tables based on generalization and suppression. We choose ε-differential privacy because it provides a strong privacy guarantee while being insensitive to any specific record. We first describe how to generate ε-differentially private records from a relational

PID	Country	Job	Age	Salary	Surgery
1	USA	Engineer	30	45K	Plastic
2	USA	Lawyer	55	45K	Valve Repair
3	Canada	Engineer	30	30K	Coronary Artery
4	USA	Lawyer	45	75K	Plastic
5	Canada	Dancer	18	60K	Coronary Artery
6	USA	Writer	30	45K	Urology
7	Canada	Writer	45	75K	Valve Repair
8	Canada	Dancer	65	45K	Valve Repair

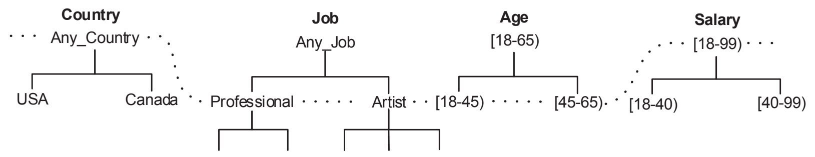


Figure 2: A raw data table D and its taxonomy trees.

data, then we explain how to transform the data using taxonomy trees, and finally we define the types of count queries the user can submit.

Differentially Private Data

In this section we review how a data provider can generate ε-differentially private records. We utilize the differentially private anonymization algorithm *(DiffGen)* [MCFY11] to maximize the data utility for classification analysis. Suppose a data provider owns an integrated patient-specific data table $D = \{A^I, A^{pr}, A^{cls}\}$, where A^I is an *explicit identifier* attribute such as *SSN* or *Name* for explicitly identifying individuals that will not be used for generating the ε-differentially private data; A^{cls} is a class attribute that contains the class value; and $\mathcal{A}^{pr}$ is a set of k predictor attributes whose values are used to predict the class attribute A^{cls}. We require the class attribute A^{cls} to be categorical, whereas the predictor attributes in A^{pr} are required to be either categorical or numerical. Furthermore, we assume that for each predictor attribute $A_i \in A^{pr}$ a taxonomy tree $\mathbb{T}^{A_i}$ is used in order to specify the hierarchy among the domain values of A_i. Figure 2 shows a raw data table D with four attributes, namely, *Country*, *Job*, *Age*, and *Salary* and the taxonomy tree for each attribute.

The data provider's objective is to generate an anonymized version $\hat{D} = \{\hat{A}^{pr}, NCount\}$ of the data table D, where $\hat{A}^{pr}$ is the set of k generalized predictor attributes, and $NCount$ is the noisy count of each record in $\hat{D}$. The objective of the data miner is to build a classifier to accurately predict the class attribute A^{cls} by submitting count queries on generalized predictor attributes $\hat{A}^{pr}$.

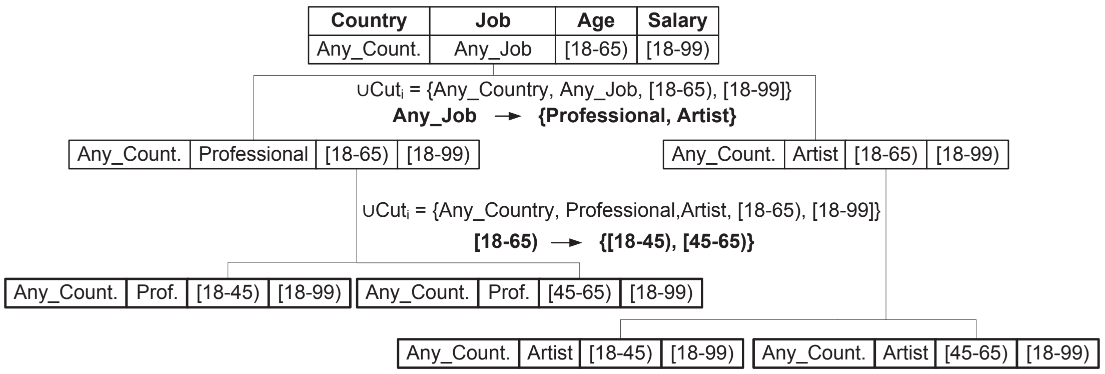

Country | Job | Age | Salary
Any_Count. | Any_Job | [18-65) | [18-99)
∪Cut$_i$ = {Any_Country, Any_Job, [18-65), [18-99]}
Any_Job → {Professional, Artist}
Any_Count. | Professional | [18-65) | [18-99)
Any_Count. | Artist | [18-65) | [18-99)
∪Cut$_i$ = {Any_Country, Professional,Artist, [18-65), [18-99]}
[18-65) → {[18-45), [45-65)}
Any_Count. | Prof. | [18-45) | [18-99)
Any_Count. | Prof. | [45-65) | [18-99)
Any_Count. | Artist | [18-45) | [18-99)
Any_Count. | Artist | [45-65) | [18-99)

Figure 3: Algorithm *DiffGen* for generating ε-differentially private data with noisy counts.

Table 3: Differentially-private data table $\hat{D}$

$\hat{Country}$	$\hat{Job}$	$\hat{Age}$	$\hat{Salary}$	$NCount$
Any_Country	Professional	[18-45)	[18-99)	4
Any_Country	Professional	[45-65)	[18-99)	2
Any_Country	Artist	[18-45)	[18-99)	1
Any_Country	Artist	[45-65)	[18-99)	5

Example 1 Consider the raw data set in Figure 2. Initially, the algorithm creates one root partition containing all the records that are generalized to ⟨ *Any_Country, Any_Job, [18-65), [18-99)* ⟩. $\cup Cut(\mathbb{T}^{A_i})$ includes {*Any_Country, Any_Job, [18-65), [18-99)*}. Let the first specialization be *Any_Job* → {*Professional*, *Artist*}. The *DiffGen* algorithm creates two new partitions under the root, as shown in Figure 3, and splits data records between them. $\cup Cut(\mathbb{T}^{A_i})$ is updated to {Any_Country, *Professional*, *Artist*, *[18-65)*, *[18-99)*}. Suppose that the next specialization is *[18-65)*→ {*[18-40)*, *[40-65)*}, which creates further specialized partitions. Finally, the algorithm outputs the equivalence groups of each leaf partition with their noisy counts as shown in Table 3. □

Input Data Transformation

We simplify the representation of the ε-differentially private records $\hat{D} = \{\hat{A}^{pr}, NCount\}$ by mapping the values of each attribute to their integer identifiers from the corresponding attribute's taxonomy tree.

Numerical Attributes. The domain of each numerical attribute $\hat{A}_i \in \hat{A}^{pr}$, consists of a set of ranges that are pair-wise disjoint and can be represented as a continuous and ordered sequence of ranges. We define an order-preserving identification function ID^{op} that assigns an integer identifier

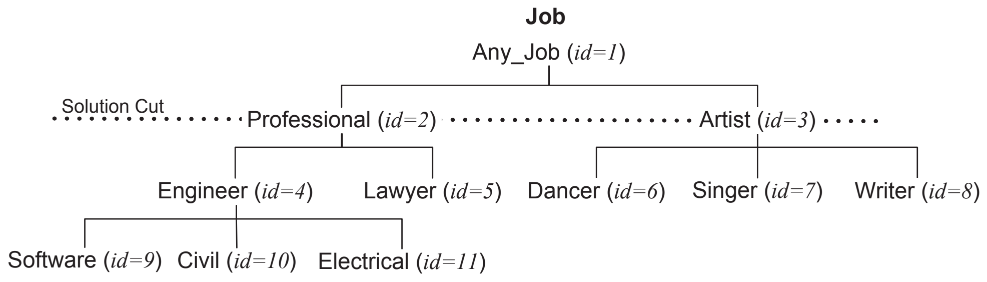


Figure 4: Taxonomy tree $\mathbb{T}^{Job}$ for attribute *Job*.

Table 4: Transformed data table $\hat{D}$

$\hat{Country}$	$\hat{Job}$	$\hat{Age}$	$\hat{Salary}$	$NCount$
1	2	1	1	4
1	2	2	1	2
1	3	1	1	1
1	3	2	1	5

to each range $r = [r^{min}, r^{max}]$ such that for any two ranges r_j and r_l, if $r_j^{max} < r_l^{min}$, then $ID^{op}(r_j) < ID^{op}(r_l)$. For example, if the domain of the generalized attribute $\hat{Age}$ is $\Omega(\hat{Age}) = \langle[18, 45), [45, 65)\rangle$, then $ID^{op}([18, 45)) = 1$ and $ID^{op}([45, 65)) = 2$.

Categorical Attributes. The domain of each categorical attribute $\hat{A}_i \in \hat{A}^{pr}$ consists of the set of values $Cut(\mathbb{T}^{A_i})$. We define a taxonomy tree identification function ID^t such that for any two nodes $v_i, v_j : v_j \neq v_i$, if v_i is a parent of v_j, then $ID^t(v_i) < ID^t(v_j)$. If v_i is the root node, then $ID^t(v_i) = 1$. Figure 4 illustrates the taxonomy tree $\mathbb{T}^{Job}$ for attribute *Job*, where each node is assigned an identification value.

Having defined the mapping functions ID^{op} and ID^t, we now transform the ε-differentially private records $\hat{D}$ by mapping the values in the domain of each attribute to their identifiers. That is, for each numerical attribute $\hat{A}_i \in \hat{A}^{pr}$, we map each range $r \in \Omega(\hat{A}_i)$ to its corresponding identification value $ID^{op}(r)$. Similarly, for each categorical attribute $\hat{A}_i \in \hat{A}^{pr}$, we map each value in $v \in \Omega(\hat{A}_i)$ to its identification value from the taxonomy tree $ID^t(v)$. Table 4 shows the differentially private data $\hat{D}$ after the transformation.

User Count Queries

The goal of the data miners is to build a classifier based on the noisy count of a query over the generalized attributes $\hat{A}^{pr}$. Therefore, they submit count queries to be processed on the ε-differentially private data $\hat{D}$ and expect to receive a noisy count as a result to each submitted query. We denote by *user count query* any data mining's count query, and it is formally defined as follows:

Definition 4 ***(User Count Query.)*** A *user count query* u over $\hat{D}$ is a conjunction of predicates $\mathcal{P}_1 \wedge ... \wedge \mathcal{P}_m$ where each predicate $\mathcal{P} = (\hat{A}_i \, Op \; s_i)$ expresses a single criterion such that $\hat{A}_i \in \hat{A}^{pr}$, Op is a comparison operator, and s_i is an operand. If $\hat{A}_i$ is a categorical attribute, then Op corresponds to the equality operator " $=$ " and s_i is a value from the taxonomy tree $\mathbb{T}^{A_i}$. If $\hat{A}_i$ is a numerical attribute, then s_i is a numerical range $[s_i^{min}, s_i^{max}]$ such that if $s_i^{min} = s_i^{max}$ then $Op \in \{>, \geq, <, \leq, =\}$; otherwise, Op is the equal operator $(=)$. □

In general, a user count query u can be either *exact*, *specific*, or *generic* depending on whether it corresponds to an exact record (equivalence class), or whether it partially intersects with one or more records in the ε-differentially private data $\hat{D}$. Note that both *specific* and *generic* queries correspond to *range queries* in the literature. The following is a formal definition of each type of a user count query.

Definition 5 ***(Exact User Count Query.)*** A user count query u is *exact* if for each predicate $\mathcal{P} = (\hat{A}_i \, Op \; s_i) \in u$, $s_i \in \Omega(\hat{A}_i)$. □

Definition 6 ***(Specific User Count Query.)*** A user count query u is *specific* if for each predicate $\mathcal{P} = (\hat{A}_i \, Op \; s_i) \in u$:

1. If $\hat{A}_i$ is categorical, then $s_i \in \Omega(\hat{A}_i)$.

2. If $\hat{A}_i$ is numerical, then $s_i \in \Omega(\hat{A}_i)$ or there exists exactly one range $r \in \Omega(\hat{A}_i)$ where $s_i \cap r \neq \phi$ and $s_i \neq r$. □

Definition 7 ***(Generic User Count Query.)*** A user count query u is *generic* if for each predicate $\mathcal{P} = (\hat{A}_i \, Op \; s_i) \in u$:

1. If $\hat{A}_i$ is categorical, then $s_i \in \mathbb{T}^{A_i}$.
2. If $\hat{A}_i$ is numerical, then $\exists\, r_j, r_l \in \Omega(\hat{A}_i)$ such that $s_i \cap r_j \neq \phi$, $s_i \cap r_l \neq \phi$, and $r_j \neq r_l$. $\square$

Example 2 The following are examples of user count queries over the ε-differentially private data $\hat{D}$ presented in Table 3:

Exact: $u_1 = (\hat{Job} = \text{“}Artist\text{”}) \wedge (\hat{Age} = [45 - 65))$

Specific: $u_2 = (\hat{Job} = \text{“}Artist\text{”}) \wedge (\hat{Age} = [50 - 57))$

Generic: $u_3 = (\hat{Job} = \text{“}Lawyer\text{”}) \wedge (\hat{Age} = [30 - 70))$

Observe that the queries conform neither to the structure nor to the data in $\hat{D}$. That is, attributes $\hat{Country}$ and $\hat{Salary}$ are missing, the value “Lawyer” is not in the domain $\Omega(\hat{Job})$, and the range $[30, 70]$ spans beyond the values covered by all ranges in $\Omega(\hat{Age})$. All these issues will be addressed in section 4.4.3 when the data miner submits her user count query for preprocessing. $\square$

4.3.3 Adversarial Model

SecDM consists of three parties: *data provider* (data bank), *data miner*, and *service provider* (cloud). In our security analysis, the adversary can statically corrupt, in *honest-but-curious (HBC)* [KMR] fashion, the service provider or the data miner, but not both. The service provider adversary tries to gain access to the contents of the anonymized data, and during query execution tries to infer information about the count queries and their results. On the other hand, the data miner adversary tries to link sensitive information to patients by attempting to gain information about the anonymized records identified by each of her queries, their count values, and the percentage of each query count. We assume the computational power of each adversary is bounded by a polynomial size circuit. We also assume that a protocol is in place to provide secure pair-wise communications between parties in the SecDM framework.

4.3.4 Problem Statement

Given ε-differentially private data $\hat{D}$, the objective is to design a framework for outsourcing $\hat{D}$ to an untrusted service provider P that can answer exact, specific, and range count queries from authorized data miners on $\hat{D}$. The framework must provide three levels of security: (1) *data confidentiality*, where $\hat{D}$ is stored in an encrypted form such that no useful information can be disclosed from $\hat{D}$ by unauthorized parties; (2) *confidential query processing*, where P is capable of processing the queries on $\hat{D}$ for classification analysis without inferring information about the queries or the underlying anonymized data; and (3) *privacy preservation*, where the result of each query provides a certain privacy guarantee.

4.4 Solution: SecDM Framework

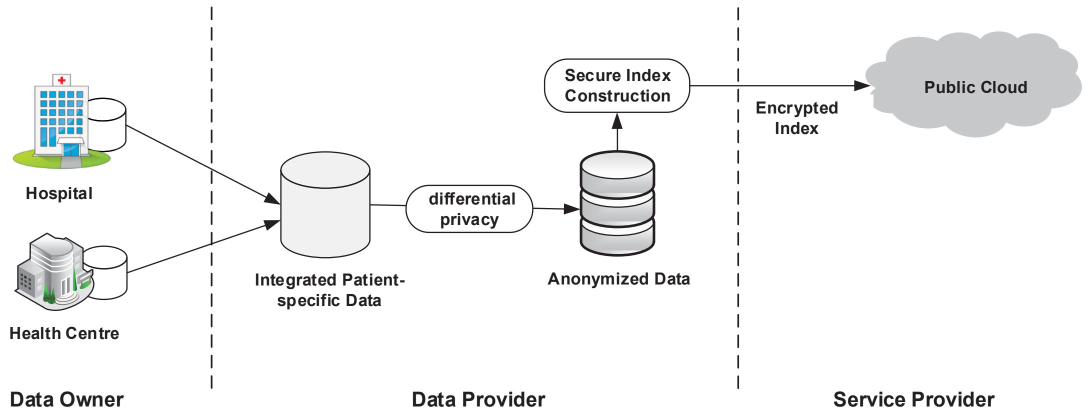
In this section we first present an overview of our proposed privacy-preserving framework for confidential query processing on ε-differentially private data in the cloud, and then we elaborate on the key steps of the algorithm, including constructing a secure index, securing the data for outsourcing, and executing count queries while preserving their privacy.

4.4.1 Solution Overview

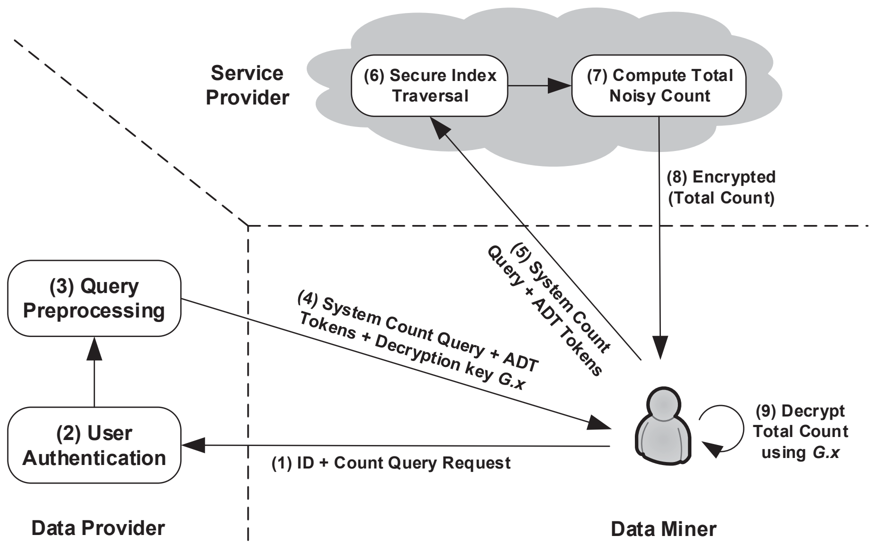
The objective of our solution is to provide a secure framework that enables data providers to outsource their ε-differentially private data $\hat{D}$ to a service provider (public cloud) such that the confidentiality of the outsourced data is protected, while the service provider is capable of securely answering count queries from authorized data miners without being able to infer any information about the queries and their results, as illustrated in Figure 5. Our framework consists of five algorithms:

Algorithm 2 - Secure Index Construction (*buildIndex*): For an efficient data retrieval, a secure kd-tree index is constructed over all categorical and numerical attributes in $\hat{D}$, where each non-leaf node is encrypted using the ACP-ABE scheme $\mathbb{A}$.

Algorithm 3 - Leaf Node Construction (*constLeafNode*): Utilized by the *buildIndex*



(a) The setup phase, where the data provider anonymizes the integrated patient-specific data and generates a *k*d-tree index over the anonymized data for efficient search and retrieval and uploads the index to the cloud.


Service Provider
(6) Secure Index Traversal
(7) Compute Total Noisy Count
(8) Encrypted (Total Count)
(5) System Count Query + ADT Tokens
(3) Query Preprocessing
(4) System Count Query + ADT Tokens + Decryption key G.x
(2) User Authentication
(1) ID + Count Query Request
(9) Decrypt Total Count using G.x
Data Provider
Data Miner

(b) The query processing phase, where authorized users interested in mining the patient-specific data obtain system count queries from the data provider and then submit the queries to the cloud for processing. The cloud confidentially process the queries and sends the privacy-preserving results back to the users.

Figure 5: SecDM: A Privacy-preserving framework for confidential count query processing on the cloud.

procedure to construct the leaf nodes in the kd-tree index. Each leaf node contains a noisy count encrypted using Exponential ElGamal and a set of tags to be utilized during query execution for determining the exact percentage that should be used of the noisy count of each reported leaf node.

Algorithm 5 - Query Preprocessing (*qPreprocess*): A user (data miner) desiring to query the outsourced data first submits a query u to the data provider that preprocess the query and sends back three components: an encrypted version of the query in the form of a secret key SK_u using anonymous ciphertext-policy attribute based encryption scheme $\mathbb{A}$, a set of ADT tokens for producing accurate count results, and a decryption key $\mathbb{G}.x$. The user then submits the encrypted query SK_u and the ADT tokens to the service provider.

Algorithm 7 - Index Traversal (*traverseIndex*): The service provider utilizes the user's secret key SK_u in order to securely traverse the index tree and determine the leaf nodes satisfying SK_u.

Algorithm 8 - Total Count Computation (*compTCount*): Using the set of ADT tokens, the service provider computes the percentage of the noisy count of each reported leaf node, homomorphically add all counts together, and then sends the encrypted result to the user.

4.4.2 Secure Index Construction

Given the ε-differentially private data $\hat{D}$ with k_c categorical attributes and k_n numerical attributes, the data provider constructs an encrypted index on all attributes in $\hat{D}$ in order to support efficient and secure processing of multi-dimensional range count queries over the k-dimensional data, where $k = k_c + k_n$. That is, it constructs a balanced kd-tree [Ben75] index, where every internal (non-leaf) node is a k-dimensional node that splits the space into two half-spaces, and each leaf node stores a noisy count corresponding to a record in $\hat{D}$.

The kd-tree index is constructed with the procedure *Secure Index Construction* (*buildIndex*) presented in Algorithm 2. *BuildIndex* is a recursive procedure that has four input parameters: $\hat{D}$, depth i, PK, and y. The first input parameter $\hat{D}$ is the set of records for which the kd-tree will

ALGORITHM 2: *buildIndex*: **Secure Index Construction**

```
Input: Shuffled $\varepsilon$-differentially private data $\hat{D}$
Input: split dimension $i$
Input: ACP-ABE public key $PK$
Input: Exponential ElGamal public key $y$
Output: $k$d-tree index $T$
1: if $|\hat{D}| = 1$ then
2:     $LeafNode \leftarrow constLeafNode(\hat{D}, y)$;
3:     return $LeafNode$;
4: end if
5: $cut \leftarrow median(\hat{A}_i, \hat{D})$;
6: $split(\hat{D}, \hat{A}_i, cut, \hat{D}_1, \hat{D}_2)$;
7: $v_{left} \leftarrow buildIndex(\hat{D}_1, (i+1) \, mod \, k, PK, y)$;
8: $v_{right} \leftarrow buildIndex(\hat{D}_2, (i+1) \, mod \, k, PK, y)$;
9: create empty node $v$;
10: $v.split\_dim \leftarrow \hat{A}_i$ ; $v.split\_value \leftarrow cut$;
11: $v.lc \leftarrow v_{left}$ ; $v.rc \leftarrow v_{right}$;
12: $v.genCT(PK)$;
13: return $k$d-tree index $T$;
```

be constructed, where each record represents a point in the k-dimensional space. The columns in $\hat{D}$ are *shuffled a priori* to randomize the order of the attributes. The second input parameter i represents the depth of the recursion that determines the split dimension. It ranges between 1 and k, where 1 is the initial value. The third input parameter PK is the public key of the anonymous ciphertext-policy attribute based encryption scheme $\mathbb{A}$, which will be used to secure each internal node in the index tree. To generate this key a security parameter λ is passed to the setup algorithm: $\mathbb{A}.\mathsf{Setup}(1^{\lambda}) \Rightarrow (PK, MSK)$. The last parameter y is the public key of the Exponential ElGamal scheme used to encrypt the noisy counts in the leaf nodes. The function *median* (Line 5) determines the median value of the domain $\Omega(\hat{A}_i)$, where $\hat{A}_i$ is an attribute from $\hat{D}$. The function *split* (Line 6) then uses a hyperplane that passes through the median value in order to split $\hat{D}$ into two subsets of records, $\hat{D}_1$ and $\hat{D}_2$. Note that the *median* value is chosen for splitting to ensure a balanced tree where each leaf node is about the same distance from the root of the tree. *BuildIndex* calls itself (Lines 7-8) using $\hat{D}_1$ and $\hat{D}_2$ as inputs in order to determine the left and right children nodes respectively. When the procedure terminates (Line 13), it returns the kd-tree index T. Next, we will discuss how internal nodes (Lines 9-12) and leaf nodes (Line 2) are constructed.

Internal Nodes Construction

Each internal (non-leaf) node v in the kd-tree index corresponds to one dimension (attribute) $\hat{A}_i \in \hat{D} : 1 \leq i \leq k$ of the k-dimensional space, where the splitting hyperplane is perpendicular to the axis of dimension $\hat{A}_i$, and the splitting value *cut* is determined by the median function (Line 4). Node v has two child nodes, namely, lc and rc, where all records containing values smaller or equal to the *cut* value with regard to $\hat{A}_i$ will appear in the left subtree, whose root is $v.lc$, and all records containing values greater than the *cut* value with regard to $\hat{A}_i$ will appear in the right subtree, whose root is $v.rc$. Furthermore, node v consists of two ciphertexts, $v.CT_{left}$ and $v.CT_{right}$, where the encrypted message in $v.CT_{left}$ is a pointer (Ptr) to the child node $v.lc$, and the encrypted message in $v.CT_{right}$ is a pointer to the child node $v.rc$. The intuition is as follows: The service provider must use the key SK_u provided by the user to be allowed to securely traverse the kd-tree index and compute the answer to the user query u. The structure of SK_u and how it is built is discussed in Section 4.4.3. At any node v in the kd-tree, if SK_u satisfies the access structure of the ciphertext $v.CT_{left}$, the ciphertext is decrypted and a pointer to the child node $v.CT_{left}$ is obtained. Similarly, if SK_u satisfies the access structure of $v.CT_{right}$, the ciphertext is decrypted and a pointer to $v.CT_{right}$ is obtained. If SK_u satisfies both access structures, then two pointers are obtained, indicating that both left and right subtrees must be traversed.

Access Structure. The ciphertexts in each node are generated using the anonymous ciphertext-policy attribute based encryption scheme $\mathbb{A}$, where each ciphertext has an access structure W. Each numerical attribute $\hat{A}_i \in \hat{D}$ is represented in the access structure of a ciphertext by two attributes, $\hat{A}_i^{min}$ and $\hat{A}_i^{max}$, where $\Omega(\hat{A}_i^{min}) = \Omega(\hat{A}_i^{max}) = \Omega(\hat{A}_i)$. On the other hand, each categorical attribute is mapped to one attribute in the access structure of a ciphertext. Below is the formal definition of an access structure of a ACP-ABE ciphertext.

Definition 8 ***(Ciphertext Access Structure*** W***.)*** Given ε-differentially private data $\hat{D}$ and a node v from the kd-tree index over $\hat{D}$, the access structure of a ciphertext of v is the conjunction

$W = [W_{\hat{A}_1} \wedge ... \wedge W_{\hat{A}_i} \wedge ... \wedge W_{\hat{A}_k}]$. If $\hat{A}_i$ is a categorical attribute, then $W_{\hat{A}_i}$ corresponds either to the wildcard character " $*$ " or to a disjunction of values from $\Omega(\hat{A}_i)$, where "$W_{\hat{A}_i} = *$" means that attribute $\hat{A}_i$ should be ignored. If $\hat{A}_i$ is a numerical attribute, then $W_{\hat{A}_i} = W_{\hat{A}_i^{min}} \wedge W_{\hat{A}_i^{max}}$, where $W_{\hat{A}_i^{min}}$ and $W_{\hat{A}_i^{max}}$ each corresponds to either the wildcard character " $*$ " or a disjunction of values from $\Omega(\hat{A}_i)$. $\square$

Note that for a given node v, the access structure of the left and right ciphertexts is mainly concerned with the splitting dimension $v.split\_dim$, and the split value $v.split\_value$ over $\hat{D}_1$ or $\hat{D}_2$, where $\hat{D}_1, \hat{D}_2 \subseteq \hat{D}$. If $v.split\_dim$ is a categorical attribute $\hat{A}_i$, then $W_{\hat{A}_i}$ in the access structure of $v.CT_{left}$ should correspond to the disjunction of all values $val \in \{\Omega(\hat{A}_i) \cup \{1\}\}$ such that $val \leq v.split\_value$ and for $val = 1$, where 1 represents "Any" value. Similarly, $W_{\hat{A}_i}$ in the access structure of $v.CT_{right}$ should correspond to the disjunction of all values $val \in \{\Omega(\hat{A}_i) \cup \{1\}\}$ such that $val > v.split\_value$ or for $val = 1$. On the other hand, if $v.split\_dim$ is a numerical attribute $\hat{A}_i$, then $W_{\hat{A}_i^{min}}$ in the access structure of $v.CT_{left}$ should correspond to the disjunction of all values $val \in \Omega(\hat{A}_i)$ for all $val \leq v.split\_value$, and $W_{\hat{A}_i^{max}}$ in the access structure of $v.CT_{right}$ should correspond to the disjunction of all values $val \in \Omega(\hat{A}_i)$ such that $val > v.split\_value$. Regardless of whether $\hat{A}_i$ is categorical or numerical, all values in $\{W \setminus W_{\hat{A}_i}\}$ should correspond to " $*$ ".

Example 3 Given $\hat{D}$ from Table 4, and given node v from kd-tree index:

a) If $v.split\_dim = Job$ (categorical) and $v.split\_value = 2$, then the access structure of the left and right ciphertexts can be represented as follows:

$W_L = (Country = *) \wedge (Job = 1 \vee Job = 2) \wedge (Age_{min} = *) \wedge (Age_{max} = *) \wedge (Salary_{min} = *) \wedge (Salary_{max} = *)$.

$W_R = (Country = *) \wedge (Job = 1 \vee Job = 3) \wedge (Age_{min} = *) \wedge (Age_{max} = *) \wedge (Salary_{min} = *) \wedge (Salary_{max} = *)$.

b) If $v.split\_dim = Age$ (numerical) and $v.split\_value = 1$, then the access structure of the left and right ciphertexts are:

$W_L = (Country = *) \wedge (Job = *) \wedge (Age_{min} = 1) \wedge (Age_{max} = *) \wedge (Salary_{min} = *) \wedge (Salary_{max} =$

$*)$.

$W_R = (Country = *) \wedge (Job = *) \wedge (Age_{min} = *) \wedge (Age_{max} = 2) \wedge (Salary_{min} = *) \wedge (Salary_{max} = *)$. $\square$

In procedure *buildIndex* presented in Algorithm 2, each internal node v is created after determining its children nodes V_{left} and v_{right} (Lines 9-12), where function *genCT* is responsible for creating the left ciphertext CT_{left} and the right ciphertext CT_{right} of the node by calling twice the ACP-ABE algorithm $\mathbb{A}.\mathsf{Enc}()$ and passing as parameters the public key PK of $\mathbb{A}$, a pointer to the child node to be encrypted, and the values in the access structure (without the attribute names):

$v.CT_{left} \leftarrow \mathbb{A}.\mathsf{Enc}(PK, Ptr(v.lc), W_L)$;

$v.CT_{right} \leftarrow \mathbb{A}.\mathsf{Enc}(PK, Ptr(v.rc), W_R)$;

For each attribute $\hat{A}_i$ in W that is assigned a wildcard, e.g. $(Country = *)$, $\mathbb{A}.\mathsf{Enc}()$ generates a random (*mal-formed*) group elements $[C_{i,j,1}, C_{i,j,2}]$ for each value in $\Omega(\hat{A}_i)$. On the other hand, for each attribute in W assigned specific values, e.g. $(Job = 1 \vee Job = 2)$, $\mathbb{A}.\mathsf{Enc}()$ generates a *well-formed* group elements for each value specified, i.e. for value 1 and for value 2, and random group elements for each remaining value in $\Omega(Job)$. As a result, all ciphertexts CT generated by $\mathbb{A}.\mathsf{Enc}()$ in the kd-tree index contain the same number of group elements regardless of the access structure.

Leaf Nodes Construction

In procedure *buildIndex*, as the multi-dimensional space is being recursively partitioned a leaf node is created whenever the number of the records being partitioned reaches 1 (Lines 1-2). Procedure *Leaf Node Construction* (*constLeafNode*), presented in Algorithm 3, is responsible for generating the leaf nodes. It takes as input a ε-differentially private record R and Exponential ElGamal's public key y and outputs a leaf node l. After creating an empty node l (Line 1), the noisy count of record R is encrypted using Exponential ElGamal encryption scheme $\mathbb{G}$ and stored in node l (Line 2). We choose the Exponential ElGamal cryptosystem due to its additive homomorphism property, which

ALGORITHM 3: *constLeafNode*: **Leaf Node Construction**

```
Input: $\varepsilon$-differentially private record $R$
Input: Exponential ElGamal's public key $y$
Output: leaf node $l$
1: create empty node $l$;
2: $l.NCount \leftarrow \mathbb{G}.\mathsf{Enc}(R.NCount, y, r)$;
3: for each numerical attribute $\hat{A}_i \in \hat{D}$ do
4:     $l \leftarrow genTAG(R.\hat{A}_i)$;
5: end for
6: return $l$;
```

ALGORITHM 4: *indexUpload*: ***k*d-Tree Index Upload**

```
1: The Data Provider submits the $k$d-tree index $T$ to the Service Provider;
2: The Service Provider receives $T$;
```

allows for homomorphically adding encrypted noisy counts together in an efficient way.

For each numerical range value $R.\hat{A}_i$ in R, a deterministic and hiding commitment function $genTAG()$ is utilized to commit $R.\hat{A}_i$ and randomly generate a unique tag (Line 3-4). Applying $genTAG()$ to the same value will always generate the same tag (deterministic). Moreover, the correspondence between each tag and its value is kept secret (hiding). As we will see in Section 4.4.3, using a deterministic function to generate tags for the numerical range values enables the service provider during query execution to compute the exact percentage of the noisy count of each reported leaf node with respect to the query being processed.

Once the kd-tree index T has been created, it is submitted to the service provider according to Algorithm 4. While Algorithm 4 is trivial, it is required in Section 4.5.3 to prove by simulation the security of the framework.

4.4.3 Confidential Query Processing

In this section, we illustrate how user count queries are executed in order to determine their exact ε-differentially private answers while preserving the confidentiality of the data as well as the queries. First, we explain how the user query is preprocessed and transformed into a system query. Next, we discuss how the service provider securely traverses the kd-tree index, computes the total count, and then sends the result back to the user.

Query Preprocessing

Upon the receipt of a user's count query u, the data provider first transforms u into a conjunction of subqueries that specify a single-value equality condition over each attribute $\hat{A}_i \in \hat{D}$. Next, it generates a *system count query* SK_u using algorithm $\mathbb{A}.\mathsf{KeyGen}()$ from ACP-ABE scheme. If an attribute $\hat{A}_i$ in $\hat{D}$ is not specified by the user in u, then it will be considered in SK_u as if the user is asking for $(\hat{A}_i = *)$. The following is the formal definition of a system count query:

Definition 9 ***(System Count Query.)*** Given ε-differentially private data $\hat{D}$ with k attributes and a user query $u = \mathcal{P}_1 \wedge ... \wedge \mathcal{P}_m \,|\, \mathcal{P} = (\hat{A}_i \, Op \; s_i)$, a system count query over $\hat{D}$ is a ACP-ABE user's secret key SK_u representing k subqueries $\{q_{\hat{A}_1}, ..., q_{\hat{A}_k}\}$ such that:

- If $\hat{A}_i$ is a categorical attribute, then $\hat{A}_i$ is represented in SK_u as a tuple of group elements $[D_{i,0}, D_{i,1}, D_{i,2}]$
- If $\hat{A}_i$ is a numerical attribute, however, it is represented in SK_u as two tuples of group elements $[D^{min}_{i,0}, D^{min}_{i,1}, D^{min}_{i,2}]$ and $[D^{max}_{i,0}, D^{max}_{i,1}, D^{max}_{i,2}]$, where each tuple corresponds to the minimum and maximum bound of the range subquery $q_{\hat{A}_i}$, respectively. □

The total number of group element tuples in a system query SK_u is: $|SK_u| = k_c + 2 \times k_n$, where k_c and k_n are the number of categorical and numerical attributes in $\hat{D}$. $|SK_u|$ is independent of the user query u. We refer the reader to Section 4.2.1 for more details on how an ACP-ABE secret key is generated.

Procedure *Query Preprocessing (qPreprocess)* presented in Algorithm 5 illustrates how a system count query SK_u is constructed based on a user's count query u. Once the user has been authenticated successfully using user identification token UIT (Line 2), the next step is to determine the attribute-value pairs in SK_u. For each *categorical* attribute $\hat{A}_i \in \hat{D}$, if predicate $(\hat{A}_i, Op, s_i)$ exists in the user count query u and s_i is in the domain of $\hat{A}_i$, then the subquery $(\hat{A}_i, s_i.ID)$ is added to q,

ALGORITHM 5: *qPreprocess*: **Query Preprocessing**

```
Input: $\varepsilon$-differentially private data $\hat{D}$
Output: system count query $SK_u$
Output: set of attribute distribution tokens $\mathcal{N}$
1: The Data Provider receives user identification token $UIT$ and user count query $u$ from Data Miner
2: if user authentication is successful using $UIT$ then
3:   $q \leftarrow \{\}$; $ADT \leftarrow \{\}$;
4:   for each categorical attribute $\hat{A}_i \in \hat{D}$ do
5:     if $(\hat{A}_i, Op, s_i) \in u$ and $s_i \in \Omega(\hat{A}_i)$ then
6:       $q \leftarrow q \cup (\hat{A}_i, s_i.ID)$;
7:     else if $(\hat{A}_i, Op, s_i) \in u$ and $s_i \notin \Omega(\hat{A}_i)$ then
8:       $n \leftarrow findSCS(s_i)$;
9:       $q \leftarrow q \cup (\hat{A}_i, n.ID)$;
10:    else if $(\hat{A}_i, Op, s_i) \notin u$ then
11:      $q \leftarrow q \cup (\hat{A}_i, 1)$;
12:    end if
13:  end for
14:  for each numerical attribute $\hat{A}_i \in \hat{D}$ do
15:    if $(\hat{A}_i, Op, s_i) \in u$ then
16:      $(v_{i,1}, v_{i,2}) \leftarrow compMinMax(\Omega(\hat{A}_i), s_i, Op_i)$;
17:      $q \leftarrow q \cup (\hat{A}_i^{min}, v_{i,1}) \cup (\hat{A}_i^{max}, v_{i,2})$;
18:      $\mathcal{N} \leftarrow \mathcal{N} \cup genADT(\Omega(\hat{A}_i), v_{i1}, v_{i2})$;
19:    else
20:      $q \leftarrow q \cup (\hat{A}_i^{min}, 1) \cup (\hat{A}_i^{max}, range_{max})$;
21:    end if
22:  end for
23:  $SK_u \leftarrow \mathbb{A}.\mathsf{KeyGen}(MSK, q)$;
24:  return $SK_u$, $\mathcal{N}$;
25: end if
```

ALGORITHM 6: *queryRequest*: **System Count Query Request**

```
1: The Data Provider sends the following to the Data Miner:
     • System count query $SK_u$ corresponding to user count query $u$
     • Set of attribute distribution tokens $\mathcal{N}$
     • Exponential ElGamal decryption key $x$
2: The Data Miner receives $SK_u$, $\mathcal{N}$, and $x$ from Data Provider;
3: The Data Miner sends $SK_u$ and $\mathcal{N}$ to Service Provider;
4: The Service Provider receives $SK_u$ and $\mathcal{N}$ from Data Miner;
```

where $s_i.ID$ is the identifier of the categorical value s_i in $\hat{A}_i$'s taxonomy tree $\mathbb{T}^{\hat{A}_i}$ (Lines 5-6); otherwise, if s_i is not in the domain of $\hat{A}_i$, then function $findSCS(s_i)$ (Line 8) is utilized to determine the position of s_i in $\hat{A}_i$'s taxonomy tree with regard to the solution cut. If s_i is below the solution cut, then there exists exactly one node n on the path from s_i to the root, such that $n \in \Omega(\hat{A}_i)$. We call such a node the *Solution Cut Subsumer (SCS)* of s_i, and the subquery $(\hat{A}_i, n.ID)$ is then added to q (Line 9). If s_i is above the solution cut or u does not have any predicate that corresponds to a categorical attribute $\hat{A}_i \in \hat{D}$, then the subquery $(\hat{A}_i, 1)$ is added to q (Lines 10-11), where 1 means "ANY" value of $\hat{A}_i$ corresponding to the root node of $\hat{A}_i$'s taxonomy tree $\mathbb{T}^{\hat{A}_i}$.

On the other hand, if $\hat{A}_i$ is a *numerical* attribute and predicate $(\hat{A}_i, Op, s_i)$ exists in u, then the values $v_{i,1}$ associated with $\hat{A}_i^{min}$ and $v_{i,2}$ associated with $\hat{A}_i^{max}$ are determined by the function *compMinMax* (Line 15). When Op is the equal operator $(=)$, if s_i is a single value, then $v_{i,1} = v_{i,2} = \mathit{Range}(s_i)$, where $\mathit{Range}(s_i)$ is a function that returns the identifier of the range in $\Omega(\hat{A}_i)$ containing s_i; otherwise, if s_i is a range, then $v_{i,1} = \mathit{Range}(\mathit{Lowerbound}(s_i))$ and $v_{i,2} = \mathit{Range}(\mathit{Upperbound}(s_i))$. If $Op = \text{“} \geq \text{”}$, then $v_{i,1} = \mathit{Range}(s_i)$ and $v_{i,2}$ is the identifier of the highest range in $\Omega(\hat{A}_i)$. Conversely, if $Op = \text{“} \leq \text{”}$, then $v_{i,1} = 1$ and $v_{i,2} = \mathit{Range}(s_i)$. If predicate $(\hat{A}_i, Op, s_i)$ does not exist in u for numerical attribute $\hat{A}_i$ (Lines 19-20), then $v_{i,1} = 1$ and $v_{i,2}$ is the identifier of the highest range $range_{max} \in \Omega(\hat{A}_i)$.

Example 4 Given Table 3 and Table 4, the following are three different users' queries and their corresponding subqueries in the access structure of the system count query:

a) $u = (Age = 50) \Rightarrow q = (Age^{min}, 2), (Age^{max}, 2)$.

b) $u = (Age = [40 - 70]) \Rightarrow q = (Age^{min}, 1), (Age^{max}, 2)$.

c) $u = (Age \leq 35) \Rightarrow q = (Age^{min}, 1), (Age^{max}, 1)$. □

Function *genADT* (Line 18) is used to generate *attribute distribution tokens (ADT)* for each numerical attribute $\hat{A}_i$ from $\hat{D}$. Two ADT tokens, ADT_{min} and ADT_{max}, are created for each numerical attribute for the purpose of computing the percentages of the noisy counts of the reported leaf nodes upon query execution in order to determine the final answer (total count) of the query. Each ADT token consists of two parts: *tag* and *value*. Assuming that r is the range for which the ADT token is constructed, then $ADT.tag = genTAG(r)$ and $ADT.value$ is the percentage of the partial overlap between query u and range r.

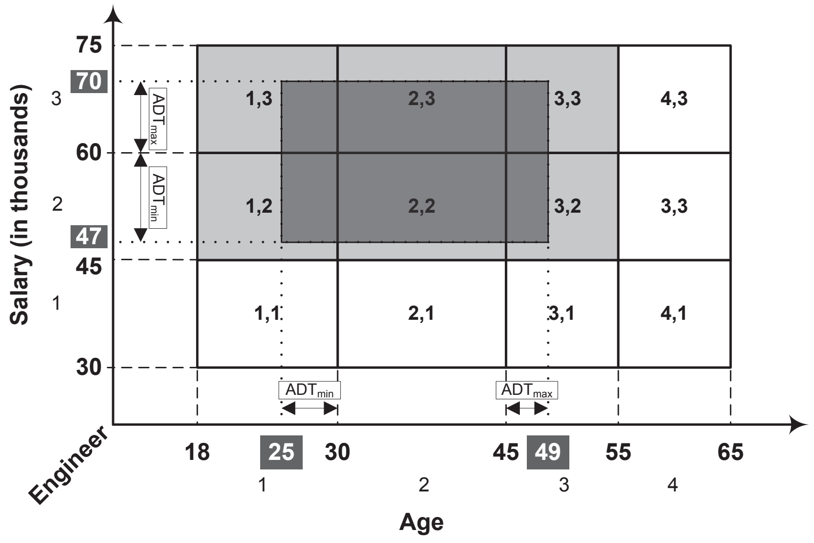


Figure 6: ADT query overlap.

Example 5 Assume that in ε-differentially private data $\hat{D}$, $\Omega(\hat{Age}) = \{[18-30), [30-45), [45-55), [55-65)\}$, $\Omega(\hat{Salary}) = \{[30-45), [45-60), [60-70)\}$, and user count query $u = (Country = \text{“}US\text{”}) \wedge (Job = \text{“}Engineer\text{”}) \wedge (Age = [25-49]) \wedge (Salary = [47-70])$. Figure 6 illustrates the equivalence classes of all records (numbered from 1,1 to 4,3), the query u (dark gray rectangle), and the set of leaf nodes identified by u (six light gray rectangles). The range $Age = [25-49]$ spans over three ranges: $[18-30), [30-45)$, and $[45-55)$. Since $[25-49]$ fully spans over $[30-45)$, no ADT token is required for $[30-45)$. However, since $[25-49]$ partially overlaps with ranges $[18-30)$ and $[45-55)$, ADT_{min} and ADT_{max} should be created. For range value $[18-30)$, $ADT_{min}.tag = genTAG([18-30))$ and $ADT_{min}.value = \frac{30-25}{30-18} = 42\%$. Similarly, for range value $[45-55]$, $ADT_{max}.tag = genTAG([45-55))$ and $ADT_{max}.value = \frac{50-45}{55-45} = 50\%$. On the other hand, $Salary = [47-70]$ partially overlaps with ranges $[45-60)$ and $[60-75)$ and ADT_{min} and ADT_{max} must be created. For range value $[45-60)$, $ADT_{min}.tag = genTAG([45-60))$ and $ADT_{min}.value = \frac{60-47}{60-45} = 87\%$. Similarly, for range value $[60-75]$, $ADT_{max}.tag = genTAG([60-75))$ and $ADT_{max}.\ value = \frac{70-60}{75-60} = 67\%$. $\square$

Once the set of attribute-value pairs q have been determined, the system count query SK_u is then generated by encrypting q with ACP-ABE master secret key MSK using algorithm $\mathbb{A}$.KeyGen (Line 23). Next, the data provider sends the following back to the user: secret key SK_u, the set of

ALGORITHM 7: *traverseIndex*: k**d-tree Index Traversal**

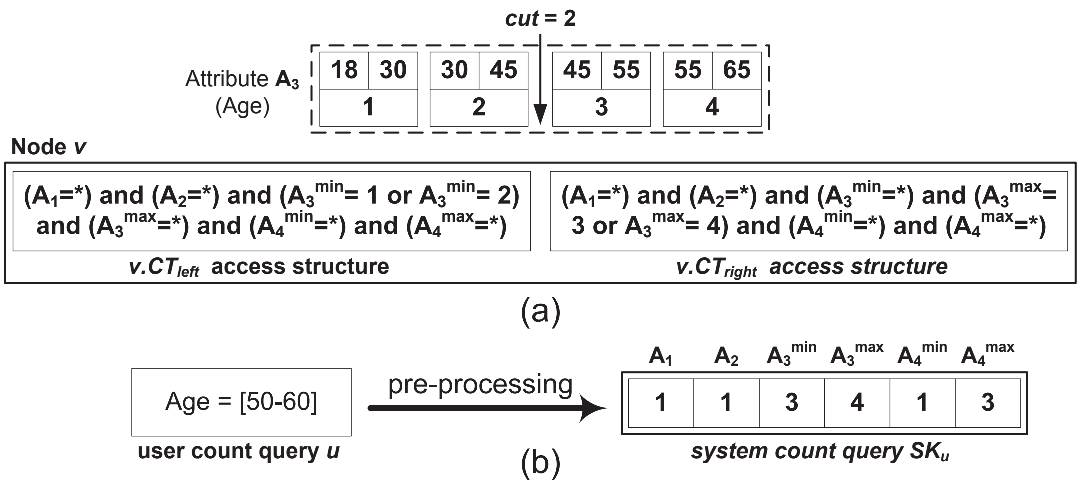
```
Input: $k$d-tree index root node $v$
Input: system count query $SK_u$
Output: set of leaf nodes $\mathcal{R}$
1: if $v$ is a leaf node then
2:     return $v$;
3: else
4:     if $\mathbb{A}.\mathsf{Dec}(v.CT_{left}, SK_u)$ then
5:         $\mathcal{R} \leftarrow \mathcal{R} \cup traverseIndex(v.lc, SK_u)$;
6:     end if
7:     if $\mathbb{A}.\mathsf{Dec}(v.CT_{right}, SK_u)$ then
8:         $\mathcal{R} \leftarrow \mathcal{R} \cup traverseIndex(v.rc, SK_u)$;
9:     end if
10: end if
11: return $\mathcal{R}$;
```

ADT tokens $\mathcal{N}$, and ElGamal decryption key $\mathbb{G}.x$ that will be used eventually to decrypt the final result of the query.

kd-tree Index Traversal

To execute a query u on $\hat{D}$, the data miner sends a system count query SK_u and a set of ADT tokens $\mathcal{N}$ to the service provider. The service provider uses the secret key SK_u to securely traverse the kd-tree index and identify the set of leaf nodes satisfying u, while it uses $\mathcal{N}$ to adjust the noisy count of each identified leaf node in order to compute an accurate final answer to the query.

Procedure *kd-tree Index Traversal (traverseIndex)* presented in Algorithm 7 illustrates how the tree is traversed recursively to answer queries. It takes two input parameters: the root node v of the kd-tree index and a system count query SK_u. If v is an internal node, then the algorithm attempts to decrypt the left ciphertext $v.CT_{left}$ and the right ciphertext $v.CT_{right}$ by separately applying the decryption function Dec from $\mathbb{A}$, with the decryption key SK_u, in order to determine whether it needs to traverse the left subtree, right subtree, or both. If the values of the attributes associated with SK_u satisfy the access structure of $v.CT_{left}$, then the decryption of $v.CT_{left}$ is successful and the procedure $traverseIndex$ calls itself while passing the left child node $v.lc$ as input parameter (Line 4-5). Similarly, if the values of the attributes associated with SK_u satisfy the access structure of $v.CT_{right}$, then the decryption is successful and the procedure $traverseIndex$ calls itself while


cut = 2
Attribute A3 (Age)
18 30
1
30 45
2
45 55
3
55 65
4
Node v
(A1=*) and (A2=*) and (A3min= 1 or A3min= 2) and (A3max=*) and (A4min=*) and (A4max=*)
v.CTleft access structure
(A1=*) and (A2=*) and (A3min=*) and (A3max= 3 or A3max= 4) and (A4min=*) and (A4max=*)
v.CTright access structure
(a)
Age = [50-60]
user count query u
pre-processing
A1 A2 A3min A3max A4min A4max
1 1 3 4 1 3
system count query SKu
(b)
Figure 7: (a) Access structure of root node v. (b) Generating system count query SK_u from user count query u.

passing the right child node $v.rc$ as input parameter (Line 7-8). When the algorithm reaches a leaf node v, then v is returned (Lines 1-2). Procedure *traverseIndex* eventually returns the set $\mathcal{R}$ containing all leaf nodes satisfying SK_u (Line 11).

Example 6 Given Example 5, assume that v is the root node where $v.split\_dim = Age$ ($\hat{A}_3$) and $v.split\_value = 2$ ($range[30 - 45)$). Figure 7.(a) illustrates the access structure of $v.CT_{left}$ and $v.CT_{right}$. Figure 7.(b) shows the system count query (secret key) SK_u that was generated from the user query u such that $Age = [50 - 60]$ equates to $\hat{A}_3^{min} = 3$ and $\hat{A}_3^{max} = 4$.

Since $\hat{A}_3^{min} = 3$ from SK_u is not in the access structure of $v.CT_{left}$, then the decryption is unsuccessful, and the left subtree will not be traversed. However, $\hat{A}_3^{max} = 4$ from SK_u is in the access structure of $v.CT_{right}$, then the decryption is successful and the procedure *traverseIndex* traverses the right subtree, whose root node is $v.rc$. □

Computing Total Noisy Count

Having identified the set of leaf nodes $\mathcal{R}$ satisfying user count query u, the next step is to compute the final answer to the count query.

Procedure *Total Count Computation (compTCount)* presented in Algorithm 8 illustrates how the total noisy count is computed. It takes as input a set of leaf nodes $\mathcal{R}$ and a set of attribute

ALGORITHM 8: *compTCount*: **Total Noisy Count Computation**

```
Input: set of leaf nodes $\mathcal{R}$
Input: set of attribute distribution tokens $\mathcal{N}$
Output: ciphertext of total count $\langle r, s\rangle$
1: $\langle r, s\rangle \leftarrow \langle 1, 1\rangle$ // initialization
2: for each leaf node $l_j \in \mathcal{R}$ do
3:     $\langle r_j, s_j\rangle \leftarrow l_j.NCount$;
4:     for each token $ADT_i \in \mathcal{N}$ do
5:         if $ADT_i.tag \in l_j$ then
6:             $\langle r_j, s_j\rangle \leftarrow \langle r_j^{ADT_i.value}, s_j^{ADT_i.value}\rangle$; // scalar multiplication
7:         end if
8:     end for
9:     $\langle r, s\rangle \leftarrow \langle r.r_j, s.s_j\rangle$; // homomorphic addition
10: end for
11: return $\langle r, s\rangle$;
```

ALGORITHM 9: *queryResult*: **User Count Query Result**

```
Input: Exponential ElGamal decryption key $x$
Output: Query result ciphertext of total count $\langle r, s\rangle$
1: The Data Miner receives ElGamal encrypted result $\langle r, s\rangle$ from Service Provider;
2: $res_u = \mathbb{G}.\mathsf{Dec}(\langle r, s\rangle, x)$; // Total noisy count decryption
3: return $res_u$;
```

distribution tokens $\mathcal{N}$. For each leaf node l_j, if there is an ADT token ADT_i whose tag matches any of the tags in l_j, then a percentage of the encrypted noisy count $\langle r_j, s_j\rangle$ is computed by raising r_j and s_j to the value associated with ADT_i (Lines 5-6). To homomorphically add two noisy counts together, their first ciphertexts are multiplied together, and the same is done for their second ciphertexts (Line 9). The output of procedure *compTCount* is the encrypted total count $\langle r, s\rangle$ (Line 11).

Computing Query Result

Once ciphertext $\langle r, s\rangle$ has been computed, the service provider returns the ciphertext to the user as the final result. As per Algorithm 9, when the data miner receives the encrypted result $\langle r, s\rangle$, she uses Exponential ElGamal's private key $\mathbb{G}.x$ to decrypt the ciphertext and determine the exact noisy count res_u such that res_u satisfies differential privacy.

4.4.4 Discussion

SecDM allows data miners to reuse previously generated system queries and eliminates the need to interact with the data provider to generate the same ones again. However, this comes at the expense of requiring the user to interact with two parties (the data provider and the service provider), and to perform public key decryption operations on the results encrypted using Exponential ElGamal. In some scenarios where query reusability is not required, our framework can be easily modified to have all communications go through the data provider, as in the *Centralized SecDM (*C-SecDM*)* framework illustrated in Figure 8. Observe that in C-SecDM, the data miner does not have access to Exponential ElGamal's decryption key $\mathbb{G}.x$, as the decryption is performed by the data provider, and the total count result is then sent in clear text to the data miner via a secure channel.

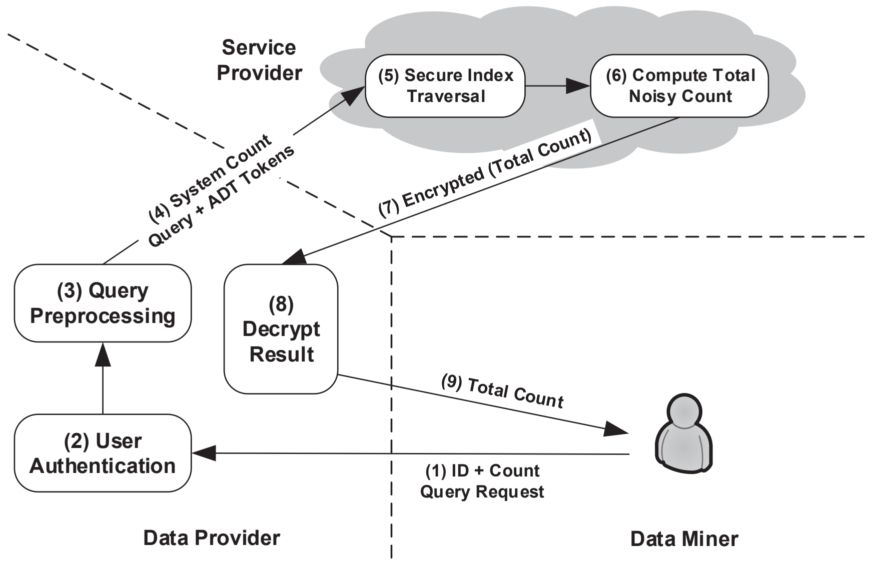

Service Provider
(5) Secure Index Traversal
(6) Compute Total Noisy Count
(4) System Count Query + ADT Tokens
(7) Encrypted (Total Count)
(3) Query Preprocessing
(8) Decrypt Result
(9) Total Count
(2) User Authentication
(1) ID + Count Query Request
Data Provider
Data Miner

Figure 8: Centralized SecDM (**C-SecDM**) framework.

To analyze the benefit of outsourcing the data to a service provider versus having the data provider handle the user queries directly, we measured the processing overhead of *specific count queries* on the data provider and the service provider when the number of queries ranges from 200 to 1000. We choose *specific count queries* to perform the experiment because they represent the worst-case scenario, where the number of nodes traversed in the kd-tree index is minimized and the number of ADT tokens is maximized. Figure 9 illustrates the results of our experiment, where we observe that the processing overhead on the proxy server is almost 10 times less than the overhead on the service provider regardless of the number of the queries.

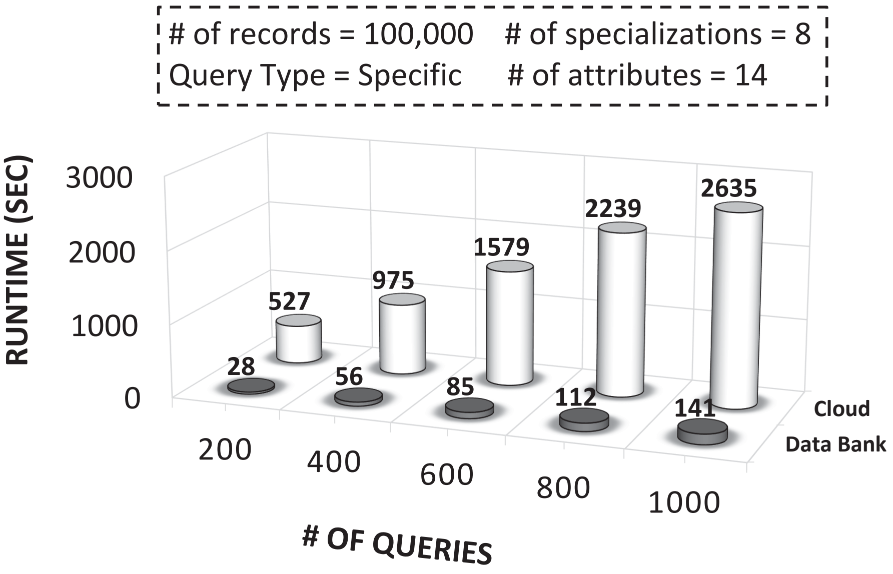


Figure 9: Performance comparison w.r.t. # of queries.

To present a practical framework we choose Exponential ElGamal to encrypt the noisy counts because this encryption scheme supports efficient homomorphic addition and integer multiplication operations. These operations are utilized by the cloud to adjust the noisy count of each identified leaf node in the kd-tree index tree using ADT tokens and then to compute the total count. However, in each ADT token, $ADT.value$ must be stored in clear text, which reveals the percentage each noisy count should be multiplied by, without reveling the actual value of the noisy count or its adjusted value. Rather than using Exponential ElGamal, we could have used other encryption schemes that support multiple homomorphic additions and multiplications. However, such schemes are inefficient and will render our solution impractical.

Shabtai *et al.* [SER12] and Shmueli *et al.* [STW+12] indicate that the anonymization approach should be chosen carefully in a multiple-release outsourcing scenario (or data update) since it normally differs from the one used in a single-release outsourcing scenario. However, this is out of the scope of this chapter.

4.5 Protocol Analysis

4.5.1 Complexity Analysis

Proposition 1 *The runtime complexity for constructing a kd-tree index from a differentially private data with d equivalent classes and k attributes using Algorithm 2 and Algorithm 3 is bounded by $\mathcal{O}(k \times d \times \log d)$ operations.*

Proof. Constructing a kd-tree with d points (equivalent classes) requires $\mathcal{O}(d \times \log d)$ [dBCvKO08]. Each node consists of two ciphertexts, each of which requires $\mathcal{O}(k_c + 2 \times k_n) = \mathcal{O}(k)$, where k_c and k_n are number of categorical attributes and numerical attributes respectively. Therefore, the required number of operations is $\mathcal{O}(k \times d \times \log d)$. ■

Proposition 2 *The runtime complexity for executing a system query SK_u over a kd-tree index with d leaf nodes using Algorithm 7 and Algorithm 8 is bounded by $\mathcal{O}(\sqrt{d} + r \times k)$ operations, where $r = |\mathcal{R}|$ and $\mathcal{R}$ is the set of reported (reached) leaf nodes.*

Proof. Since SK_u is an axis-parallel rectangular range query, the time required to traverse a kd-tree and report the points (equivalent classes) stored in its leaves is $\mathcal{O}(\sqrt{d} + r)$ [dBCvKO08]. For each *reported* leaf node, $\mathcal{O}(2 \times k_c) = \mathcal{O}(k)$ time is required to compute the total noisy count. As a result, the number of operations required to traverse the tree and answer SK_u is $\mathcal{O}(\sqrt{d} + r \times k)$. ■

4.5.2 Correctness Analysis

The correctness proof is twofold. First, we prove that Algorithm 7 identifies all the leaf nodes satisfying the user count query u. Second, we prove that Algorithm 8 produces the exact total count answer to u, and the answer is differentially private.

Proposition 3 *Given a user count query $u = \mathcal{P}_1 \wedge ... \wedge \mathcal{P}_m$, Algorithm 7 produces a set $\mathcal{R}$ containing all leaf nodes satisfying u.*

Proof. To prove the correctness of Algorithm 7 we prove *partial correctness* and *termination.*

1. Partial Correctness. We provide a proof by induction.

Basis. When u includes no predicate for any of the attributes in $\hat{D}$, then each categorical attribute in SK_u is assigned the value 1 (the identifier of the root node of the corresponding taxonomy tree), whereas for each numerical attribute $\hat{A}_i \in \hat{D}$, $\hat{A}_i^{min} = 1$ (the lowest range identifier) and $\hat{A}_i^{max}$ is assigned the highest range identifier in $\Omega(\hat{A}_i)$. When SK_u is used to traverse the kd-tree index, all internal nodes will be traversed until the leaf nodes are reached. That is, if the current node v is internal, $\mathbb{A}.\mathsf{Dec}(v.CT_{left}, SK_u)$ and $\mathbb{A}.\mathsf{Dec}(v.CT_{right}, SK_u)$ will always be true because the attributes in SK_u will always satisfy the access structure in $v.CT_{left}$ and $v.CT_{right}$, and pointers to the left child node and right child node will always be obtained.

Induction Step. Assume that traversing the kd-tree index using SK_u produces the correct set of leaf nodes $\mathcal{R}$ satisfying u. We show that if a new predicate $\mathcal{P} = (\hat{A}_i\, Op\ s_i)$ is added to u such that $\acute{u} = u + \mathcal{P}$, then traversing the kd-tree index using $SK_{\acute{u}}$ produces the correct set of leaf nodes $\acute{\mathcal{R}}$ satisfying $\acute{u}$. We observe that $\acute{\mathcal{R}} \subseteq \mathcal{R}$. To complete the proof in this step, we assume that $\mathcal{P}$ corresponds to a categorical attribute; however, the same analogy can be applied to a numerical attribute's predicate. When v is an internal node and $v.split\_dim = \hat{A}_i$, if $s_i.ID \leq v.split\_value$ then $\mathbb{A}.\mathsf{Dec}(v.CT_{right}, SK_u)$ will evaluate to *false*, and no recursive call of procedure *traverseIndex* over node $v.rc$ will be executed. This behaviour is correct because in this case the subtree whose root is $v.rc$ includes the leaf nodes that do not satisfy $\mathcal{P}$, and hence there is no need to search the subtree rooted at $v.rc$. The same logic can be used to reason about the case when $s_i.ID > v.split\_value$.

2. Termination. Each recursive call on a child node partitions the space of the parent node in half. This shows that the algorithm strictly moves from one level to a lower level in the kd-tree index while reducing the search space by half until all leaf nodes satisfying u are reached. ■

Proposition 4 *Given a set of leaf nodes $\mathcal{R}$ generated by a system count query SK_u and a set of attribute distribution tokens $\mathcal{N}$, the output of Algorithm 8 is the exact noisy count answer corresponding to SK_u.*

Proof. To prove the correctness of Algorithm 8, we prove *partial correctness* and *termination*.

1. Partial Correctness. We provide a proof by induction.

Basis. When $\mathcal{N} = \phi$, the inner loop will never be executed. In this case, procedure *compTCount* will go through all the leaf nodes in $\mathcal{R}$ and add together all corresponding noisy counts by utilizing the homomorphic addition property of Exponential ElGamal. This is correct because if no ADT token was originally generated, then the user query is an *exact* query, and 100% of the noisy count of each leaf node in $\mathcal{R}$ must be used.

Induction Step. Assume that for $\mathcal{N} = \{ADT_1, ..., ADT_l\}$, procedure *compTCount* computes the exact noisy count answer to the user count query u. We show that if a new token ADT_{l+1} for numerical attribute $\hat{A}_i$ is added such that $\acute{\mathcal{N}} = \mathcal{N} \cup ADT_{l+1} = \{ADT_1, ..., ADT_{l+1}\}$, where $\acute{\mathcal{N}}$ corresponds to the system count query $SK_{\acute{u}}$, then procedure *compTCount* computes the exact noisy count answer to the user count query $\acute{u}$. Without loss of generality, we assume that the set of leaf nodes $\mathcal{R}$ remains the same. Since ADT_{l+1} is for numerical attribute $\hat{A}_i$, then $ADT_{l+1}.value$ represents the percentage of the partial intersection between query $\acute{u}$ and attribute $\hat{A}_i$ by definition. If $\acute{u}$ is a *generic* query, then not all leaf nodes in $\mathcal{R}$ will contain a tag that corresponds to $ADT_{l+1}.tag$. However, the noisy count of each leaf node l containing a tag that matches $ADT_{l+1}.tag$ must be adjusted by multiplying $l.NCount$ with $ADT_{l+1}.value$.

2. Termination. We denote by n the initial number of leaf nodes in $\mathcal{R}$. If $n > 0$ then we enter the outer loop. We also denote by m the initial number of ADT tokens in $\mathcal{N}$. If $m > 0$ then we enter the inner loop such that after each iteration, the variable m is decreased by one, and it keeps strictly decreasing until $m = 0$ where the inner loop terminates. Similarly, the outer loop will terminate as n keeps strictly decreasing until it reaches 0; at that stage the algorithm terminates. ■

Proposition 5 *The noisy count answers satisfy ε-differential privacy.*

Proof. The proposed query processing algorithms operate on a differentially private data table and do not have access to the raw data. Because the input table is differentially private, the computed noisy count answers based on the input data table is also differentially private. Note that any post-processing does not violate the ε-differential privacy [KL10]. ■

4.5.3 Security Analysis

The proposed framework is sound since all adversaries are non-colluding and semi-honest, according to our adversarial model. In the rest of this section, we focus on proving that the protocol is confidentiality-preserving. We also illustrate the accessibility of the keys in the framework, and show that all keys are properly distributed between the parties.

Privacy by Simulation. Goldreich [Gol04] defines the security of a protocol in the semi-honest adversarial model as follows.

Definition 10 ***(Privacy w.r.t. Semi-honest Behavior) [Gol04].*** Let $f : (\{0,1\}^*)^m \mapsto (\{0,1\}^*)^m$ be an m-ary deterministic polynomial-time functionality, where $f_i(x_1, \ldots, x_m)$ is the ith element of $f(x_1, \ldots, x_m)$. Let Π be an m-party protocol for computing f. The view of the i-th party during an execution of Π over $x = (x_1, \ldots, x_n)$ is $\mathsf{view}_i^{\Pi}(x) = (x_i, r_i, m_{i,1}, \ldots, m_{i,t})$, where r_i equals the contents of the ith party's internal random tape, and $m_{i,j}$ represents the jth message that it received. For $I = \{i_1, \ldots, i_l\} \subseteq \{1, \ldots, m\}$, $\mathsf{view}_I^{\Pi}(x) = (I, \mathsf{view}_{i_1}^{\Pi}(x), \ldots, \mathsf{view}_{i_l}^{\Pi}(x))$. We say that Π securely computes f in the presence of static semi-honest adversaries if there exist probabilistic polynomial-time algorithm (simulator) S such that for every $I \subseteq \{1, \ldots, m\}$:

$$\{S(I, (x_{i_1}, \ldots, x_{i_l}), f_I(x))\}_{x \in (\{0,1\}^*)^m} \overset{c}{\equiv} \{\mathsf{view}_I^{\Pi}(x)\}_{x \in (\{0,1\}^*)^m}$$

where $\overset{c}{\equiv}$ denotes computational indistinguishability. □

According to Definition 10, it is sufficient to show that we can effectively simulate the view of each party during the execution of the SecDM protocol given the input, output and *acceptable* leaked information of that party, in order to prove that our protocol is secure. We achieve that by simulating each message *received* by a party in each algorithm. The algorithm can then be utilized to simulate the rest of the view.

First, we define the concepts *query distribution* and *query processing threshold*.

Definition 11 ***(Query Distribution.)*** The distribution of the data mining queries, denoted by U, is the set of all possible queries, where each query consists of $k_c + 2 \times k_n$ integers, each of which maps to a value in the domain of a categorical or numerical attribute. $\square$

Definition 12 ***(Query Processing Threshold.)*** Query processing threshold, denoted by α, is the maximum number of queries allowed to be processed on a kd-tree before the latter is replaced by a new shuffled and re-encrypted kd-tree submitted by data provider to the service provider. $\square$

Definition 13 ***(Privacy-preserving Data Outsourcing Framework).*** Let $\mathcal{F}$ be a framework that enables a service provider (cloud) to answer queries from data miners on hosted (outsourced) data. $\mathcal{F}$ is a privacy-preserving framework if the following properties hold:

1. **Correctness.** For any user query $u \in U$, the cloud returns res_u to the data miner such res_u is the correct answer to u.

2. **Data Confidentiality.** A semi-honest adversary $\mathcal{E}$, statically corrupting the service provider, cannot learn anything more about the hosted data from an accepted transcript of $\mathcal{F}$ than she could given only the total number of numerical and categorical attributes, and the size of each attribute's domain.

3. **Query Confidentiality.** A semi-honest adversary $\mathcal{E}$, statically corrupting the service provider, cannot learn anything about the query.

4. **Differentially Private Output.** For all $u \in U$, res_u satisfies differential privacy. $\square$

Definition 14 ***(α-Privacy-preserving Data Outsourcing Framework).*** An outsourcing framework $\mathcal{F}$ is α-privacy-preserving if it satisfies all properties in Definition 13 except that the cloud learns the search pattern of at most α number of queries. $\square$

Theorem 4.5.1 SecDM*, as specified in Protocols 2–8, is an α-privacy-preserving data outsourcing framework.*

Proof. We proved in Section 4.5.2 Property 1 (correctness) and Property 4 (differentially private output).

To prove Property 2 (data Confidentiality) and Property 3 (query Confidentiality), we build a simulator $\mathcal{S}$ that generates a view that is statistically indistinguishable from the view of $\mathcal{E}$ in real execution.

In Algorithm 4 - Line 2, the service provider receives kd-tree index T from the data provider.

Simulation :

1. Supplied with k, the total number of attributes in $\hat{D}$, and the size of each attribute's domain $|\Omega(\hat{A}_i)| : 1 \leq i \leq k$, the simulator $\mathcal{S}$ generates attribute domains $\Omega(\hat{A}'_1), \Omega(\hat{A}'_2), \ldots, \Omega(\hat{A}'_k)$ such that each domain $\Omega(\hat{A}'_i)$ consists of $|\Omega(\hat{A}_i)|$ distinct values, e.g., $1, 2, \ldots, |\Omega(\hat{A}_i)|$.

2. $\mathcal{S}$ constructs a contingency table $\hat{D}'$ with k columns each of which represents one attribute $\hat{A}'_i$, and n records each of which represents one possible combination of attribute values such that $n = \prod_{i=1}^{k} |\Omega(\hat{A}'_i)|$.

3. Supplied with the total number of numerical attributes k_n and categorical attributes k_c in $\hat{D}$ such that $k_n + k_c = k$, the size of each attribute's domain, and the security parameter of ACP-ABE, $\mathcal{S}$ runs $\mathbb{A}.\mathsf{Setup}(1^\lambda)$ to generate public key PK' and master secret key MSK'. Similarly, given the security parameter of ElGamal, $\mathcal{S}$ runs $\mathbb{G}.\mathsf{KeyGen}()$ to generate public key y' and secret key x'.

4. Given $\hat{D}'$, split dimension $i = 1$, PK' and y', $\mathcal{S}$ runs Algorithm 2 and Algorithm 3 to construct a balanced kd-tree T' over $\hat{D}'$:

   (a) In Line 12 of Algorithm 2, n random group elements are generated for each ciphertext CT_{left} or CT_{right} of each internal node v.

(b) In Line 2 of Algorithm 3, a random ElGamal ciphertext, e.g., encryption of ‘0’, is assigned to the encrypted $NCount$ of each leaf node l.

Indistinguishability Argument : T' is computationally indistinguishable from T.

First, we construct a hybrid tree called T'', and then show the relation between T'' and real the kd-tree T, and between T'' and the simulated kd-tree T'.

1. Let T'' be a kd-tree index over $\hat{D}'' = \hat{D}$ constructed using Algorithm 2 and Algorithm 3, where:

   (a) The ACP-ABE ciphertexts CT_{left} and CT_{right} of each internal node are random group elements, as per Step 4a above.

   (b) The noisy count $NCount$ in each leaf node is a random ElGamal ciphertext, as per Step 4b above.

2. T'' is computationally indistinguishable from T, denoted by $T'' \overset{c}{\equiv} T$, because:

   (a) The ACP-ABE ciphertexts in the internal nodes of the kd-tree are IND-CPA-secure under the *decisional bilinear diffie-hellman* (DBDH) assumption [Jou00] and the *decision linear* (D-Linear) assumption [BBS04].

   (b) Since ElGamal is IND-CPA-secure, the distribution of the ciphertext (output) space is independent of the key/message. Therefore, encrypting any message with a random factor is sufficient to generate a computationally indistinguishable $NCount$.

3. T'' is statistically indistinguishable from T', denoted by $T'' \overset{s}{\equiv} T'$, because:

   (a) $\hat{D}'' \overset{s}{\equiv} \hat{D}'$, where there is one-to-one correspondence between the equivalent classes in $\hat{D}''$ and the records in $\hat{D}'$.

   (b) The random coins used in ACP-ABE encryption in Algorithm 2 are drawn from the same distribution.

(c) The random coins used in ElGamal encryption in Algorithm 3 are drawn from the same distribution.

4. From Steps (2) and (3), we conclude that $T' \overset{c}{\equiv} T$.

In Algorithm 6 - Line 4, the service provider receives system count query SK_u and a set of attribute distribution tokens $\mathcal{N}$.

Simulation :

1. $\mathcal{S}$ obtains α sample queries $\bar{U} = \{u'_1, u'_2, \ldots, u'_\alpha\}$ from U.

2. For each query $u'_i \in \bar{U}$, $\mathcal{S}$ constructs a query pair $(SK_{u'_i}, \mathcal{N}'_i)$ as follows:

   - $\mathcal{S}$ runs $\mathbb{A}.\mathsf{KeyGen}(MSK', u'_i)$ to construct system count query $SK_{u'_i}$.
   - $\mathcal{S}$ constructs a set $\mathcal{N}'_i$ containing $2 \times k_n$ ADT tokens, where $ADT.value$ for each token is a randomly generated ElGamal ciphertext, e.g., encryption of '0'.

3. Up to α times, each time a data miner in the real world submits a query, $\mathcal{S}$ submits to the service in the simulation world a different query pair from the set of pairs generated in Step 2.

Indistinguishability Argument :

1. Given any real system query SK_u, $SK_{u'_i} \overset{c}{\equiv} SK_u$ because:

   (a) $u'_i \overset{s}{\equiv} u$.

   (b) $SK_{u'} \overset{c}{\equiv} SK_u$ since $|SK_{u'}| = |SK_u| = k_c + 2 \times k_n$ group element tuples, and the ACP-ABE scheme is IND-CPA-secure.

2. Given any real ADT set $\mathcal{N}$, $\mathcal{N}'_i \overset{c}{\equiv} \mathcal{N}$ because:

   - $|\mathcal{N}'_i| = |\mathcal{N}| = 2 \times k_n$.
   - The $ADT.value$ of each token in $\mathcal{N}'_i$ is computationally indistinguishable from the $ADT.value$ of any real token due to the IND-CPA-secure property of ElGamal.

Discussion. The threshold parameter α can range between 1 and ∞. To better understand the impact of revealing α queries to $\mathcal{S}$, we analyze the security when $\alpha = 1$ and $\alpha > 1$.

Case 1 : $\alpha = 1$. This represents the highest security level of our protocol, where one system query is executed per one kd-tree. Since the kd-tree index is constructed by Algorithm 2 as a *balanced* tree and since each path contains all attributes, then no correlation can be established between any two attributes and the attributes are protected when evaluated for splitting the k-dimensional space. As for the data mining query, the service provider cannot determine what attributes are included in the query, nor know what values or ranges the data miner is interested in. Since Algorithm 7 yields how many leaf nodes (equivalent classes) identified, this reveals how general the query is. In general, the more leaf nodes identified by a query, the more general the query is. The revealing of the number of identified leaf nodes, however, won't help the service provider better guess the final result of the query since it cannot access the encrypted noisy counts.

Although setting α to 1 provides the highest security w.r.t. query search pattern, it is impractical due to the cost of reconstructing the kd-tree. We refer the reader to *solution construction scalability* in Section 4.6.2 for more details about the cost of reconstructing the kd-tree.

CASE 2 : $\alpha > 1$. While our proposed framework supports *confidential access* to the data, executing multiple queries on the same kd-tree index reveals the search pattern of the queries, where the service provider is able to determine the number of leaf nodes that overlap between the queries. Let u and u' be two user queries that satisfy the same set of leaf nodes $l = \{l_1, \ldots, l_r\}$, and let *collision set* denote the set of all unique queries that could satisfy l. The size of the collision set can be determined as follows:

$$|\mathsf{collision\ set}(l)| = \prod_{i=1}^{r} \prod_{j=1}^{k} |l_i.Range(\hat{A}_j)| \ : \ \hat{A}_j \text{ is numerical,}$$

where $|l_i.Range(\hat{A}_j)|$ denotes the size of the range of attribute $\hat{A}_j$ in the equivalent class represented by leaf node l_i. Note that since the noisy counts are encrypted using ElGamal, the position of the attributes in the tree is hidden and is shuffled every time the kd-tree is constructed, disclosing the search pattern on the differentially private data reveals nothing about the final (noisy) result of each query, nor about the attributes/values in each query. The smaller the value of α is, the less overlap between queries is revealed. Several techniques have been proposed in the literature to address the problem of private search pattern, such as [WSC08]; however, it is out of the scope of this chapter.

Note that each time the data provider generates a shuffled and re-encrypted kd-tree, a different ACP-ABE master secret key MSK should be used to prevent the service provider from processing new queries on the old tree.

In our model, we assume the data miner can have access to the entire differentially-private dataset. The data privacy is guaranteed by differential privacy. Therefore, there is no need to simulate the view of the data miner. ■

Key Accessibility. Protecting the data distributed between different parties from unauthorized access is an essential part of securing the SecDM framework. We must ensure that all keys are properly distributed such that no party can decrypt any data it is not supposed to have access to in plaintext. Table 5 illustrates the accessibility of each key by each party in SecDM.

Observe that the data provider is the generator of all encryption keys in the system and maintains full control over them. The service provider, on the other hand, has no access to Exponential ElGamal's private key, $\mathbb{G}.x$, that would have allowed her to fully decrypt the contents of each leaf node in the kd-tree index. Moreover, not having access to the ACP-ABE master secret key $\mathbb{A}.MSK$ prevents the service provider from being able to determine the access structures of the ciphertexts in each internal node of the kd-tree index. As for the user (data miner), not having access to $\mathbb{A}.MSK$ prevents her from bypassing authentication and creating her own system count queries.

Table 5: Key accessibility w.r.t. all parties in SecDM framework

Encryption Scheme	**Key**	**Data Bank**	**Service Provider**	**Data Miner**
$\mathbb{G}$	private key x	generator, full Control	no access	read access
$\mathbb{G}$	public key y	generator, full Control	read access	read access
$\mathbb{A}$	master secret key MSK	generator, full Control	no access	no access
$\mathbb{A}$	public key PK	generator, full Control	read access	read access
$\mathbb{A}$	user secret key SK_u	generator, full Control	read access	read access

4.6 Performance Evaluation

In this section we evaluate the performance of the SecDM framework. First, we discuss the implementation details, and then we present the experimental results that include solution construction scalability, the scalability of query processing with respect to the number of records, and the efficiency with respect to the size of the queries.

4.6.1 Implementation and Setup

The SecDM framework is implemented in C++. Experiments were conducted on a machine equipped with an Intel Core i7 3.8GHz CPU and 16GB RAM, running 64-bit Windows 7. The index tree is implemented according to the kd-tree description in [dBCvKO08]. Both of the cryptographic primitives, *ACP-ABE* and *Exponential ElGamal*, were implemented using MIRACL[3], an open source library for big number and elliptic curve cryptography. To implement ACP-ABE, we chose Boneh-Lynn-Shacham (BLS) pairing-friendly curve from [BLS01]: $Y^2 = X^3 + b$, where $b = \sqrt{w + \sqrt{m}}$, $m = \{-1, -2\}$, and $w = \{0, 1, 2\}$. The chosen elliptic curve has a pairing embedding degree of 24 and a AES security level of 256. The pairing $e : G_1 \times G_2 \to G_T$ is a type 3 pairing where G_1 is a point over the base field, G_2 is a point over an extension field of degree 3, and G_T is a finite field point over the k-th extension, where $k = 24$ is the embedding degree for the BLS curve. To

[3] MIRACL: https://certivox.org/display/EXT/MIRACL

implement *Exponential ElGamal* we randomly choose the message space and calculation modulus p to be a large 2048-bit prime for which $q = (p-1)/\alpha$ is a 256-bit prime. Since *Exponential ElGamal* depends on the multiplicative order of g and having a large collection of ciphertexts, we choose g to be a generator of the multiplicative subgroup $\mathbb{G}_q$ such that $order(g) = q - 1$.

We utilize a real-life *adult* data set [BL13] in our experiments to illustrate the performance of SecDM framework. The adult data set consists of 45,222 census records containing six numerical attributes, eight categorical attributes, and a *class* attribute with two levels: "$\leq 50K$" and "$> 50K$". A further description of the attributes can be found in [FWY07a]. Since the maximum number of attributes is 14, we assume that the number of attributes in a query can range from 2 to 14, and the average number of attributes in a query is 8. We generate ε-differentially private records using the *DiffGen* algorithm, where the privacy budget $\varepsilon = 1$, the number of specializations is set to 8, 10, or 12, and choose the utility function $Max(D, v)$ to determine the score of each candidate v during the specialization process.

4.6.2 Experimental Results

Scalability

Solution Construction Scalability There are three major phases involved in constructing the SecDM framework: data anonymization using the *DiffGen* algorithm, data preprocessing, and kd-tree index construction; the latter can be further divided into two subphases: *internal nodes construction* and *leaf nodes construction.* According to procedure *buildIndex* in Algorithm 2, the complexity for constructing SecDM is dominated by the number of ε-differentially private records, which in turn is impacted by the number of raw data records and the setting of the number of specializations for the *DiffGen* algorithm. The objective is to measure the runtime of each construction phase to ensure its capability to scale up in terms of records size.

Figure 10 depicts the runtime of each of the construction phases, where the number of data records ranges from 20,000 to 100,000 records, and the number of specializations is set to 8. We

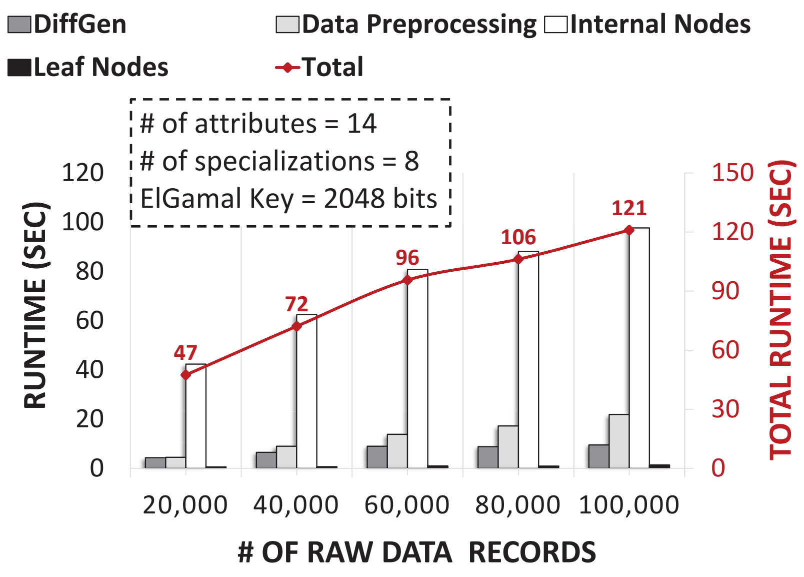
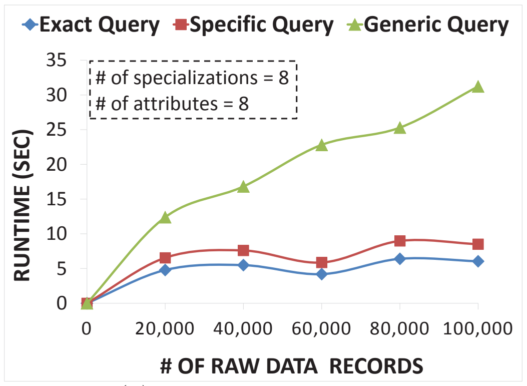


Figure 10: Scalability of framework construction w.r.t. # of records.

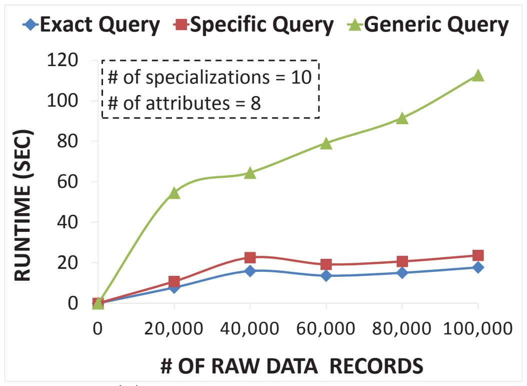
observe that the runtime of each phase grows linearly as the number of records increases. We also observe that the overall construction runtime scales up linearly as well, as it takes 47 sec to construct the framework for a data set with 20,000 records, 72 sec for 40,000 records, 96 sec for 60,000 records, 106 sec for 80,000 records, and 121 sec for 100,000 records. Since each phase of the algorithm, as well as the overall construction time, grow linearly with respect to the total number of records, this suggests the construction of SecDM is scalable with regard to the data size.

Query Processing Scalability One major contribution of our work is the development of a scalable framework for query processing on anonymized data in the cloud. Since the number of specializations during the anonymization process impacts the total number of anonymized records, we study the runtime for answering different types of user count queries under a different number of specializations, while the number of raw data records ranges from 20,000 to 100,000. Given the three user count query types, *Exact*, *Specific*, and *Generic*, we randomly create 500 queries of each type, and report the average runtime, where the average number of attributes in each query is 8.

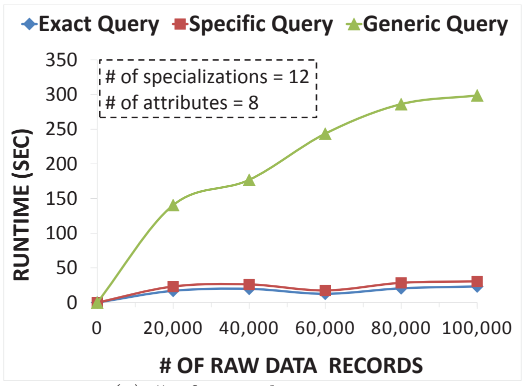
Figures 11a, 11b, and 5b depict the processing runtime of each type of user count queries when the number of specializations is set to 8, 10, and 12, respectively. In Figure 11a, we observe that the processing runtime of each query type grows linearly as the number of raw data records continues to



(a) # of specializations = 8



(b) # of specializations = 10



(c) # of specializations = 12

Figure 11: Scalability of query processing w.r.t. the number of raw data records and the number of specializations.

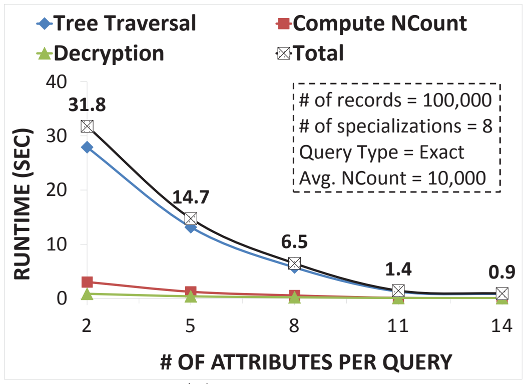
increase at the same rate. That is, the processing runtime grows from 4.8 sec for 20,000 records to 6 sec for 100,000 records when the query type is *exact*; from 6.5 sec for 20,000 records to 8.5 sec for 100,000 records when the query type is *specific*; and from 12.4 sec for 20,000 records to 31.2 sec for 100,000 records when the query type is *generic*. Similarly, in Figures 11b and 5b we observe that the processing runtime of each query type is linear with regard to the number of raw data records for all three types. The increase in the number of specializations leads to a higher number of anonymized records, thus explaining the increase in the average query processing runtime for each query type in Figures 11a, 11b, and 5b.

Note that performance is affected by the total number of attributes (dimensions) in the data. For the purpose of this work, the *k*d-tree index is sufficient for the *Adult* dataset with 14 attributes. However, for higher dimensionality, another data structure, such as x-tree, might be needed to preserve practicality in high dimensional data.

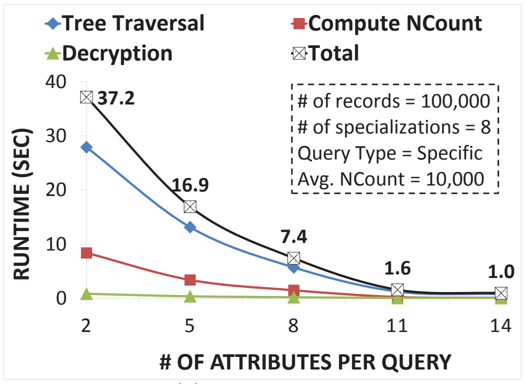
Efficiency

To demonstrate the efficiency of our SecDM framework we measure the impact of the number of attributes in a query on the processing time needed by the cloud to process the query and by the user to decrypt the result. We split the query processing phase into two subphases: *tree traversal* and *compute NCount*. We assume the number of specializations is 8, while the number of raw data records is 100,000. We create 500 queries of each query type, and report the average runtime.

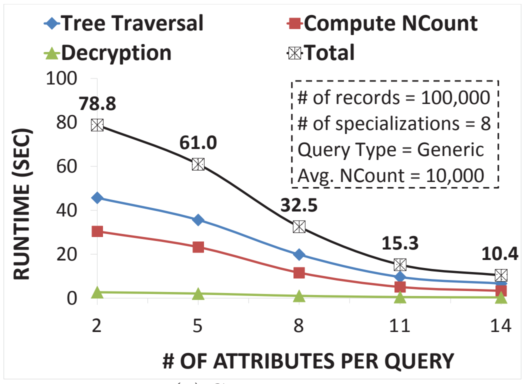
Figures 12a, 12b, and 12c depict the processing runtime of *exact*, *specific*, and *generic* queries, respectively, when the average number of attributes in a query ranges from 2 to 14. We observe that the most dominant phase with regard to the processing runtime is the tree traversal phase, whereas the resulting decryption phase is the least dominant. The total processing runtime of each query type decreases linearly as the number of attributes per query increases. That is, the total runtime decreases from 31.8 sec to 0.9 sec when the number of attributes per query increases from 2 to 14 for *exact* queries, decreases from 37.2 sec to 1 sec when the number of attributes per query increases from 2 to 14 for *specific* queries, and decreases from 78.8 sec to 10.4 sec when the number of attributes per query increases from 2 to 14 for *generic* queries. The total processing runtime improves as the number of attributes increases because adding more attributes to a query makes it more restrictive and, consequently, requires fewer nodes to be traversed in the kd-tree index. Assuming the average noisy count value for each anonymized record is 10,000, we observe that the decryption phase, which involves decrypting Exponential ElGamal ciphertexts, is very small (less than 2 sec) and barely sensitive to the increase in the number of attributes per query.



(a) Exact query



(b) Specific query



(c) Generic query

Figure 12: Efficiency w.r.t. the number of attributes per a query for *exact*, *specific*, and *generic* queries.

4.7 Summary

In this chapter, we propose a privacy-preserving framework for confidential count query processing in a cloud computing environment. Our framework maintains the privacy of the outsourced data while providing data confidentiality, confidential query processing, and privacy-preserving results. Users (data miners) of the system are not required to have prior knowledge about the data, and incur lightweight computation overhead. The framework also allows for query reusability, which reduces the communication and processing time. We perform several experimental evaluations on real-life data, and we show that the framework can efficiently answer different types of queries and is scalable with regard to the number of data records.

Chapter 5

Secure and Privacy-preserving Set-Valued Data Integration with Public Verifiability

5.1 Introduction

Set-valued data, a set of items selected from a pre-defined domain, is commonly associated with stored objects (e.g. persons, products) in databases such as web search logs, market basket, and passengers' transit records [TMK08][HN09]. Integrating related data from different sources enables businesses and government agencies to perform better data analysis and make better decisions.

The vast majority of the literature about distributed data integration and publishing deals with *semi-honest* adversaries [MAFD14][JC06][GLM$^+$13b][NH12], where each participant is assumed not to deviate from the protocol, while trying to infer information from the other parties, even if deviating enables them to infer more information. Although this assumption may be suitable in some cases, it is often unrealistic. By contrast, designing protocols that are secure against *malicious* adversaries provides assurance that the protocol will execute correctly and that secret inputs will stay secret regardless of the behaviour of the other participants. However, it generally comes at a cost in performance. As just one example, Kantarcioglu and Kardes [KK07] show that the secure dot product protocol is 700 times slower in the malicious model than it is in the semi-honest.

To ensure that a distributed protocol executes correctly, participants must prove they are following the protocol at each step. Such proofs may only convince the other participants in the protocol by being contingent on their interactions or individual secrets. A publicly (or universally) verifiable protocol produces a transcript proving correct execution that is verifiable by anyone at any time. In the database community, verifiability has been mainly studied in the areas of data authentication [Tam03, DBP07], data streams [CCM09, CKLR11], and public auditing [WCW$^+$13, WLL12]. In verifiable computation, a single data owner delegates a computationally heavy task to the cloud while being able to verify the results [BGV11, CKV10, GGP10, GKR08]. Outside of databases, public verifiability is an important property for a number of cryptographic protocols (e.g., voting schemes [Ben87, CGS97]). Public verifiability also adds an additional performance cost. As one example, the publicly verifiable voting protocol for complex scoring rules in [TRN08] is estimated to take 10 000 hours to produce a verifiable tally.

For data integration, public verifiability enables any third party to verify the data is properly protected (auditors, data holders, individuals in the data set) and properly integrated (data miner) in a non-interactive fashion, offline and at any time. We are the first to study public verifiability for this application. We answer open questions surrounding the achievability of public verifiability for a complex, full-featured integration protocol. We present a flexible protocol that supports horizontally and vertically partitioned data, two and multiples parties, and differentially private outputs. The result is a feasible but expensive protocol: we estimate 389 hours (single-threaded) to integrate datasets of 600 records and 10000 distinct items. We show public verifiability is achievable and believe improving its performance is a promising research direction.

Let us consider the following scenario to illustrate the threat model. An Australian government agency uses a set of principles to govern the process of Commonwealth person-specific data integration for statistical and research purposes[1]. Data custodians (agencies) collect, store and manage datasets on behalf of data providers. For each integration project, an integrating authority – a single trusted organisation specialized in data privacy and security – is appointed to handle the reception of the related and de-identified datasets from data custodians, and the running of the integration process to generate a dataset that prevents disclosure attacks against individuals. The integrating authority might outsource part of the integration process to a third-party, while adhering to the data integration principles and maintaining full responsibility for complying with the confidentiality requirements specified by the data custodians. To ensure transparency in each integration project, the government conducts scheduled audits and periodically publishes detailed information about the project. Similar agencies can be found in other countries, e.g., Population Data BC (PopData) [2] in Canada.

A major concern in this scenario is *data privacy*, where for each integration project, data custodians (on behalf of data providers) must fully trust the appointed integrating authority with their data, and must trust that the involved third-parties will not collude together trying to learn sensitive information about individuals from the data. Although the data is de-identified, attacks such as record/attribute linkage and re-identification of individuals can still be achieved (e.g., GIC [Swe02b], AOL [BZ06], Netflix [NS08]). The following example illustrates this privacy threats when straightforwardly integrating distributed set-valued data.

Example 7 Let Alice, Bob, and Carol represent three data custodians owning person-specific datasets DS_1, DS_2, and DS_3, respectively. Each transaction corresponds to an individual, and has a unique identifier (TID) and a set of distinct items from the item universe $\mathsf{Univ} = \langle I_1, I_2, I_3, I_4 \rangle$, as shown in Table 6. These parties are interested in integrating DS_1, DS_2, and DS_3 to build a classifier. Straightforward integration is achieved by grouping items together by TID, and then removing duplicate items. Integrating transactions $\mathsf{DS}_1.\mathsf{T}_2$, $\mathsf{DS}_2.\mathsf{T}_2$, and $\mathsf{DS}_3.\mathsf{T}_2$ together results in a transaction

[1]Statistical Data Integration Involving Commonwealth Data: http://statistical-data-integration.govspace.gov.au/
[2]PopData: https://www.popdata.bc.ca/

Table 6: Sample set-valued datasets

(a) Alice: DS_1

TID	**Items**
T_1	$\{I_1, I_4\}$
T_2	$\{I_2\}$

(b) Bob: DS_2

TID	**Items**
T_1	$\{I_2\}$
T_2	$\{I_2, I_3, I_4\}$
T_3	$\{I_1\}$
T_4	$\{I_1, I_3\}$

(c) Carol: DS_3

TID	**Items**
T_1	$\{I_3\}$
T_2	$\{I_2, I_4\}$
T_3	$\{I_1, I_2\}$
T_4	$\{I_4\}$
T_5	$\{I_1, I_2, I_3, I_4\}$
T_6	$\{I_1\}$

with the set of items $\{I_2, I_3, I_4\}$ after removing the duplicates of I_2 and I_4. However, since I_2 exists only once in DS_1 (in transaction T_2), Alice can use her knowledge of the duplicates to conclude that the integrated transaction corresponds to T_2, and to determine with 100% certainty that I_4 exists in both $\mathsf{DS}_2.\mathsf{T}_2$ and $\mathsf{DS}_3.\mathsf{T}_2$. $\square$

Another concern in this scenario is *high-dimensionality*. Set-valued data is high dimensional by nature, and therefore, any approach on set-valued data must ensure scalability with respect to the number of distinct items in the data. That is why existing approaches designed for relational data, e.g. Mohammed *et al.* [MAFD14], are not suitable to handle set-valued data. Our proposed protocol grows at a logarithmic rate with respect to the number of dimensions. For example, increasing the number of dimensions by 20,000 (from 20,000 to 100,000 distinct items) increases the overall runtime on average by only 8 hours.

Another concern in the motivating scenario is *audit malfeasance*. Although the government performs periodic auditing to protect against errors and corruptions by individuals and ensure the correctness of the integration process, such *administrative verifiability* does not protect against wrongdoing by government officials or entities responsible for conducting the audit.

In this chapter, we address the aforementioned concerns by proposing a privacy-preserving multi-party protocol for integrating person-specific set-valued data in a malicious environment with dishonest majority. Our protocol supports *public verifiability* — a stronger version of administrative verifiability — where any internal or external party at any time can independently verify the correctness of the performed integration. We take the *single-party* algorithm for differential privacy recently proposed by Chen *et al.* [CMF+11] as a basis to our approach and extend it to the *multi-party* setting. We adopt differential privacy [DMNS06], a rigorous privacy model that provides a provable privacy guarantee while making no assumption about the background knowledge of the adversary. Our contributions are summarized as follows:

Contribution 1. We propose a distributed protocol, named SecSVD, for integrating high-dimensional set-valued data and releasing integrated differentially private data with an effective utility for data mining. The complexity of our approach is logarithmic with respect to the number of distinct items (dimensions) in the item universe.

Contribution 2. Our proposed approach supports public verifiability, where any interested party can fully verify the integrity of an execution of the protocol, independently ensuring that the output satisfies differential privacy. To our knowledge, this is the first work to construct a publicly verifiable protocol for data integration.

Contribution 3. While most data integration protocols only provide security against semi-honest (passive) adversaries, we show that our proposed protocol is secure against fully malicious active adversaries. Further, we do not require an honest majority; this enables both two-party and multi-party settings, where the data is distributed among n parties such that $n \geq 2$.

Contribution 4. Unlike most of the existing works in the literature that assume the data is partitioned either horizontally [JX09, AMFD12, KC04, ZRZ+13] or vertically [JC06, MAFD14, DKM+06], our proposed solution supports a more general data model where the data can be partitioned horizontally or vertically, or both. In the case of vertical partitioning, our solution supports items overlap between transactions corresponding to the same individual but belong to different data owners.

In Table 7, we summarize the main features of the representative approaches in the areas related to our work, including our proposed protocol.

The results of this chapter are currently under review in PoPETs [DCF].

Table 7: Comparative evaluation of main features in related approaches including our proposed approach

Approach	Data Type		Privacy-Preserving Domain				Hosting Environment					Security	
			Non-Interactive		Interactive			Two		Multiple			
	Set-Valued	Other	Differential Privacy	Syntactic Privacy	Differential Privacy	Syntactic Privacy	Single	Horiz.	Vert.	Horiz.	Vert.	Threat Model †	Public Verifiability
Terrovitis *et al.* [TMK08, TMK11]	●			●			●						
He and Naughton [HN09]	●			●			●						
Chen *et al.* [CMF+11]	●		●				●						
Jiang and Clifton [JC06]		●		●					●			○	
Jurczyk and Xiong [JX09]		●		●						●		○	
Alhadidi *et al.* [AMFD12]		●	●					●				○	
Mohammed *et al.* (DistDiffGen) [MAFD14]		●	●						●			○	
Mohammed *et al.* (TIPS) [MFD11]		●		●					●			●	
Bhaskar *et al.* [BLST10], Li *et al.* [LQSC12]	●				●		●						
Wong *et al.* [WCH+07]	●						●					○	
Giannotti *et al.* [GLM+13b]	●					●	●					○	
Kantarcioglu and Clifton [KC04]	●									●		○	
Zhang *et al.* [ZRZ+13]	●							●				○	
Wahab *et al.* [WHZ+14]	●				●			●		●		○	
Dwork *et al.* [DKM+06]		●			●					●		●	
Narayan and Haeberlen [NH12]		●			●			●		●		○	
Our proposed solution SecSVD	●		●					●	●	●	●	●	●

† In this column, ○ denotes semi-honest threat model whereas ● denotes malicious threat model.

5.2 Preliminaries

5.2.1 Centralized Protocol for Publishing Set-Valued Data via Differential Privacy

Chen *et al.* [CMF+11] proposed a single-party top-down partitioning approach using *context-free taxonomy tree* to produce anonymized set-valued data in a centralized environment. We present a secure multi-party version in Section 5.3.

Next, we summarize the approach in [CMF+11]. The algorithm first constructs a context-free taxonomy tree that will be utilized during the anonymization process to ensure utility of the anonymized set-valued data. The tree is constructed from a universe of items, where each internal node is a set of its leaf nodes (items). Figure 13 presents an example of a context-free taxonomy tree.

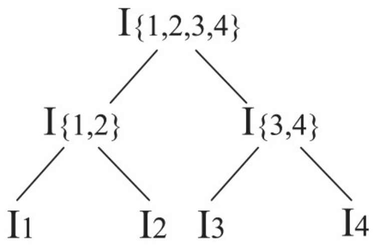


Figure 13: A context-free taxonomy tree for item universe $\{I_1, I_2, I_3, I_4\}$.

Next, all data records are generalized into one partition, and the root node of the taxonomy tree is assigned to the partition as a common representation called *hierarchy cut*. A hierarchy cut typically consists of one or more taxonomy tree nodes.

Definition 15 ***Record Generalization [CMF+11].*** A set-valued record R can be generalized to a hierarchy cut $HCut$ if:

1. Every item in R can be generalized to a node in $HCut$.
2. Every node in $HCut$ generalizes some items in R. □

For example, according to the context-free taxonomy tree in Figure 13, if $R = \{I_1, I_2\}$ then R

can be generalized to $HCut_1 = \{I_{1,2}\}$ and $HCut_2 = \{I_{1,2,3,4}\}$, but not to $HCut_3 = \{I_{\{1,2\}}, I_{\{3,4\}}\}$.

Next, a top-down partitioning process based on the taxonomy tree is applied in order to generate disjoint sub-partitions with more specific hierarchy cuts. For each sub-partition p, a Laplace noise is generated based on its allocated privacy budget. If the noisy count of p is smaller that threshold $thres_1 = \sqrt{2}C \times height(p.HCut)/p.\alpha$, where C is a given constant and α is the partitioning budget, then p is considered "empty" and not further partitioned. Otherwise, the p is considered "non-empty" and it is further split into sub-partitions with more specific hierarchy cuts. The splitting stops when no further sub-partitions can be generated. Each partition whose hierarchy cut is a single leaf node from the taxonomy tree is called a *leaf partition*. To reduce the impact of noise, a leaf partition p' is considered non-empty if its noisy count is greater or equal threshold $thres_2 = \sqrt{2}C'/(\varepsilon/2 + p'.\tilde{B})$, where C' is a constant $\in [1, C]$, ε is the total privacy budget and $\tilde{B}$ is the allocated budget.

Privacy Budget Allocation. Half of the total privacy budget, $\varepsilon/2$, is used to guide the partitioning process, whereas the other half is added to the leaf partitions to obtain their noisy counts. For each partition, the privacy budget assigned to its partitioning operation depends on the maximum number of partitioning operations needed to reach leaf partitions.

Theorem 5.2.1 *[CMF+11] Given taxonomy tree T, the maximum number of partitioning operations of a partition p with hierarchy cut $HCut$ is:*

$$|InternalNodes(HCut)| = \Sigma_{u_i \in HCut} |InternalNodes(u_i, T)|,$$

where $|InternalNodes(u_i, T)|$ is the number of internal nodes of the sub-tree of T rooted at u_i. □

Since sub-partitions are disjoint, the privacy budget of a partitioning operation is further assigned in full to each resulting sub-partitions, due to the *parallel composition property* [McS09].

5.2.2 Encryption Scheme

Our protocol requires an additively homomorphic encryption scheme that allows ciphertexts to be re-randomized without private information. It must also admit distributed key generation (DKG) and distributed decryption, enabling participants to use key shares to perform a decryption operation. Finally it must allow the key holders to transfer a ciphertext from their key to any other key without decrypting the ciphertext (proxy re-encryption). The best candidate is exponential ElGamal [CGS97]: it is fast when implemented over elliptic curves, distributed key generation is straightforward (unlike Paillier) [Bra06], and decryption is feasible for our plaintext space. In the rest of the thesis, we denote the encryption of message m as $[\![m]\!]$ for brevity.

5.2.3 Mix and Match Protocol

Mix and Match Protocol could alone realize our entire protocol given that lookup tables are sufficient for implementing general computing. However, such approach will be expensive, i.e., the complexity will be exponential in the number of input variables. Our protocol is designed to use small single-input lookup tables sparingly. Like our overall protocol, Mix and Match itself is publicly verifiable, secure against malicious adversaries, and secure with a dishonest majority.[3]

5.3 Solution: SecSVD Protocol

In this section, we first present an overview of our differentially private set-valued data release protocol, and then elaborate on the key components of our protocol.

5.3.1 Solution Overview

Given $p \geq 2$ data owners $\mathcal{P}_1$, $\mathcal{P}_1$, …, $\mathcal{P}_p$ respectively owning set-valued datasets DS_1, DS_2, …, DS_p, where items in all datasets are drawn from an item universe $\mathsf{Univ} = \langle I_1, I_2, \ldots, I_n \rangle$, and given a privacy budget ε agreed upon by the data owners, the objective of our proposed solution is to securely generate an integrated and sanitized set-valued dataset OutDS for data mining purposes such that (1) no unnecessary information is disclosed about the individual datasets during the integration

[3]Note that security against a dishonest majority is not explored by the authors themselves [JJ00] but the bound follows solely from the bound on the DKG. As in any case of a dishonest majority, causing the protocol to terminate early without a result becomes possible.

process, (2) communications between all parties are secure under the malicious adversarial model, (3) OutDS satisfies ε-differential privacy, and (4) OutDS is encrypted such that only the data miner receiving the dataset can decrypt it. Our solution consists of two main protocols:

Protocol 5.1 - Data Preparation. The data owners jointly run this protocol to construct a context-free taxonomy tree TaxTree which will be later utilized to construct the anonymized set-valued data. Moreover, this protocol enables the data owners to securely merge their datasets into one encrypted dataset MixDS such that all items and records in the dataset are shuffled and randomized.

Protocol 5.2 - Secure Distributed Differentially Private Set-Valued Data Release. This protocol represents our distributed differentially private set-valued data anonymization approach based on generalization. Using the context-free taxonomy tree TaxTree generated by Sub-Protocol 1.1, the data owners jointly run this protocol to partition MixDS using a top-down partitioning algorithm, where Laplace noise is used on each partition to non-deterministically decide whether child partitions should be created. The result is OutDS, a set-valued differentially private dataset encrypted under the data miner's public key.

5.3.2 Adversarial Model

Our protocol protects against computationally-bounded *malicious* adversaries who are are *not* trusted to execute the steps of the protocol correctly. Adversaries are *static* (i.e., parties are corrupted before the start of the protocol). Privacy and integrity can be realized while up to $(p - 1)$ out of p data owners are corrupted and colluding together. The protocol guarantees *completeness* as it always produce the desired result (differentially private data); however, *fairness* is not guaranteed since malicious parties can stop participating and cause the protocol to halt. Data miners may also be corrupted; however, we assume that data owners and miners cannot be simultaneously corrupt. While this assumption is not ideal and can be removed with an exponentially-expensive solution, we utilize it in our protocol as a trade-off for efficiency. (We do not prove that efficiency *requires* such a non-collusion assumption—this is an interesting open problem).

5.3.3 Data Model

Most approaches concerning the problem of privacy-preserving data integration and publishing assume that the distributed data is partitioned either horizontally [Tas14, AMFD12] or vertically [JC06, AMFD12]. In this chapter, we adopt a broader and more realistic view by assuming that the data can be a mix of horizontal and vertical partitioning. That is, multiple data owners might own data about the same set individuals, with a possibility of item overlap between records referencing the same individual.

5.3.4 Auxiliary Functions

To simplify the presentation of our cryptographic protocols, we define a set of general auxiliary functions that will be used throughout the remainder of the chapter. Construction 5.1 illustrates how the auxiliary functions can be constructed.

Auxiliary Functions

Given a vector of binary ciphertexts $V = \langle [\![v_1]\!], [\![v_2]\!], \ldots, [\![v_n]\!] \rangle$, all participants use Mix and Match to jointly generate and evaluate lookup tables for the following helper functions:

1.
$$\mathsf{n\text{-}AND}(V) := \begin{cases} [\![0]\!] & : 0 \le \sum_{i=1}^{n} v_i < n \\ [\![1]\!] & : \sum_{i=1}^{n} v_i = n \end{cases}$$

2.
$$\mathsf{n\text{-}OR}(V) := \begin{cases} [\![0]\!] & : \sum_{i=1}^{n} v_i = 0 \\ [\![1]\!] & : 1 \le \sum_{i=1}^{n} v_i \le n \end{cases}$$

3.
$$\mathsf{NOT}([\![v]\!]) := \begin{cases} [\![0]\!] & : v = 1 \\ [\![1]\!] & : v = 0 \end{cases}$$

4.
$$\mathsf{n\text{-}NOT}(V) := \langle [\![\bar{v}_1]\!], [\![\bar{v}_2]\!], \ldots, [\![\bar{v}_n]\!] \rangle : \ [\![\bar{v}_i]\!] := \mathsf{NOT}([\![v_i]\!])$$

Construction 5.1: Auxiliary functions we construct for the protocol

5.3.5 Data Preparation

As a precursor to our protocol, we assume the p data owners generate a public key and p shares of the decryption key such that all p data owners must participate in any decryption operation.

Data preparation is an initial step that involves constructing an encrypted context-free taxonomy tree $\mathsf{TaxTree}$ to help generate a sanitized set-valued data while guaranteeing utility for data mining tasks such as counting queries and frequent itemsets mining [CMF+11]. It also involves securely merging all datasets into one encrypted dataset MixDS where values are blinded and columns and rows are randomly shuffled to prevent possible record linkage by the data owners. Protocol 5.1 is evaluated to construct $\mathsf{TaxTree}$ and MixDS as follows:

$\mathsf{TaxTree}$ **Construction** (Steps 1-2). The data owners jointly apply the Verifiable Mix Network protocol on Univ and generate a universe of mixed item names $\mathsf{MixUniv}$ such that the items are shuffled and randomized. Given $\mathsf{MixUniv}$, and a fan-out value f that is agreed upon by the data owners, $\mathsf{TaxTree}$ is then constructed by evaluating Sub-Protocol 1.1, where each leaf node is a single item from $\mathsf{MixUniv}$, and each non-leaf node represents a set of items from $\mathsf{MixUniv}$.

Example 8 Given item universe $\mathsf{Univ} = \langle I_1, I_2, I_3, I_4\rangle$ for the datasets presented in Table 6, let $\mathsf{MixUniv} = \langle \mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3, \mathbf{A}_4\rangle$ be the universe of the shuffled and randomized items under random permutation π, where $\mathbf{A}_1 = [\![I_{\pi(1)}]\!] = [\![I_4]\!]$, $\mathbf{A}_2 = [\![I_{\pi(2)}]\!] = [\![I_1]\!]$, $\mathbf{A}_3 = [\![I_{\pi(3)}]\!] = [\![I_2]\!]$, and $\mathbf{A}_4 = [\![I_{\pi(4)}]\!] = [\![I_3]\!]$. Given fan-out value $f = 2$, Figure 14 illustrates a context-free taxonomy tree $\mathsf{TaxTree}$ constructed according to Sub-Protocol 1.1, where $\mathbf{A}_{\{1,2,3,4\}} = \{\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3, \mathbf{A}_4\}$, $\mathbf{A}_{\{1,2\}} = \{\mathbf{A}_1, \mathbf{A}_2\}$, and $\mathbf{A}_{\{3,4\}} = \{\mathbf{A}_3, \mathbf{A}_4\}$. $\square$

Generating a shuffled and randomized context-free taxonomy tree $\mathsf{MixUniv}$ hides the *hierarchy cut* and the *complementary cut* of each partition from the data owners (as we will see in Section 5.3.6).

MixDS **Construction** (Steps 3-6). Given DS_1, DS_2, $\ldots$, DS_p, Sub-Protocol 1.2 is first evaluated to merge all datasets into one encrypted dataset EncDS. Specifically, each data owner transforms each person-specific record in his dataset into a vector of $|\mathsf{Univ}| = n$ binary ciphertexts, each of which indicates whether or not the corresponding item from Univ exists in the record. That is, the

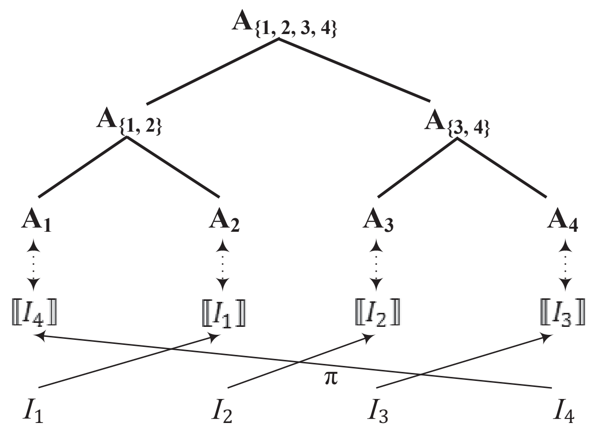


Figure 14: A context-free taxonomy tree TaxTree for the datasets presented in Table 6.

first ciphertext in the vector corresponds to the first item in Univ, the second ciphertext corresponds to the second item, etc. For each item in Univ, if the item exists in a record, then its corresponding ciphertext is set to $[\![1]\!]$; otherwise, it is set to $[\![0]\!]$. Next, the data owners jointly merge their encrypted datasets, where the merging process is dependent on how the distributed datasets are partitioned. If the datasets are vertically partitioned, then $|\mathsf{DS}_k| = d : 1 \leq k \leq p$ and each dataset contains records about the same set of individuals, with possible item overlaps. In this case, for each individual, the corresponding ciphertext vectors from all encrypted datasets are homomorphically merged together to output a vector of n binary ciphertexts, each of which corresponds to an item from Univ. For each item in Univ, if the item exists in at least one of the individual's ciphertext vectors, then its corresponding ciphertext in the output vector is set to $[\![1]\!]$; otherwise, it is set to $[\![0]\!]$. On the other hand, if the datasets are horizontally partitioned, i.e., each dataset $\mathsf{DS}_k : 1 \leq k \leq p$ contains records about different set of individuals, then the encrypted datasets are simply appended together to construct EncDS.

Finally, the data owners jointly apply the Verifiable Mix Network protocol on the columns and rows in EncDS to generate a shuffled and randomized dataset MixDS. The columns are shuffled using the same random permutation applied on the items in TaxTree to maintain the correspondence between the items in TaxTree and the columns in MixDS.

Example 9 Table 8.a illustrates the output dataset EncDS of Sub-Protocol 1.2 when evaluated on

Data Preparation

Prior to the generation of the differentially private set-valued data, the following steps are performed:

1. All participants (data owners) jointly apply the Verifiable Mix Network protocol on all items $I_1, I_2, \ldots, I_n$ in Univ, and generate shuffled and randomized item names $\mathsf{MixUniv} = [\![I_{\pi(1)}]\!], [\![I_{\pi(2)}]\!], \ldots, [\![I_{\pi(n)}]\!]$, where π is a random permutation on n elements such that no single participant knows the permutation. For ease of reference, we refer to the first item $[\![I_{\pi(1)}]\!]$ as $\mathbf{A}_1$, second item $[\![I_{\pi(2)}]\!]$ as $\mathbf{A}_2$, etc.
2. Build a context-free taxonomy tree $\mathsf{TaxTree}$ according to Sub-Protocol 1.1.
3. Generate EncDS according to Sub-Protocol 1.2.
4. All participants jointly apply the Verifiable Mix Network protocol on all columns $I_1, I_2, \ldots, I_n$ in EncDS, and generate shuffled and re-randomized columns corresponding to $I_{\pi(1)}, I_{\pi(2)}, \ldots, I_{\pi(n)}$, where π is the *same* random permutation used in Step 1 above.
5. All participants jointly apply the Verifiable Mix Network protocol on all rows $\mathsf{EncT}_1, \mathsf{EncT}_2, \ldots, \mathsf{EncT}_d$ in EncDS, and generate shuffled and re-randomized rows $\mathsf{EncT}_{\pi'(1)}, \mathsf{EncT}_{\pi'(2)}, \ldots, \mathsf{EncT}_{\pi'(d)}$, where π' is a random permutation on d elements such that no single participant knows the permutation. For ease of reference, we refer to the first row $\mathsf{EncT}_{\pi'(1)}$ as R_1, the second $\mathsf{EncT}_{\pi'(2)}$ as R_2, etc.
6. Output a shuffled and re-randomized dataset MixDS.

Protocol 5.1: Data Preparation

Context-Free Taxonomy Tree

To construct a context-free taxonomy tree $\mathsf{TaxTree}$ from item universe $\mathsf{MixUniv} = \langle \mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_n \rangle$ given fan-out f:

1. Partition $\mathsf{MixUniv}$ into n/f partitions, each of which contains f items. If n is not divisible by f, smaller groups can be created.
2. Group the items from each partition into one node.
3. Group each f adjacent nodes into an upper level node.
4. Step (c) is repeated from one level to an upper level until a single root is reached.

Sub-Protocol 1. 1: Context-Free Taxonomy Tree, adapted from [CMF+11]

the datasets in Table 6. Table 8.b illustrates dataset MixDS generated by permuting columns and rows in EncDS according to random permutation π and π' respectively such that all ciphertexts have been randomized. □

PartTree **Construction**

An initial partition Part is created, where $HCut$ is set to the root of TaxTree, $CCut = \phi$, and $ABudget = \varepsilon/2$, where ε is the global privacy budget. The partitioning tree PartTree is constructed as follows:

1. Securely assign to Part the records in MixDS as follows:

   (a) If Part is the *initial partition*, then all records in MixDS are assigned to Part by default, and all ciphertexts in $Recs$ are set to $[\![1]\!]$:

   $$Recs := \langle [\![r_1]\!], [\![r_2]\!], \ldots, [\![r_d]\!] \rangle = \langle [\![1]\!], [\![1]\!], \ldots, [\![1]\!] \rangle$$

   (b) Otherwise, Part is either an *internal partition* or a *leaf partition*:

      i. Determine the set of TaxTree nodes DifNodes that exists in Part.$HCut$ but not in Parent(Part).$HCut$.

      ii. For each record $\mathsf{R}_j \in$ MixDS for $1 \leq j \leq d$, the participants securely determine whether R can be assigned to Part according to Definition 17 by utilizing the functions from Construction 5.1. If all conditions in Definition 17 are satisfied, then ciphertext $Recs\langle j \rangle$ is set to $[\![1]\!]$; otherwise, it is set to $[\![0]\!]$:

      ◦ $\mathsf{A} :=$ Parent(Part).$Recs\langle j \rangle$,
      ◦ $\mathsf{B} := \mathsf{n\text{-}OR}(\mathsf{CV}(\alpha_1)), \mathsf{n\text{-}OR}(\mathsf{CV}(\alpha_2)), \ldots, \mathsf{n\text{-}OR}(\mathsf{CV}(\alpha_{|\mathsf{DifNodes}|}),$
      ◦ $\mathsf{C} := \mathsf{n\text{-}NOT}(\mathsf{n\text{-}OR}(\mathsf{CV}(\beta_1)), \mathsf{n\text{-}OR}(\mathsf{CV}(\beta_2)), \ldots, \mathsf{n\text{-}OR}(\mathsf{CV}(\beta_{|CCut|}))$

      $$Recs\langle j \rangle = [\![r_j]\!] := \mathsf{n\text{-}AND}(\mathsf{A}, \mathsf{B}, \mathsf{C}) \tag{4}$$

      where $\mathsf{DifNodes} = \{\alpha_1, \alpha_2, \ldots, \alpha_{|\mathsf{DifNodes}|}\}$, $CCut = \{\beta_1, \beta_2, \ldots, \beta_{|CCut|}\}$, and $\mathsf{CV}(N)$ is a function that returns the ciphertexts from record R_j corresponding to the items represented by the leaf nodes from the subtree of TaxTree rooted at node N.

2. Securely compute the true number of records assigned to Part by homomorphically adding together all ciphertexts in $Recs$:

   $$[\![TCount]\!] := \sum_{j=1}^{d} Recs\langle j \rangle = [\![r_1]\!] + [\![r_2]\!], \ldots, [\![r_d]\!]$$

3. All participants evaluate the function LapNoise from Sub-Protocol 2.2 to generate an encrypted random noise $[\![LNoise]\!]$ satisfying Laplace distribution, and then homomorphically add the result to $[\![TCount]\!] \times 10^D$ to determine the encrypted noisy count $[\![NCount]\!]$:

   $$[\![NCount]\!] := [\![TCount]\!] \times 10^D + \mathsf{LapNoise}(Budget) : \; Budget := \begin{cases} \mathsf{Part}.PBudget & : \mathsf{Part} \text{ is internal partition} \\ \mathsf{Part}.ABudget & : \mathsf{Part} \text{ is leaf partition} \end{cases}$$

4. All participants jointly decrypt $[\![NCount]\!]$, and then compute $NCount \times 10^{-D}$ to determine the decimal and differentially private noisy count.

5. If Part is an internal partition and $NCount$ is greater than or equal to the noise size threshold $thres_1 = \sqrt{2}C \times height(HCut)/PBudget$, where C is a constant used for pruning empty partitions as early as possible [CMF+11]:

   (a) Randomly select a partitioning node $PNode$ from the set of non-leave nodes in $HCut$ with the largest height in TaxTree. Let $l \leq f$ be the number of child nodes of $PNode$ in TaxTree, then there are 2^l possible combinations of child nodes: $Combs = \{comb_1, comb_2, \ldots, comb_{2^l}\}$.

   (b) Expand Part over $PNode$ by generating $2^l - 1$ non-overlapping child partitions such that for any child partition $\mathsf{CPart}_i : 1 \leq i \leq 2^l$:

   ◦ $HCut := \{\mathsf{Parent}(\mathsf{CPart}_i).HCut \setminus PNode\} \cup comb_i$
   ◦ $CCut := Combs \setminus comb_i$
   ◦ $ABudget := \mathsf{Parent}(\mathsf{CPart}_i).ABudget + L : L := \begin{cases} -\mathsf{Parent}(\mathsf{CPart}_i).PBudget & : \mathsf{CPart}_i \text{ is internal partition} \\ \varepsilon/2 & : \mathsf{CPart}_i \text{ is leaf partition} \end{cases}$

   (c) Recursively apply Sub-Protocol 2.1 starting from Step 1 on each generated child partition $\mathsf{CPart}_i = \mathsf{Part}$.

6. Otherwise, if Part is an internal partition but $NCount < thres_1$, then Part is considered *empty partition*, $NCount := 0$, and no further partitioning on it is performed.

7. Terminate the partitioning process once no further partitioning is possible on any partition.

Sub-Protocol 2. 1: PartTree Construction

Secure Data Integration

The purpose of this protocol is to securely integrate the datasets from all participants into one encrypted dataset:

1. Each participant $\mathcal{P}_k : 1 \leq k \leq p$ encrypts his dataset DS_k under the data owners' distributed public key by representing each transaction T_j^k as a binary vector of n ciphertexts:

$$\mathsf{EncT}_j^k = \langle [\![x_{j,1}^k]\!], [\![x_{j,2}^k]\!], \ldots, [\![x_{j,n}^k]\!] \rangle : [\![x_{j,i}^k]\!] = \begin{cases} [\![0]\!] & : I_i \notin \mathsf{T}_j^k \\ [\![1]\!] & : I_i \in \mathsf{T}_j^k \end{cases}$$

for $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant |\mathsf{DS}_k|$. The outputs are encrypted datasets $\{\mathsf{EncDS}_1, \mathsf{EncDS}_2, \ldots, \mathsf{EncDS}_p\}$, each of which represents a binary matrix of ciphertexts with size $|\mathsf{DS}_k| \times n$. $\mathcal{P}_k$ also proves knowledge of each ciphertext using Non-Interactive Zero-Knowledge Proofs [FS87].

2. The procedure for merging together the encrypted datasets depends on the partitioning nature of the underlying data:

   (a) **Vertically Partitioned Data**. The function Merge is applied to merge EncDS_1, EncDS_2, …, EncDS_p together into one encrypted dataset EncDS by applying the function $\mathsf{n\text{-}OR}$ on the corresponding ciphertexts from the input datasets:

$$\mathsf{Merge}(\mathsf{EncDS}_1, \mathsf{EncDS}_2, \ldots, \mathsf{EncDS}_p) = \mathsf{EncDS} = \begin{bmatrix} [\![x_{1,1}]\!] & [\![x_{1,2}]\!] & \ldots & [\![x_{1,n}]\!] \\ [\![x_{2,1}]\!] & [\![x_{2,2}]\!] & \ldots & [\![x_{2,n}]\!] \\ \vdots & \vdots & \ddots & \vdots \\ [\![x_{d,1}]\!] & [\![x_{d,2}]\!] & \ldots & [\![x_{d,n}]\!] \end{bmatrix}$$

   where $[\![x_{j,i}]\!] = \mathsf{n\text{-}OR}([\![x_{j,i}^1]\!], [\![x_{j,i}^2]\!], \ldots, [\![x_{j,i}^p]\!])$ for $1 \leqslant j \leqslant d$ and $1 \leqslant i \leqslant n$.

   (b) **Horizontally Partitioned Data**. The encrypted datasets EncDS_1, EncDS_2, …, EncDS_p are appended together to construct EncDS such that $|\mathsf{EncDS}| = \sum_{k=1}^{p} |\mathsf{EncDS}_k|$.

3. Output EncDS, which has the same structure regardless of whether the underlying data is horizontally or vertically partitioned.

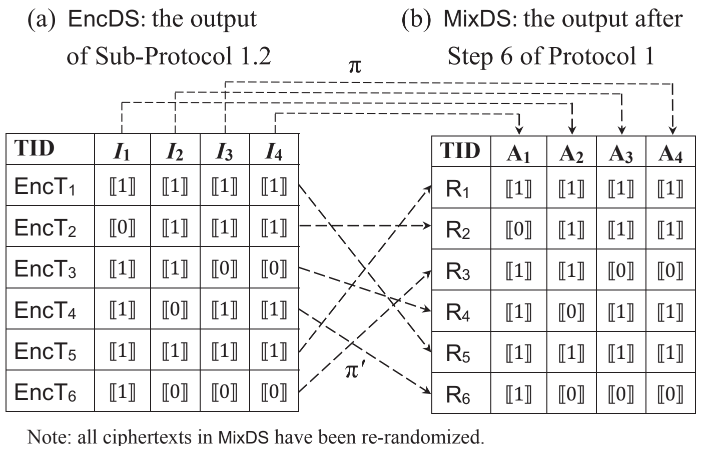
Sub-Protocol 1. 2: Secure Data Integration

5.3.6 Secure Distributed Differentially-private Set-valued Data Release Protocol

Given the integrated dataset MixDS, the context-free taxonomy tree $\mathsf{TaxTree}$, and a privacy budget ε, the goal is to securely generate an encrypted and sanitized set-valued data OutDS from MixDS that satisfies ε-differential privacy.

The general idea of our solution is to initially assign all records in MixDS to a data structure called *partition*, and then apply a top-down partitioning process using $\mathsf{TaxTree}$ to recursively assign

Table 8: Integrated dataset of the sample datasets in Table 6



(a) EncDS: the output of Sub-Protocol 1.2

TID	I_1	I_2	I_3	I_4
EncT$_1$	⟦1⟧	⟦1⟧	⟦1⟧	⟦1⟧
EncT$_2$	⟦0⟧	⟦1⟧	⟦1⟧	⟦1⟧
EncT$_3$	⟦1⟧	⟦1⟧	⟦0⟧	⟦0⟧
EncT$_4$	⟦1⟧	⟦0⟧	⟦1⟧	⟦1⟧
EncT$_5$	⟦1⟧	⟦1⟧	⟦1⟧	⟦1⟧
EncT$_6$	⟦1⟧	⟦0⟧	⟦0⟧	⟦0⟧

(b) MixDS: the output after Step 6 of Protocol 1

TID	A$_1$	A$_2$	A$_3$	A$_4$
R$_1$	⟦1⟧	⟦1⟧	⟦1⟧	⟦1⟧
R$_2$	⟦0⟧	⟦1⟧	⟦1⟧	⟦1⟧
R$_3$	⟦1⟧	⟦1⟧	⟦0⟧	⟦0⟧
R$_4$	⟦1⟧	⟦0⟧	⟦1⟧	⟦1⟧
R$_5$	⟦1⟧	⟦1⟧	⟦1⟧	⟦1⟧
R$_6$	⟦1⟧	⟦0⟧	⟦0⟧	⟦0⟧

Note: all ciphertexts in MixDS have been re-randomized.

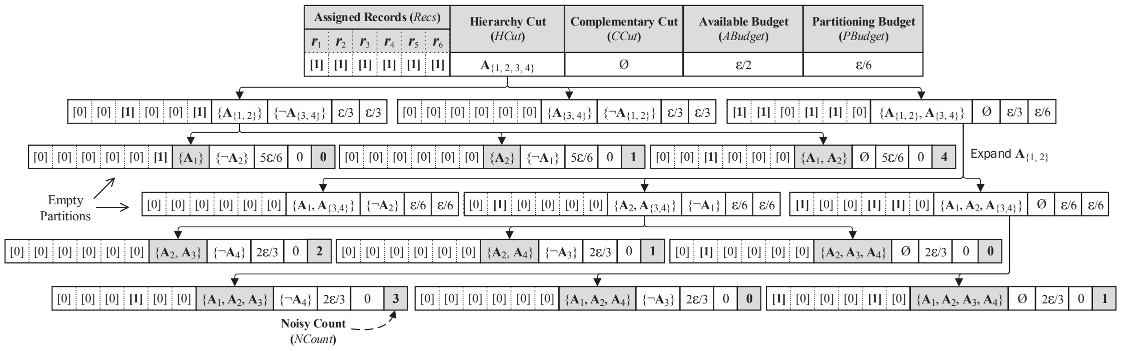
Secure Distributed Differentially Private Set-Valued Data Release

1. Construct PartTree according to Sub-Protocol 2.1.
2. For each leaf partition Part in PartTree:
   (a) If $NCount$ is greater than or equal to the noise size threshold $thres_2 = \sqrt{2}C'/ABudget$, where C' is a constant for minimizing the effect of noise agreed upon by all participants:
      i. All participants jointly run the Distributed Proxy Re-encryption protocol (it doesn't use a proxy despite the name) to re-encrypt each item $A_i = [\![I_{\pi(i)}]\!] \in HCut$ and the noisy count $NCount$ under the data miner's public key without revealing the underlying values to any of the participants.
      ii. Add the pair of re-encrypted $HCut$ and $NCount$ to the output dataset OutDS.
   (b) Otherwise, if $NCount < thres_2$, then Part is considered *empty partition* such that $NCount := 0$.
3. Release the sanitized dataset OutDS to the data miner over a secure channel.

Protocol 5.2: Secure Distributed Differentially Private Set-Valued Data Release

the records into disjoint partitions in a noisy way until no further partitions can be created. The output of the partitioning process is a partitioning tree PartTree, where the root partition is called the *initial partition*, the partitions with at least one child partition are called *internal partitions*, and the partitions with no child partitions are called *leaf partitions*. Next, we formally define the structure of a partition.

Definition 16 ***Partition.*** A partition Part is a tuple:



- The partitioning noisy counts are not shown. Only the noisy counts of the leaf nodes are displayed.
- The complementary cut (*CCut*) is shown in negative form indicating that any record assigned to the partition should not satisfy the negative cut.
- The gray fields *HCut* and *NCount*, in the leaf nodes, indicate the differentially private data (OutDS) to be returned to the data miner.

Figure 15: PartTree: ε-differentially private partitioning tree of MixDS.

$[HCut, CCut, Recs, ABudget, PBudget, NCount]$, where:

- Hierarchy cut ($HCut$) and complementary cut ($CCut$) are two set of nodes from TaxTree.
- $Recs = \langle [\![r_1]\!], [\![r_2]\!], \ldots, [\![r_d]\!] \rangle$ is a binary vector of d ciphertexts, where $d = |\text{MixDS}|$. If a record $R_j \in \text{MixDS}$ for $1 \leq j \leq d$ is assigned to Part, then $[\![r_j]\!]$ is set to $[\![1]\!]$. All ciphertexts in $Recs$ are initially set to $[\![0]\!]$.
- $ABudget$ is the available privacy budget.
- $PBudget = ABudget/|\text{InterNodes}(HCut)|$ is the partitioning privacy budget, where $|\text{InterNodes}(HCut)|$ is the total number of all non-leaf nodes from each subtree of TaxTree rooted at each node in $HCut$.
- $NCount$ is the noisy count generated using Laplace mechanism. □

Jointly executed by all data owners, Sub-Protocol 2.1 illustrates how the partitioning tree PartTree is constructed in a differentially private manner. When the initial partition is created, only half of the global privacy budget ε is allocated to it to guide the partitioning process. The other half of the privacy budget will be assigned to each leaf partition to generate its noisy count. When a partition is expanded over a node from its hierarchy cut $HCut$, $2^l - 1$ child partitions are created, where $l < f$ is the number of child nodes of the partitioning node from TaxTree. The same available privacy budget is allocated to each child partition, which is possible due to the *parallel composition property* [McS09] since the child partitions will be assigned disjoint set of records from MixDS.

Example 10 Figure 15 illustrates the partitioning tree PartTree of dataset MixDS from Table 8.b. Let Part be the partition where $HCut = \{A_{\{1,2\}}, A_{\{3,4\}}\}$, $Recs = \langle [\![1]\!], [\![1]\!], [\![0]\!], [\![1]\!], [\![1]\!], [\![0]\!] \rangle$, $ABudget = \varepsilon/3$, and $PBudget = (\varepsilon/3)/2 = \varepsilon/6$, where $|\mathsf{InterNodes}(\{\ A_{\{1,2\}},\ A_{\{3,4\}}\ \})| = 2$.

Since internal nodes $\{A_{\{1,2\}}$ and $A_{\{3,4\}}\}$ are at the same height in TaxTree, either node can be used for partitioning (expanding) Part. Assume that we randomly select $A_{\{1,2\}}$ which has $l = 2$ children in TaxTree, then a maximum of $2^l - 1 = 3$ partitions can be created. As illustrated in Figure 15, $\mathsf{Part}_1.HCut = \{A_1, A_{\{3,4\}}\}$, $\mathsf{Part}_2.HCut = \{A_2, A_{\{3,4\}}\}$, $\mathsf{Part}_3.HCut = \{A_1, A_2, A_{\{3,4\}}\}$, and $\mathsf{Part}_1.ABudget = \mathsf{Part}_2.ABudget = \mathsf{Part}_3.ABudget = \mathsf{Part}.ABudget - \mathsf{Part}.PBudget = \varepsilon/3 - \varepsilon/6 = \varepsilon/6$. □

The data owners utilizes the functions from Construction 5.1 on each record from MixDS to securely determine which record should be assign to a partition. The following definition specifies the conditions for successful record assignments.

Definition 17 ***Record-Partition Assignment.*** A record R can be assigned to a non-initial partition Part if the following conditions hold:

1. Every item in DifNodes exists in R, where DifNodes is the set of nodes from TaxTree that exist in $\mathsf{Part}.HCut$ but not in the hierarchy cut of its parent $\mathsf{Parent(Part)}.HCut$.
2. Every item in $\mathsf{Part}.CCut$ does not exist in R.
3. R has already been assigned to Parent(Part). □

Note that $HCut$ could have been used instead of DifNodes in the first condition of Definition 17. However, verifying the nodes in DifNodes is sufficient since the remaining nodes that exist in $HCut$ but not in DifNodes will be automatically verified by the third condition of the definition. This optimization eliminates the need to verify each bit in R, where $|\mathsf{R}| = |\mathsf{Univ}| = n$, against Part in order to determine if R should be assigned to Part.

Distributed Laplace Noise Generation

$LNoise \leftarrow \mathsf{LapNoise}(Budget)$:

1. Determine the partial noise $[\![X_k]\!]$ from each participant $\mathcal{P}_k : 1 \leq k \leq p$:
    (a) All participants run the Multiparty Coin Toss protocol to generate a uniformly random bitstring s.
    (b) $\mathcal{P}_k$ generates l independent and identically gamma-distributed random variables deterministically derived from s:
    $\gamma_{i,k} = \mathsf{Gamma}(p, 1/Budget)$ for $1 \leq i \leq l$
    (c) $\mathcal{P}_k$ convert each gamma variable $\gamma_{i,k}$ into integer value $\bar{\gamma}_{i,k}$:
    $\bar{\gamma}_{i,k} = \gamma_{i,k} \times 10^D$, where D is the number of decimal places (precision) agreed upon by all parties.
    (d) $\mathcal{P}_k$ encrypts each integer value and broadcasts the set of ciphertexts:
    $\Gamma_k = \{[\![\bar{\gamma}_{1,k}]\!], [\![\bar{\gamma}_{2,k}]\!], \ldots, [\![\bar{\gamma}_{l,k}]\!]\}$
    (e) The rest of the participants use a random beacon to randomly choose $(l-2)$ ciphertexts from Γ_k.
    (f) All participants jointly decrypt all chosen ciphertexts, and verify that Steps 1b-d are correct.
    (g) If the test is successful, then the partial random noise from $\mathcal{P}_k$ is homomorphically computed as follows:
    $[\![X_k]\!] = [\![\bar{\gamma}_{i',k}]\!] - [\![\bar{\gamma}_{i'',k}]\!]$, where $[\![\bar{\gamma}_{i',k}]\!]$ and $[\![\bar{\gamma}_{i'',k}]\!]$ are the two ciphertexts that were not chosen in Step 1e.
2. Homomorphically compute the encrypted Laplace noise $[\![LNnoise]\!]$ (multiplied by 10^D):

$$[\![LNoise]\!] = \sum_{k=1}^{p} [\![X_k]\!]$$

Sub-Protocol 2. 2: Laplace Noise Generation

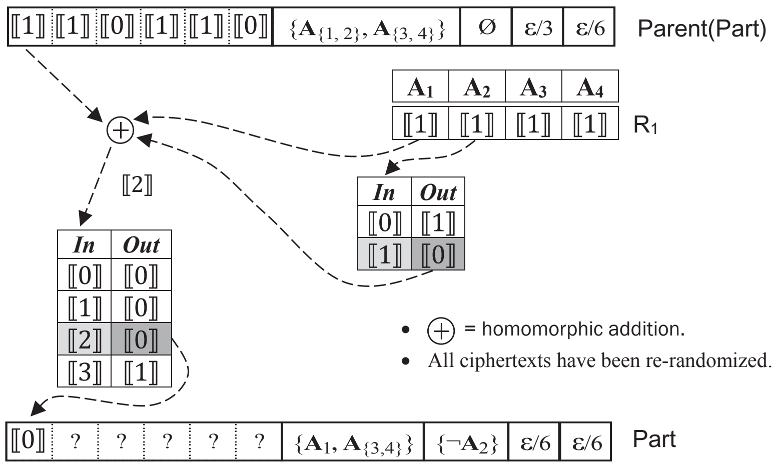


Figure 16: Using Mix and Match to verify whether record R_1 should be assigned to partition Part.

Example 11 Let Part be the partition from Figure 15, where:

Part.$HCut = \{A_1, A_{\{3,4\}}\}$ and Part.$CCut = \{A_2\}$. Consequently, DifNodes := $\{A_1, A_{\{3,4\}}\} \backslash \{A_{\{1,2\}}, A_{\{3,4\}}\} = \{A_1\}$.

Giving record R_1 from Table 8.b, Figure 16 illustrates how to use the Verifiable Mix and Match protocol to securely verify whether record R_1 should be assigned to partition Part. □

Since the rows in MixDS and items in TaxTree are both encrypted (randomized) and shuffled, data owners cannot use Part.$HCut$ or Part.$CCut$ to infer which records is assign to which partition in each level of PartTree. Recall that $\mathbf{A}_1$ corresponds to ciphertext $[\![I_{\pi(1)}]\!]$, $\mathbf{A}_2$ corresponds to ciphertext $[\![I_{\pi(2)}]\!]$, etc.

Distributed Laplace Noise Generation. To verify in a *non-deterministic* way whether or not a partition is empty, a count query is issued for its noisy count by homomorphically adding a Laplace noise to the true number of records assigned to the partition. To securely generate Laplace noise in a distributed fashion, the data owners run Sub-Protocol 2.2, which is based on the following Lemma.

Lemma 1 ***Laplace Distribution [KKP01].*** *Let $Lap(\lambda)$ be a random variable satisfying a Laplace distribution with probability distribution function $Pr(x|\lambda) = \frac{1}{2\lambda}e^{\frac{|x|}{\lambda}}$. Then:*

1. The distribution of $Lap(\lambda)$ is infinitely divisible.

2. $\forall r \geq 1$, *$Lap(\lambda) = \sum_{i=1}^{r}[\gamma_{1,i} - \gamma_{2,i}]$, where $\gamma_{1,i} = \mathsf{Gamma}(r, \lambda)$ and $\gamma_{2,i} = \mathsf{Gamma}(r, \lambda)$ are independent and identically distributed random variables satisfying gamma distribution with probability distribution function $Pr(x|k, \lambda) = \frac{(1/\lambda)^{1/k}}{\Gamma(1/k)} x^{\frac{1}{k}-1} e^{\frac{-x}{\lambda}}$, where $x \geq 0$ and $\Gamma(1/k)$ is the gamma function evaluated at $1/k$.* □

The general idea is to take advantage of the infinite divisibility of Laplace distribution, and have each data owner generate a random partial noise based on two gamma-distribution random variables. According to Lemma 1, the shape parameter $r = p$ (number of parties), and the inverse scale parameter $\lambda = \frac{1}{privacy\ budget}$. Several algorithms exist in the literature for generating random gamma variables, e.g. [MT00] and [AD74], where at least two variables must be drawn from the uniform distribution in order to generate one gamma variable. Cut-and-Choose is used to ensure that each data owner has used the bitstring (randomness) generated by the Multiparty Coin Tossing protocol to construct its gamma variables. All partial noises are converted to integers according to the decimal precision value D agreed upon by all parties, before they are encrypted using Exponential ElGamal. The encrypted noises are then homomorphically added together to construct a random variable satisfying Laplace distribution that is multiplied by 10^D.

5.4 Protocol Analysis

5.4.1 Complexity Analysis

Proposition 6 ***Complexity.*** *The average runtime complexity of our proposed protocol is bounded by $\mathcal{O}(\log_f n \times d^2)$ operations, where d is the number of records and n is the number of columns in* EncDS.

Proof. The generation of EncDS from Sub-Protocol 1.2 requires $\mathcal{O}(n \times d)$. The shuffling of EncDS also requires $\mathcal{O}(n \times d)$ [SHKS12]. As a result, Protocol 5.1 takes $\mathcal{O}(n \times d)$ operations to generate MixDS. The shuffling of n items in Univ and the construction of TaxTree with fan-out f requires $\mathcal{O}(n)$ operations. In Sub-Protocol 2.1, 2^n distinct partitions might be created in the worst-case scenario. However, since it is impossible in practice to have $|\mathsf{MixDS}| = d = 2^n$ records to fill out all the partitions, we argue that the average-case complexity reflects a more accurate measure of our protocol's performance. Given that the mean of Laplace distribution is 0, the noise in the average case is cancelled out, while assuming that the records are assign evenly between all partitions at the same level in PartTree (worst case). The number of levels in PartTree is $(\log_2 d/f)$, and the total number of partitions in all levels is $\sum_{i=0}^{\log_2 d/f} 2^{i \times f} = \mathcal{O}(d)$, where $\mathcal{O}(\log_f n)$ operations are applied on each partition. Since all records in MixDS are validated against each partition in PartTree for assignment, then the required number of operations is $\mathcal{O}(\log_f n \times d^2)$. ■

Discussion. The analysis shows that our approach is suitable for high-dimensional data since the complexity is logarithmic with regard to the number of dimensions. On the other hand, it is quadratic with respect to the number of records. This is due to the security protections of our protocol, where no information can be inferred by malicious adversaries either during data integration or during the partitioning process. Lowering record complexity while maintaining the same level of security is a non-trivial open problem.

5.4.2 Security Analysis

Proposition 7 ***Integrity.*** *The overall protocol is sound under the malicious adversarial model.*

Proof. All steps in our solution are publicly verifiable, which prevents a compromised data owner from deviating from the correct computation without detection. If detected, honest data owners will not proceed, preventing the completion of the protocol (as the decryption operations throughout the protocol, including the last step, require all participants). Table 9 illustrates the publicly verifiable primitive of each security-sensitive step in each proposed protocol and sub-protocol. We inherit integrity against a dishonest majority from our building blocks (and we can provide robustness against dishonest minority by adjusting the threshold of the decryption operation).

We must also ensure that all inputs to the protocol are correctly formed. In the setup phase, where the data owners interact together to construct the public key, the distributed key generation (DKG)

Table 9: The publicly verifiable primitives involved in each security-sensitive step of the proposed protocol

P. V. Primitive	**Construction 1**	**Protocol 1**	**Sub-Protocol 1.1**	**Sub-Protocol 1.2**	**Protocol 2**	**Sub-Protocol 2.1**	**Sub-Protocol 2.2**
Mix Network		$\mathbf{1, 4, 5}^{\dagger}$					
Mix and Match	$\mathbf{1-4}$					**1.b**	
NIZKP			**1**				
Homomorphic Operation			**2**			$\mathbf{2, 3}$	**2**
Public Encryption						$\mathbf{1.a}^{\ddagger}$	
Distributed Decryption						**4**	
Cut-and-Choose							**1**
Distributed Proxy Re-encryption					**2.a.i**		
Cleartext Operation*		$\mathbf{2, 3, 6}$	**3**	$\mathbf{1-4}$	**1, 2.a.ii, 2.b, 3**	$\mathbf{5-7}$	

† The shuffling in Step 2 and 5 is performed at the same time to ensure that the same random permutation π is used.
‡ All participants agree on one randomness value for encryption so that anyone can verify the ciphertexts by regenerating them.
$*$ Cleartext operations involve steps that do not require a secret, such as sub-protocol calls and broadcasting an output.

protocol ensures that the output is uniformly distributed at random [GJKR07]. In the case of data encryption, each ciphertext must be from $\langle \mathbb{G}_q \times \mathbb{G}_q \rangle$ such that the data owners are able to check the independency of the ciphertexts. When operations of mixing (shuffling & re-randomization), distributed proxy re-encryption, and plaintext equality test are performed, each data owner inputs a random exponent from $\mathbb{Z}_q^*$ for blinding. As long as there is at least one exponent that is uniformly distributed at random, the addition of all exponents is also random. Finally, during the generation of a uniformly random bitstring using the Coin Toss protocol, the same property holds: as long as there is at least one honest data owner, then the result is uniformly random. ■

Proposition 8 ***Privacy-preserving.*** *The overall protocol is privacy-preserving.*

Proof. To prove that our protocol is privacy-preserving, we show that the data is protected throughout the protocol execution.

Input Data. Each data owner $\mathcal{P}_i$ encrypts his data, proves knowledge of it, and then inputs it to the protocol. The proof is *zero-knowledge*, where $\mathcal{P}_i$ proves that he knows the underlying plaintexts of the encrypted data without revealing any information about the plaintexts.

Encrypted Data. While encrypted, the data is protected under the CPA-security of the encryption scheme (e.g., DDH for ElGamal) and the proof is zero-knowledge. The adversary cannot decrypt items arbitrarily, as the decryption key is (n, n)-shared between all data owners, requiring the adversary to corrupt every data owner to be successful (in which case, all the inputs are already known). Moreover, applying verifiable mixing on the columns and rows of the encrypted data removes any correspondence between ciphertexts and the original items/records.

Decrypted Data. The underlying data remains encrypted throughout the protocol except in two areas: within Mix and Match (during plaintext equality tests) and within proxy re-encryption. However, both subprotocols are verifiable and already provide protection against a malicious adversary. ■

5.4.3 Correctness and Utility Analysis

Proposition 9 ***Correctness.*** *Given $p \geq 2$ set-valued datasets with record and item overlaps, the proposed protocol generates ε-differentially private set-valued data.*

Proof. We first show that our protocol can handle record and item overlaps, and then show that the released data is ε-differentially private.

Data Overlap. If data about the same individual exists in more than one dataset (record overlap), then Step 2.a of Sub-Protocol 1.2 is applied to generate *one* integrated record for that individual. Function n-OR is used to set the total number of occurrences of an item to *one* if the item exists in more than one record for the same individual (item overlap).

ε-Differentially Private Data Generation. Sub-Protocol 2.1 performs the same sequence of partitioning operations as the algorithm in [CMF$^+$11], except that our protocol is in a distributed setting. Since [CMF$^+$11] generates ε-differentially private set-valued data, we prove the correctness of Sub-Protocol 2.1 by only proving the correctness of the different steps:

- *Record-Partition Assignment.* Verifying that every item in DifNodes exists in R, and R has already been assigned to Parent(Part), is equivalent to verifying that every item in $HCut$ exists in R. Moreover, verifying that every item in Part.$CCut$ does not exist in R ensures that the same record is not assigned to more than one sibling partition.

- *Noise Generation.* In Sub-Protocol 2.2, even though $LNnoise$ is differentially private, the partial noise $[\![X_k]\!]$ from $\mathcal{P}_k$ is not; hence the use of Cut-and-Choose protocol to allow for encrypted partial noises while ensuring they are random variables satisfying gamma distribution. Moreover, since the total noise $LNnoise$ of a leaf partition is equal to $\mathsf{LapNoise}(\varepsilon/2)$, the output is guaranteed to be $\varepsilon/2$-differentially private.

■

5.5 Performance Evaluation

5.5.1 Experimental Results

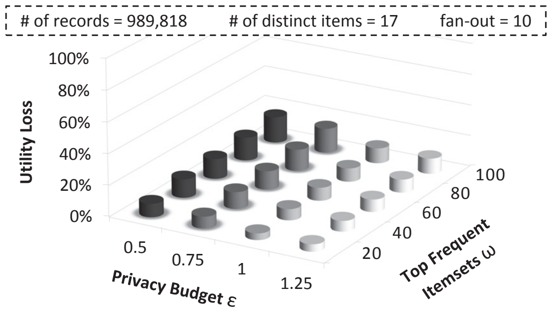
In this section, we verify the utility loss of the anonymized dataset OutDS based on frequent itemset mining, a common data mining technique for extracting knowledge from set-valued data.

We utilize the publicly available dataset *MSNBC* [4], a real-life web log dataset that have been used for testing several data mining approaches [LQSC12] [EVK05]. It consists of $989,818$ records and 17 distinct items, where the average number of items per record is 1.72 items. We also utilize STM[5], a real-life dataset that lists the subway and bus stations visited by passengers in Montréal city within a week. It consists of $1,210,096$ records and $1,012$ distinct items, where the average number of items per record is 64 items.

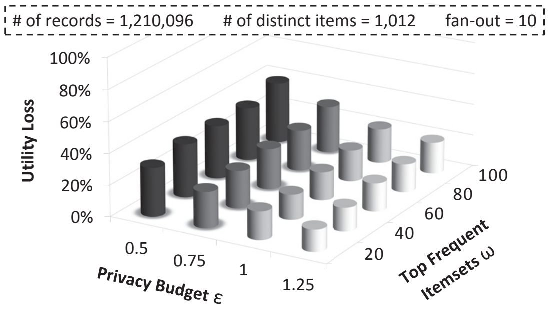
Let ω be the number of top frequent itemsets. Our goal is to compute the sets $\mathcal{F}_\omega(\mathsf{EncDS})$ and $\mathcal{F}_\omega(\mathsf{OutDS})$ that contains the ω top frequent itemsets from EncDS and OutDS, respectively. We measure the similarity between the two sets in order to determine the utility loss (UL) in OutDS dataset due to anonymization as follows:

$$UL(\mathsf{OutDS}) = \frac{1}{\omega} \times \sum_{I_i \in \mathcal{F}_\omega(\mathsf{OutDS})} 1 - \frac{Sup(I_i, \mathcal{F}_\omega(\mathsf{OutDS}))}{Sup(I_i, \mathcal{F}_\omega(\mathsf{EncDS}))},$$

[4]MSNBC dataset: https://archive.ics.uci.edu/ml/datasets.html
[5]STM dataset: http://www.stm.info/en/



(a) MSNBC dataset



(b) STM dataset

Figure 17: Utility loss for frequent itemset mining with respect to privacy budget and top frequent itemsets.

where $Sup(I_i, \mathcal{F}_\omega(\mathsf{OutDS}))$ and $Sup(I_i, \mathcal{F}_\omega(\mathsf{EncDS}))$ represent the support of frequent itemset $I_i \in \mathcal{F}_\omega(\mathsf{OutDS})$ in $\mathcal{F}_\omega(\mathsf{OutDS})$ and $\mathcal{F}_\omega(\mathsf{EncDS})$, respectively.

Figure 22 illustrates the utility loss for frequent itemset mining with respect to different privacy budgets and different number of top frequent itemsets ω. We observe that for both datasets: MSNBC in Figure 17a and STM in Figure 17b, the utility loss is directly affected by the the privacy budget. That is, the more privacy budget is allocated, the less the utility loss is. This is because a higher privacy budget leads to a more accurate partitioning, and less noise is added to the count of each partition at the leaf level. On the other hand, with regard to ω, we observe that the more top frequent itemsets we consider, the more the utility loss is. This due to the fact that considering more top frequent itemsets leads to more false positives in OutDS, as well as to more false negatives in the itemsets being dropped. However, even when $\omega = 100$, the utility of our approach is over 60% in both datasets, and over 70% except in STM dataset when $\varepsilon = 0.5$.

5.5.2 Cost Estimation

We assume all ciphertexts are generated using exponential ElGamal. We count the modular exponentiation operations to estimate the cost of the protocol. We only report our cost analysis of Protocol 2.1 as it dominates the analysis. Modular exponentiations are mainly required when a partition is being constructed and when a record is validated against a partition.

- *Partition Construction.* The total number of modular exponentiations required for generating one ciphertext is 5, where 2 exponentiations are required to perform the operation, 1 to generate the proof, and 2 to verify the proof. Given that each partition consists of d binary ciphertexts, where $d = |\mathsf{MixDS}|$, the total cost of constructing a partition is $5 \times d$.

- *Record-Partition Assignment.* To determine the cost of assigning a record to a partition, we calculate the modular exponentiations required by Equation 4 in Protocol 1. More specially, we calculate the modular exponentiations required by the plaintext equality test (PET) on the Mix and Match truth tables, assuming that all tables were created offline. Given p data owners, one PET requires $15 \times p$ exponentiations, where $3 \times p$ are required to perform the operation, $4 \times p$ to generate the proof, and $8 \times p$ to verify the proof [SHKS12]. Table 10 illustrates the average number of modular exponentiations for the Mix and Match truth tables, where each table contains two columns, and PET is performed on the ciphertexts in the first column.

  Table 10: Average cost of PET on Mix and Match truth tables

Table	**# of Rows**	**Modular Exponentiations**
$\mathsf{n\text{-}AND}(V)$	$\lvert V\rvert + 1$	$\frac{\lvert V\rvert+1}{2} \times 15 \times p$
$\mathsf{n\text{-}OR}(V)$	$\lvert V\rvert + 1$	$\frac{\lvert V\rvert+1}{2} \times 15 \times p$
$\mathsf{NOT}(\llbracket v \rrbracket)$	2	$\frac{2}{2} \times 15 \times p = 15 \times p$
$\mathsf{n\text{-}NOT}(V)$	$2 \times \lvert V\rvert$	$\frac{2\times\lvert V\rvert}{2} \times 15 \times p = 15 \times \lvert V\rvert \times p$

  The average number of $\mathsf{n\text{-}OR}(V)$ operations in part B or C of Equation 4 is $f/2$. Since the number of items in $\mathsf{DifNodes}$ decreases by f times at every partitioning step, the function $\mathsf{CV}(N)$ returns on average $\log_f n$ ciphertexts. So the cost of Equation 4 in terms of modular exponentiations is $(f \times (\frac{(\log_f n)+1}{2} \times 15 \times p) + 15 \times \frac{f}{2} \times p + \frac{(f+1)+1}{2} \times 15 \times p) \backsimeq (8 \times \log_f n + 24) \times f \times p$.

Given that all records will be evaluated against each partition, and $2d$ partitions are expected to be created on average, the total cost for constructing $\mathsf{PartTree}$ is $(16 \times \log_f n + 58) \times f \times p \times d^2$. Assuming that the cost for one exponential operation is 2 ms, Table 11 and Table 12 illustrate the estimated runtime of our protocol w.r.t. the number of records and the number of distinct items (universe size), respectively.

Table 11: Estimated Performance of Protocol 2.1 (Hours) w.r.t. # of records, where # of parties is 3 and fan-out is 2.

Dataset	**200**	**400**	**600**	**800**	**1000**
MSNBC	16	66	148	263	411
STM	29	116	261	464	726

Table 12: Estimated Scalability of Protocol 2.1 (Hours) w.r.t. # distinct items, where # of records is 600, # of parties is 3 and fan-out is 2.

20K	**40K**	**60K**	**80K**	**100K**
344	363	374	382	389

We observe from Table 11 that our protocol can be practical when the number of records is relatively small, while the number of dimensions is high. It shows, for instance, that integrating 3 datasets with up to 400 records and 1012 distinct items takes less than 5 days (116 hours) to complete. One example where our protocol can be utilized is in medical research labs that perform tests on new drugs and need to integrate their data in a privacy-preserving manner. In such case, the number of participants is typically small, whereas the number of symptoms to be tracked is large. We also observe from Table 12 that our proposed protocol grows at a logarithmic rate with respect to the number of dimensions. It shows that when the number of dimensions is 20,000, and then gradually increases by 20,000 distinct items at each step, up to 100,000 dimensions, the average increase of runtime per step is only 8.25 hours. Recall that the costs are expensive due to the support of public verifiability. We hope that this work inspires researchers to further study the problem of data integration with public verifiability and propose more efficient and practical solutions.

5.6 Summary

In this chapter, we propose a protocol for secure set-valued data integration, where the output is a ε-differentially private data with an effective level of data utility for data mining. Our proposed protocol is the first in the literature to support public verifiability in the context of data integration. The protocol also supports a more general distributed data model, and is efficient for handling high-dimensional data. We show that the protocol is secure in the malicious adversarial model with dishonest majority.

Chapter 6

Secure and Privacy-preserving Relational Data Integration

6.1 Introduction

As the amount of data available from wide range of domains has increased tremendously in recent years, the demand for data sharing and integration has also risen. The integration of related data from different sources enables businesses, organizations and government agencies to perform better data analysis and make better decisions. In this chapter, we present Fusion, a protocol that enables multiple data providers to engage in a privacy-preserving integration process to generate a anonymous mashup data with high information utility for data mining tasks such as classification analysis. Throughout the integration process, a *score function* needs to be computed between the parties to guide the process. Therefore, we propose a secure protocol for evaluating the score function in a distributed setting. Figure 18 presents an example of a distributed environment for privacy-preserving data integration. The challenges of mashing-up data from different data providers in a privacy-preserving manner are summarized as follows.

A major challenge is *privacy concerns.* Data providers are often reluctant to share their data

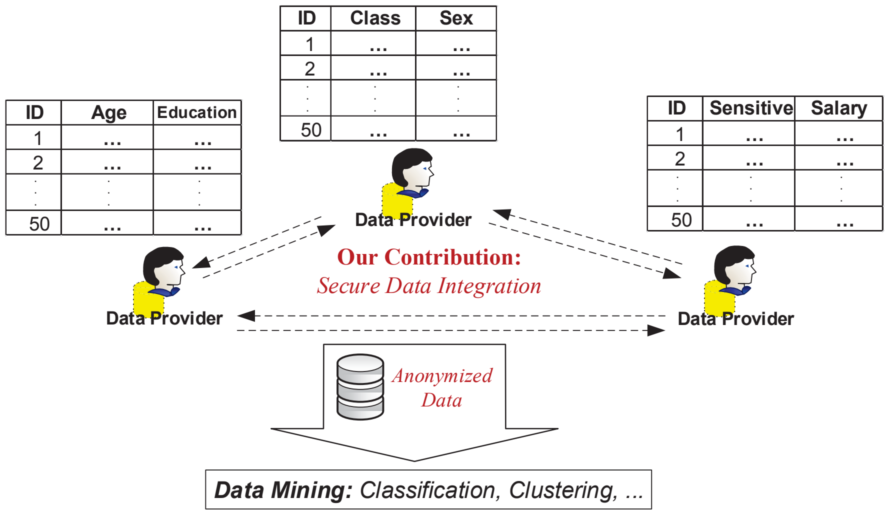


Figure 18: Privacy-preserving distributed data integration.

due to privacy concerns. We distinguish between two types of concerns. The first is to allow data providers to evaluate functions on the collective data while ensuring that no party learns more information about other parties' data, other than what is revealed in the final mashup data.

Example 12 Consider the data in Table 13, where three data providers: $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$, owns different set of attributes about the same individuals, and $\mathcal{P}_2$ owns the *Class* attribute. Assume that the parties are building a classifier and need to compute *information gain* [Qui93] for each attribute. $\mathcal{P}_2$ can directly compute the information for attribute *Sex* since it knows the class values. However, $\mathcal{P}_1$ and $\mathcal{P}_3$ should be able to compute the information for each of their attributes while the class values remain private (only known to $\mathcal{P}_2$). □

The second concern is to ensure the final mashup data is anonymized such that potential linkage attacks are thwarted. The adversary can perform two types of linkage attacks: *record linkage*, where an individual can be linked to a record if the record is very specific, and *attribute linkage*, where a frequent sensitive value can be inferred about an individual.

Example 13 In Table 13, if the adversary knows $\langle 44, 12th, Female\rangle$ about an individual, then the adversary can link the individual to record #7 and sensitive value s_2. On the other hand, if the adversary knows $\langle Bachelor, Male\rangle$, then he infers sensitive value s_2 about the individual with 67%

Table 13: Raw data owned by providers $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$

UID	**Data Provider $\mathcal{P}_1$**		**Data Provider $\mathcal{P}_2$**		**Data Provider $\mathcal{P}_3$**	
	Age	**Education**	**Class**	**Sex**	**Sen**	**Salary**
1	54	11th	Yes	Male	s_2	65K
2	26	Bachelor	No	Male	s_1	37K
3	39	7th	No	Female	s_2	51K
4	67	Master	Yes	Female	s_2	55K
5	32	Bachelor	Yes	Male	s_2	87K
6	59	Doctorate	No	Female	s_1	107K
7	44	12th	No	Female	s_2	26K
8	29	Bachelor	Yes	Male	s_2	77K
9	53	9th	No	Female	s_2	29K
10	46	Master	Yes	Female	s_1	72K

probability. □

Another major challenge is *data utility*. The anonymous mashup data should preserve as much information as possible for the targeted data mining tasks such as classification or cluster analysis. However, since each data provider can own several attributes, each of which is considered a *dimension*, the resulting mashup data is usually high-dimensional. Many anonymization approaches, such as k-anonymity [Sam01a], generate useless anonymous data when applied on high-dimensional data due to *the curse of high dimensionality* [Don00]. Therefore, choosing the appropriate anonymization approach is critical for maintaining high data quality.

The contributions of this chapter can be summarized as follows:

Contribution #1. We present a secure protocol for the distributed evaluation of an information gain-based score function, and show that the protocol is privacy-preserving.

Contribution #2. We present a multi-party protocol that applies a hierarchal approach to anonymize high-dimensional data and generate mashup data satisfying LKC-privacy [MFHL09].

Contribution #3. We performed experimental evaluation on real-life data in different distributed settings. Extensive experimental results suggest that the mashup data provides better data utility, and our approach is scalable *w.r.t.* the number of records as well as the number of attributes.

The results of this chapter have been published in [DIAF15] and [ADFH14].

6.2 Preliminaries

6.2.1 Privacy Model

In recent years, several privacy models [Swe02a][Sam01b] have been proposed to prevent linkage attacks against published data. In the problem studied in this chapter, each data provider can own many attributes and the result is often a high-dimensional mashup table. Therefore, we choose LKC-*privacy* [MFHL09], a privacy model that was originally designed for preventing linkage attacks on *high-dimensional data*, i.e., data with a large number of attributes. The authors in [MFHL09] have shown that LKC-privacy yields better utility for data mining in the anonymized data in comparison to the traditional privacy models.

Let T be an *attribute table* in the form of $T = (A_1, \ldots, A_m, Sen)$, where each record contains information about a unique individual. $QID = \{A_1, \ldots, A_m\}$ is a set of quasi-identifier attributes, such as *sex* and *age*, that *may* identify an individual if some combinations of QID values are specific enough. Sen is a sensitive attribute that contains some sensitive information about the individuals in the table, such as salaries or diseases. We assume that the an adversary looking to identify the *record* or *sensitive value* about an individual in T has a limited prior knowledge qid. More specifically, the adversary knows values from at most L attributes in QID, where $|qid| \leq L$. Given qid, the adversary can identify the set of records in T that satisfy qid, denoted by $T[qid]$, and launch two privacy attacks:

Record Linkage Attack. If the number of records $|T[qid]|$ is small, the adversary can distinguish the individual's record, and consequently, the sensitive value. For example, if $qid = \langle 44, 12th, Female \rangle$ in Table 13, then $T[qid] = \{UID\#7\}$, $|T[qid]| = 1$ and $Sen = s_2$.

Attribute Linkage Attack. If the number of records $|T[qid]|$ is large, the adversary can still infer the sensitive value s with confidence $Pr(s|qid) = \frac{|T[qid \wedge s]|}{|T[qid]|}$, where $T[qid \wedge s]$ denotes the set of records containing both qid and s. For example, if $qid = \langle Bachelor, Male \rangle$, then $T[qid \wedge s_2] = \{UID\#5, 8\}$ and $T[qid] = \{UID\#2, 5, 8\}$. Accordingly, $Pr(s_2|qid) = \frac{2}{3} = 67\%$.

To prevent such privacy attacks, LKC-privacy requires that in the anonymized table, for every

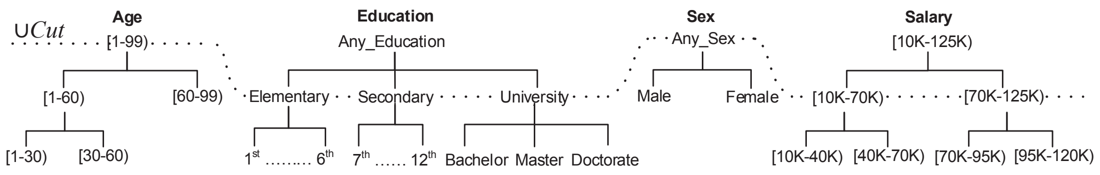


Figure 19: Taxonomy trees: $\mathbb{T}^{Age}$, $\mathbb{T}^{Education}$, $\mathbb{T}^{Sex}$ and $\mathbb{T}^{Salary}$ for $\cup QID$ attributes in Table 13, and the union cut $\cup Cut$ of the LKC anonymous Table 14.

$qid : |qid| \leq L$, qid is shard by at least K records, and the percentage of each sensitive value in every group cannot exceed a certain value C. LKC-privacy guarantees that the probability of a successful record linkage to be $\leq 1/K$ and the probability of a successful attribute linkage to be $\leq C$.

6.2.2 Encryption Scheme

The requirements of the encryption scheme is to: 1) support homomorphic addition, 2) allow for re-randomization of ciphertexts without the need for private information, 3) admit distributed key generation (DKG), and 4) allow for distributed decryption where participants use key shares to perform a decryption operation. We choose Exponential ElGamal [CGS97] as it is fast when implemented over elliptic curves, distributed key generation is straightforward, and decryption is feasible for our plaintext space. For the purpose of this chapter, we assume that the secret key x of Elgamal scheme is shared according to $(2, n)$-threshold between the participants, where any two participants can jointly decrypt Elgamal encrypted messages.

6.2.3 Taxonomy Tree [FWY07b]

A taxonomy tree of an attribute A, denoted by $\mathbb{T}^A$, is a context-specific hierarchical structure that classifies the items in the attribute's domain. In a taxonomy tree for a categorical attribute, the leaf nodes are the domain items, and the non-leaf nodes represent more generalized concepts of their children. On the other hand, in a taxonomy tree for a numerical attribute, the root node represents the full numerical range of the attribute, and the children nodes represent an optimal split of the parent range. We assume for each attribute $A \in \cup QID$, a context-specific taxonomy tree is defined. Figure 19 presents taxonomy trees for the $\cup QID$ attributes in Table 13.

6.3 Problem Formulation

In this section we formally define the research problem. First, we present an overview of the problem of privacy-preserving data integration with privacy guarantees on output data in Section 6.3.1. Next, we describe the utility measures in Section 6.3.2. We then describe the adversarial model in Section 6.3.3. Finally, we present the problem statement in Section 6.3.4.

6.3.1 Problem Overview

This chapter addresses the problem of integrating distributed person-specific data while preserving both privacy and information utility on the final mashup data. Each party involved in the protocol represents a data provider who is interested in integrating its data with other providers' data without leaking any unnecessary information. The mashup data is then released to the public for data mining.

We assume that the data being integrated is in the form of a relational table that is vertically partitioned into sub-tables, each of which is owned by one data provider. Let $\mathcal{P}_1$, $\mathcal{P}_1$, $\ldots$, $\mathcal{P}_p$ be the group of data providers participating in the protocol. Each party $\mathcal{P}_i : 1 \leq i \leq p$ owns a table in the form of $T_i = (UID, EID_i, QID_i, Sen_i, Class)$. UID is a system-generated unique identifier of an individual, and is shared by all data providers. EID_i is a set of explicit identifiers containing information that can explicitly identify an individual, and should be removed before the protocol is executed. QID_i is a set of quasi-identifier attributes, each of which is either categorical or numerical. QID_i attributes cannot be removed as they are useful for the data mining task, and each attribute can be shared by any number of data providers. We denote by $\cup QID = \bigcup_{i=1}^{p} QID_i$ the union of all quasi-identifier attributes owned by the parties. Sen_i is a set of sensitive attributes containing some sensitive information about the individuals, and it is shared between all data providers. $Class$ is a categorical target class attribute for classification analysis. We assume that only *one* data provider owns (have knowledge to) this attribute. The result of the integration is an anonymous mashup data that satisfies an LKC-privacy requirements agreed upon by all the parties. Note that increasing the anonymity threshold K, increasing the prior knowledge threshold L, or decreasing the confidence

Table 14: LKC anonymous mashup data, for $L = 2$, $K = 2$ and $C = 50\%$, w.r.t. sensitive value s_1

UID	**Age**	**Education**	**Class**	**Sex**	**Sen**	**Salary**
1	[1-99)	Secondary	Yes	Any_Sex	s_2	[10K-70K)
2	[1-99)	University	No	Any_Sex	s_1	[10K-70K)
3	[1-99)	Secondary	No	Any_Sex	s_2	[10K-70K)
4	[1-99)	University	Yes	Any_Sex	s_2	[10K-70K)
5	[1-99)	University	Yes	Any_Sex	s_2	[70K-125K)
6	[1-99)	University	No	Any_Sex	s_1	[70K-125K)
7	[1-99)	Secondary	No	Any_Sex	s_2	[10K-70K)
8	[1-99)	University	Yes	Any_Sex	s_2	[70K-125K)
9	[1-99)	Secondary	No	Any_Sex	s_2	[10K-70K)
10	[1-99)	University	Yes	Any_Sex	s_1	[70K-125K)

threshold C imposes a higher level of privacy protection, which in general would result in a lower information utility (data quality) of the anonymized data.

6.3.2 Data Utility Measures

Our idea for generating anonymous mashup data is to anonymize the raw data by performing a sequence of specializations, starting from the topmost general state. To specialize a value v, denoted by $v \rightarrow child(v)$, we replace v by its children values $child(v)$. The specialization process can be viewed as pushing the *cut* of each taxonomy tree downwards. A Cut^A of a taxonomy tree $\mathbb{T}^A$ contains exactly one value on each root-to-leaf path. We denote by $\cup Cut = \bigcup_{A \in \cup QID} Cut^A$ the union of all cuts. In Figure 19, the dashed curve represents $\cup Cut$ (also referred to as *solution cut*) of the LKC anonymous Table 14. The specialization starts from the topmost cut and pushes down the cut iteratively by specializing a value in the current cut until no further specialization that satisfies the LKC-privacy requirements is possible. We define two utility measures to help us determine at each level the best value v for specialization.

UM1: Classification Analysis

We utilize *information gain* [Qui93] to measure the quality of specialization on a value v for the purpose of classification analysis. Construction 6.1 illustrates how the score function $\mathsf{Score}(v)$ can be computed.

At any level during the specialization process, the score of every *valid attribute for specialization*

Classification Analysis

Let $T[v]$ be the set of records in T that are generalized to v. The score for a specialization on a value v can be determined as follows:

1. Compute the entropy of $T[v]$:

$$E(T[v]) = -\sum_{cls \in Class} \frac{|T[v \wedge cls]|}{|T[v]|} \times \log_2(\frac{|T[v \wedge cls]|}{|T[v]|}) \tag{5}$$

   where $|T[v \wedge cls]|$ denotes the records in $T[v]$ with class value cls.

2. Given that $|T[v]| = \Sigma_c |T[c]|$, where $c \in child(v)$, compute the entropy of $T[c]$ for each $c \in child(v)$:

$$E(T[c]) = -\sum_{cls \in Class} \frac{|T[c \wedge cls]|}{|T[c]|} \times \log_2(\frac{|T[c \wedge cls]|}{|T[c]|}) \tag{6}$$

3. Compute the score of specializing $T[v]$ on value v:

$$\mathsf{Score}(v) = E(T[v]) - \sum_{c \in child(v)} \frac{|T[c]|}{|T[v]|} E(T[c]) \tag{7}$$

Construction 6.1: Utility Measure for Classification Analysis

can be computed according to Construction 6.1, and then the value with the highest score is chosen to perform the actual specialization.

UM2: General Analysis

We utilize *discernibility cost* [SR92] to measure the quality of specialization on a value v when the data mining task is unknown. The discernibility cost penalizes each record that is indistinguishable from the rest of the records in a group, and the penalty cost equals the size of the group. That is, each record in an equivalence class qid is penalized by $|T[qid]|$, and the total penalty cost of the class is $|T[qid]|^2$. Hence, the score for specialization on v considers all qid combinations that contain v:

$$\mathsf{Score}(v) = \sum_{qid} |T[qid]|^2 \, : \, v \in qid \tag{8}$$

Similar to the utility measure for classification analysis, we choose the specialization that yields the highest score.

6.3.3 Adversarial Model

Fusion is secure in the semi-honest adversarial model [KMR], where each party is trusted to follow the protocol, but during the execution, tries to infer information from other parties. All parties are assumed to be non-colluding and their computational powers are polynomially bounded.

6.3.4 Problem Statement

Let $\mathcal{P}_1, \ldots, \mathcal{P}_p$, where $p > 2$, be a group of data providers respectively owning vertically-partitioned data tables $T_1, \ldots, T_p$, where any quasi-identifier attribute can be shared by any number of data providers, all sensitive attributes are shared between all data providers and the *Class attribute is only owned by one provider*. Let L, K and C be the prior knowledge threshold, the anonymity threshold and the confidence threshold values agreed upon by all parties. The objective of our work is to propose a protocol for generating an anonymous mashup data table $\hat{T}$ such that (1) $\hat{T}$ satisfies LKC-privacy requirements, (2) no party learns unnecessary information about other parties' data than what is in $\hat{T}$ (which is LKC-private), (3) $\hat{T}$ preserves an effective level of information utility for data mining purposes, and (4) the protocol is scalable with respect to high-dimensional data.

6.4 Solution: Fusion Protocol

6.4.1 Solution Overview

This section introduces a multi-party protocol, named Fusion, for integrating distributed person-specific data while preserving both privacy and information utility on the final mashup data. The main idea is to anonymize the raw data by generalizing all raw data records to a topmost general state, and then perform a sequence of specializations such that in each specialization step we choose the specialization with the highest score to maintain the highest possible information utility. The specialization process continues until there is no more specialization that satisfies the LKC-privacy requirements. Our solution consists of two main protocols:

Distributed Specialization Score (DSS)

Let $\mathcal{P}^{cls}$ be the party that owns the *Class* attribute. We assume that each party already received from $\mathcal{P}^{cls}$ the class values encrypted under the data providers's distributed public key, as shown in Table 15.

1. Following Construction 6.1, $\mathcal{P}^{cls}$ *directly* computes the score for each valid specialization of every attribute in $\cup QID$ it owns.
2. For each valid specialization $v \rightarrow child(v)$ of every attribute in $\cup QID$ owned by $\mathcal{P}_k : 1 \leq k \leq p$ (except $\mathcal{P}^{cls}$), $\mathcal{P}_k$ does the following:
   (a) Choose a party randomly and then together execute Sub-Protocol 1.3 to compute $[\![E(T[v])]\!]$.
   (b) For each $c \in child(v)$, it randomly chooses another party and then together execute Sub-Protocol 1.3 to compute $[\![E(T[c])]\!]$.
   (c) Homomorphically compute the score of specialization over v as follows:

$$\begin{aligned} [\![\mathsf{Score}(v)]\!] &= 10^d \times ([\![E(T[v])]\!] - \sum_{c \in child(v)} \frac{|T[c]|}{|T[v]|} \times [\![E(T[c])]\!]) \\ &= 10^d \times [\![E(T[v])]\!] - \sum_{c \in child(v)} \left\lceil \frac{10^d \times |T[c]|}{|T[v]|} \right\rceil \times [\![E(T[c])]\!] \end{aligned} \tag{9}$$

   where d is the number of decimal places (precision) agreed upon by all parties.
   (d) Request one of the parties to *partially* decrypt $[\![\mathsf{Score}(v)]\!]$, and then uses its own share of the secret key to fully decrypt the ciphertext and obtain Score(v).
3. All parties engage in a secure circuit evaluation process using Yao's Protocol [Yao82] to determine which party has the specialization value with the highest score.

Protocol 6.1: Distributed Specialization Score

Protocol 6.1 - Distributed Specialization Score (DSS). Since *Class* attribute is only owned by one party, this protocol enables all parties to *securely* determine at each specialization step the best value for specialization.

Protocol 6.2 - Hierarchal High-dimensional Data Integration (HHDI). This protocol presents a distributed hierarchical approach for integrating high dimensional data from multiple data providers, while preserving the data quality for the data mining tasks. The output of this protocol is an LKC-anonymous mashup data.

Input: Principle party $\mathcal{P}_i$, assisting party $\mathcal{P}_j$, potential specialization value x
Output: Encrypted entropy of $T(x)$

1. $\mathcal{P}_i$ chooses random integer r from $\mathbb{Z}_q^*$, and then for each ciphertext $[\![cls]\!] \in T[x].[\![Class]\!]$, where $T[v].[\![Class]\!]$ denotes the set of ciphertexts from the encrypted $[\![Class]\!]$ attribute that corresponds to the group of records generalized to v, it performs the following:
    (a) Blind $[\![cls]\!]$ by exponentiating in r: $[\![cls]\!]^r = [\![cls^r]\!]$.
    (b) Partially decrypt $[\![cls^r]\!]$ using its own share of the secret key.
2. $\mathcal{P}_i$ sends the set of *partially* decrypted ciphertexts to $\mathcal{P}_j$ through a secure channel.
3. Using its own share of the secret key, $\mathcal{P}_j$ decrypts the set of ciphertexts and obtains a set of blinded class values.
4. $\mathcal{P}_j$ computes the entropy $E(T[x])$ according to Equation 5 from Construction 6.1.
5. $\mathcal{P}_j$ computes the integer value $\lceil E(T[x]) \times 10^d \rceil$, encrypts it using the distributed public key, and then sends the ciphertext $[\![\lceil E(T[x]) \times 10^d \rceil]\!]$ to $\mathcal{P}_i$ through a secure channel.

Sub-Protocol 1. 3: Compute Encrypted Entropy

6.4.2 Multi-Party Protocol for Computing Specialization Score

In this section, we present a multi-party protocol for securely determining the best value for specialization. As we discussed in Section 6.3.2, Construction 6.1 can be used to compute the score of each valid specialization, and then the specialization that yields the highest score is selected. In distributed settings, however, different QID attributes are owned by different parties and the $Class$ attribute is owned by only one party. Therefore, a secure protocol is required to compute the scores and determine the best specialisation while ensuring no extra information is leaked to the parties.

Protocol 6.1 explains how the data providers can securely determine the winner candidate for specialization. As we will see in Section 6.4.3, the table T is constructed from the leaf partitions of the specialization tree. $\mathcal{P}^{cls}$, the party that owns the $Class$ attribute, can independently compute the score of its own valid specialization values. On the other hand, any other party that has a valid specialization value must utilize the other parties to achieve that. The intuition is to ask different parties to compute different parts of the each score, and then the party owning the specialization value puts things together by homomorphically computing the total score. Sub-Protocol 1.3 illustrates how $\mathcal{P}_i$ (the party owning the specialization value v) and $\mathcal{P}_j$ (assisting party) compute the entropy of $T[x]$. The idea is for $\mathcal{P}_i$ to blind the class ciphertexts corresponding to $T[x]$ using a random number, decrypt them using its secret decryption share, and then send them to $\mathcal{P}_j$ who in turn decrypts, counts the equivalent classes, and then computes the entropy. Using the same random number to blind each class ciphertext (Step 1 of Sub-Protocol 1.3) ensures the decrypted data is protected but the other party can still count and compute the entropy. We assume that at the beginning of the protocol, all parties agreed on a parameter d for converting decimal values to integers to be able to encrypt them. Observe that in Equation 9, we needed to multiply by 10^d to convert $\frac{|T[c]|}{|T[v]|}$ to an integer while maintaining the scale between all computed scores. $\mathcal{P}_k$ can then perform the multiplication: $\frac{|T[c]|}{|T[v]|} \times [\![E(T[c])]\!]$. The following example illustrates how to compute the score of a specialization according to Protocol 6.1.

Example 14 Assume that all records in Table 15 are generalized to *Any_Education*. Data provider $\mathcal{P}_1$, who owns the attribute *Education*, wants to securely compute the score for the specialization $Any\_Education \rightarrow \{Elementary, Secondary, University\}$ according to $\mathbb{T}^{Education}$ from Figure 19.

Step 1. To compute $E(T[Any\_Education])$, $\mathcal{P}_1$ blinds all ciphertexts in $T[Any\_Education].[\![Class]\!]$ with a random number r_1: $[\![Yes^{r_1}]\!]$, $[\![No^{r_1}]\!]$, $[\![No^{r_1}]\!]$, $[\![Yes^{r_1}]\!]$, $[\![Yes^{r_1}]\!]$, $[\![No^{r_1}]\!]$, $[\![No^{r_1}]\!]$, $[\![Yes^{r_1}]\!]$, $[\![No^{r_1}]\!]$, $[\![Yes^{r_1}]\!]$, *partially* decrypts them using its private key share, and the send them to $\mathcal{P}_3$ (randomly selected). $\mathcal{P}_3$ then decrypts the ciphertexts using its private key share to obtain the blinded values: Yes^{r_1}, No^{r_1}, No^{r_1}, Yes^{r_1}, Yes^{r_1}, No^{r_1}, No^{r_1}, Yes^{r_1}, No^{r_1}, Yes^{r_1}, and computes $-\frac{5}{10} \times \log_2(\frac{5}{10}) - \frac{5}{10} \times \log_2(\frac{5}{10}) = 1$. It then computes the integer $\lfloor 1 \times 10^d \rfloor = \lfloor 1 \times 10^2 \rfloor = 100$, and then sends the ciphertext $[\![100]\!]$ to $\mathcal{P}_1$.

Step 2. Since the minimum education value in any records $T[Any\_Education]$ is $7th$ grade, then no record can be specialized to *Elementary*. As a result, $|T[Elementary]| = 0$ and $E(T[Elementary]) = 0$. On the other hand, $T[Secondary] = \{UID\#1, 3, 7, 9\}$ (4 records can be specialized to *Secondary*). Therefore, $\mathcal{P}_1$ blinds all ciphertexts in $T[Secondary].[\![Class]\!]$ with a random number r_2: $[\![Yes^{r_2}]\!]$,

Table 15: Data owned by $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ after $\mathcal{P}_2$ sends encrypted $[\![Class]\!]$ attribute to $\mathcal{P}_1$ and $\mathcal{P}_3$

UID	$\mathcal{P}_1$			$\mathcal{P}_2$		$\mathcal{P}_3$		
	Age	**Education**	$[\![Class]\!]$	**Class**	**Sex**	**Sen**	**Salary**	$[\![Class]\!]$
1	54	11th	$[\![Yes]\!]$	Yes	Male	s_2	65K	$[\![Yes]\!]$
2	26	Bachelor	$[\![No]\!]$	No	Male	s_1	37K	$[\![No]\!]$
3	39	7th	$[\![No]\!]$	No	Female	s_2	51K	$[\![No]\!]$
4	67	Master	$[\![Yes]\!]$	Yes	Female	s_2	55K	$[\![Yes]\!]$
5	32	Bachelor	$[\![Yes]\!]$	Yes	Male	s_2	87K	$[\![Yes]\!]$
6	59	Doctorate	$[\![No]\!]$	No	Female	s_1	107K	$[\![No]\!]$
7	44	12th	$[\![No]\!]$	No	Female	s_2	26K	$[\![No]\!]$
8	29	Bachelor	$[\![Yes]\!]$	Yes	Male	s_2	77K	$[\![Yes]\!]$
9	53	9th	$[\![No]\!]$	No	Female	s_2	29K	$[\![No]\!]$
10	46	Master	$[\![Yes]\!]$	Yes	Female	s_1	72K	$[\![Yes]\!]$

$[\![No^{r_2}]\!]$, $[\![No^{r_2}]\!]$, $[\![No^{r_2}]\!]$, *partially* decrypts them, and then sends them to $\mathcal{P}_2$. $\mathcal{P}_2$ decrypts the ciphertexts and computes $-\frac{1}{4} \times \log_2(\frac{1}{4}) - \frac{3}{4} \times \log_2(\frac{3}{4}) = 0.8112$. It then computes $\lceil 0.8112 \times 10^2 \rceil = 81$ and sends $[\![81]\!]$ to $\mathcal{P}_1$. Similarly, $T[University] = \{UID\#2, 4, 5, 6, 8, 10\}$ (6 records can be specialized to *University*). Therefore, $\mathcal{P}_1$ blinds all ciphertexts in $T[University].[\![Class]\!]$ with a random number r_3: $[\![No^{r_3}]\!]$, $[\![Yes^{r_3}]\!]$, $[\![Yes^{r_3}]\!]$, $[\![No^{r_3}]\!]$, $[\![Yes^{r_3}]\!]$, $[\![Yes^{r_3}]\!]$, *partially* decrypts them, and then sends them to $\mathcal{P}_3$. $\mathcal{P}_3$ decrypts the ciphertexts and computes $-\frac{4}{6} \times \log_2(\frac{4}{6}) - \frac{2}{6} \times \log_2(\frac{2}{6}) = 0.9183$. It then computes $\lceil 0.9183 \times 10^2 \rceil = 92$ and sends $[\![92]\!]$ to $\mathcal{P}_1$.

Step 3. Since $|T[Any\_Education]| = 10$, $|T[Secondary]| = 4$ and $|T[University]| = 6$, $\mathcal{P}_1$ homomorphically computes the score for specializing on value *Any_Education* as follows:

$[\![\mathsf{Score}(Any\_Education)]\!] = 10^2 \times [\![100]\!] - \lceil \frac{10^2 \times 4}{10} \rceil \times [\![81]\!] - \lceil \frac{10^2 \times 6}{10} \rceil \times [\![92]\!] = [\![1240]\!]$. $\square$

Proposition 10 ***Privacy.*** *Protocol 6.1 is privacy-preserving.*

Proof. To prove that Protocol 6.1 is privacy-preserving, we show that the data is protected throughout the protocol execution.

Encrypted Data. While encrypted, all ciphertexts exchanged between the parties (encrypted class attributes, entropies and scores) are protected under the CPA-security (Decisional Diffie-Hellmen (DDH)) of ElGamal [EG85] encryption scheme. The adversary cannot decrypt items arbitrarily, as the decryption key is $(2, n)$-shared between all data owners, requiring a collusion with another party, which contradicts our non-colluding semi-honest adversarial model.

Decrypted Data. In Step 2 of Sub-Protocol 1.3, $\mathcal{P}_j$ receives a set of class attribute ciphertexts

Hierarchal High-dimensional Data Integration (HHDI)

1. $\mathcal{P}^{cls}$, the party that owns the *Class* attribute, encrypts the class values under the data providers' distributed public key, and then broadcasts the ciphertexts to the remaining parties.
2. Create an initial partition such that:
   - The hierarchy cut value $HCut.A$ of every attribute $A \in \cup QID$ is set to the root of $\mathbb{T}^A$.
   - All records are assigned to the partition.
3. Set the union cut $\cup Cut$ to the hierarchy cut $HCut$ of the initial partition.
4. For each value $v \in \cup Cut$ from a taxonomy tree $\mathbb{T}^A$, $\mathcal{P}^A$ determines whether v is *valid* for specialization.
5. While there is at least one value $v \in \cup Cut$ such that v is valid for specialization:
   (a) All parties jointly run Protocol 6.1 to compute the specialization score for each valid value in $\cup Cut$, and determine the party that owns the winning value w with the highest score.
   (b) The party owning w runs Sub-Protocol 2.3 to perform the specialization $w \rightarrow child(w)$.
   (c) For each value $v \in \cup Cut$ from a taxonomy tree $\mathbb{T}^A$, $\mathcal{P}^A$ verifies whether v is *valid* for specialization.
6. The hierarchy cut $HCut$ and the record count of each leaf partition constitute the anonymous data for release satisfying the LKC-private requirements.

Protocol 6.2: Hierarchal High-dimensional Data Integration

from $\mathcal{P}_i$ in order to decrypt and compute the entropy. Decrypting the ciphertexts enables $\mathcal{P}_j$ to count the equivalent class values. However, since the decrypted data is blinded (exponentiated with a random number), $\mathcal{P}_j$ cannot determine the actual class values due to the the hardness of computing discrete logarithms. Moreover, using different random numbers for blinding different set of class values prevents $\mathcal{P}_j$ from comparing two different set of blinded class values it has received from two separate requests. Our protocol, however, leaks partial information about a score to each assisting party, since entropies are computed by assisting parties in clear text. We argue that this leakage is tolerable since assisting party $\mathcal{P}_j$ can determine neither to which attribute the computed entropy belongs, nor what the underlying class values are. ■

Perform Specialization

1. For each partition Part where $w \in$ Part.$HCut$:
   (a) For each child value $v \in child(w)$, create a child partition CPart such that CPart.$HCut$ is the same as Part.$HCut$ except that in the former, w is replaced by v.
2. Assign the records in Part to the child partitions according to Definition 18.
3. Update $\cup Cut$ by replacing w by its children $child(w)$.

Sub-Protocol 2. 3: Perform Specialization

6.4.3 Multi-Party Protocol for LKC-private Data Integration Release

Given the distributed data tables $T_1, \ldots, T_p$, the taxonomy trees for $\cup QID$, thresholds L, K and C, the goal is to generate an integrated and anonymous data for data mining while satisfying LKC-privacy. To ensure that no party learns unnecessary information about other parties' data during the integration process, we propose a hierarchal approach for specializing the data called *Hierarchal High-dimensional Data Integration (HHDI)*.

The general idea of our solution is to initially generalize and assign *all* records to a *partition*, and then apply a top-down specialization process guided by the taxonomy trees to specialize the records and assign then to disjoint child partitions until no further partitions can be created without violating LKC-privacy. A *partition* is a data structure that consists of two components: $HCut$ and *Recs*. Hierarchy cut $HCut$ is an ordered set of values $\langle v_1, \ldots, v_{|\cup QID|}\rangle$, where each value is from a taxonomy tree $\mathbb{T}^A$ of an attribute $A \in \cup QID$. *Recs* contains the unique identifiers UIDs of the records assigned to the partition.

Definition 18 Record Generalization. A record R can be assigned to a partition Part if for each attribute $A \in \cup QID$, R.A can be generalized to Part.$HCut.A$, where R.A and Part.$HCut.A$ are the values in R and Part.$HCut$ that correspond to attribute A, respectively. □

Jointly executed by all data providers, Protocol 6.2 illustrates how the specialization process is performed in order to generate the LKC-anonymous table. The parties coordinate their actions

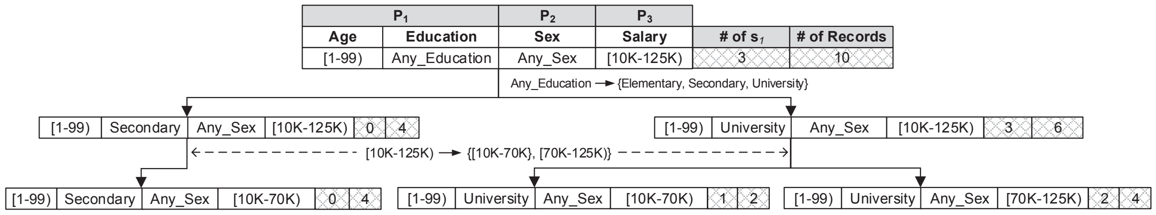

P1
P2
P3
Age
Education
Sex
Salary
# of s1
# of Records
[1-99)
Any_Education
Any_Sex
[10K-125K)
3
10
Any_Education → {Elementary, Secondary, University}
[1-99) Secondary Any_Sex [10K-125K) 0 4
[1-99) University Any_Sex [10K-125K) 3 6
[10K-125K) → {[10K-70K}, [70K-125K)}
[1-99) Secondary Any_Sex [10K-70K) 0 4
[1-99) University Any_Sex [10K-70K) 1 2
[1-99) University Any_Sex [70K-125K) 2 4

Figure 20: Hierarchal high-dimensional data integration (HHDI) on the data in Table 13.

using a private broadcast channel called a bulletin board. Initially, all records are assigned to the initial partition. This assignment satisfies Definition 18 since The $HCut$ of the initial partition contains the most general values (roots) of the taxonomy trees.

Example 15 Figure 20 illustrates the specialization process on Table 13 in order to generate an LKC-anonymous table that satisfies $L = 2$, $K = 2$ and $C = 50\%$. The root partition represents the *initial partition* such that $HCut = \langle [1-99), Any\_Education, Any\_Sex, [10K-125K) \rangle$ and $Recs = \{UID\#1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. The union cut is also set to the most general values $\cup Cut = \langle [1-99), Any\_Education, Any\_Sex, [10K-125K) \rangle$. □

To determine which valid value to specialize on, all parties jointly run Protocol 6.1. In general, a specialization $v \rightarrow child(v)$ involves generating a child partition for each child value in $child(v)$. The cut of a taxonomy tree to which v belongs is pushed downwards, and v is replaced in the hierarchy cuts of the newly generated partitions by its children values $child(v)$. The party that owns the winner value preforms the actual specialization according to Sub-Protocol 2.3.

Example 16 In Figure 20, the winner value for the first specialization is $Any\_Education$. Therefore, part $\mathcal{P}_1$, which owns attribute $Education$, creates two partitions Part_1 and Part_2, $\mathsf{Part}_1.HCut = \langle [1-99), Secondary, Any\_Sex, [10K-125K) \rangle$ and $\mathsf{Part}_1.Recs = \{UID\#1, 3, 7, 9\}$, and $\mathsf{Part}_2.HCut = \langle [1-99), University, Any\_Sex, [10K-125K) \rangle$ and $\mathsf{Part}_2.Recs = \{UID\#2, 4, 5, 6, 8, 10\}$. $\mathcal{P}_1$ updates the union cut: $\cup Cut = \langle [1-99)$, $Secondary$, $University$, Any_Sex, $[10K-125K) \rangle$. □

A specialization is *valid* if after the child partitions are created, the leaf partitions *as a whole* in the partitioning tree still satisfies LKC-privacy. The specialization process terminates when no more valid specialization is available. The mashup data for the final release can be constructed from

Table 16: *Adult* dataset, and the distribution of attributes among parties in each multiparty setting

Attribute	**Type**	**Distributed Settings**		
		3 parties	**4 parties**	**5 parties**
Age	Numerical	$\mathcal{P}_2$	$\mathcal{P}_2$	$\mathcal{P}_2$
Work-class	Categorical	$\mathcal{P}_2$	$\mathcal{P}_3$	$\mathcal{P}_1$
Final-weight	Numerical	$\mathcal{P}_1$	$\mathcal{P}_1$	$\mathcal{P}_3$
Education	Categorical	$\mathcal{P}_2$	$\mathcal{P}_4$	$\mathcal{P}_3$
Education-num	Numerical	$\mathcal{P}_1$	$\mathcal{P}_4$	$\mathcal{P}_4$
Marital-status	Categorical	$\mathcal{P}_3$	$\mathcal{P}_3$	$\mathcal{P}_3$
Occupation	Categorical	$\mathcal{P}_3$	$\mathcal{P}_1$	$\mathcal{P}_4$
Relationship	Categorical	$\mathcal{P}_2$	$\mathcal{P}_1$	$\mathcal{P}_5$
Race	Categorical	$\mathcal{P}_2$	$\mathcal{P}_2$	$\mathcal{P}_4$
Sex	Categorical	$\mathcal{P}_1$	$\mathcal{P}_3$	$\mathcal{P}_2$
Capital-gain	Numerical	$\mathcal{P}_1$	$\mathcal{P}_1$	$\mathcal{P}_1$
Capital-loss	Numerical	$\mathcal{P}_1$	$\mathcal{P}_2$	$\mathcal{P}_5$
Hours-per-week	Numerical	$\mathcal{P}_3$	$\mathcal{P}_2$	$\mathcal{P}_1$
Native-country	Categorical	$\mathcal{P}_3$	$\mathcal{P}_4$	$\mathcal{P}_2$

the hierarchy cut of the leaf partitions, where each hierarchy cut is duplicated $|Recs|$ times (the number of records assigned to the partition).

Example 17 The output of the specialization process in Figure 20 is: $\langle[1-99)$, *Secondary*, *Any_Sex*, $[10K-705K)\rangle$ $\times 4$, $\langle[1-99)$, *University*, *Any_Sex*, $[10K-70K)\rangle$ $\times 2$, and $\langle[1-99)$, *University*, *Any_Sex*, $[70K-125K)\rangle$ $\times 4$, which is equivalent to the records presented in the *LKC*-anonymous Table 14. □

Based on *LKC*'s *anti-monotonic property* [FWCY10], once a specialization on a value becomes *invalid*, further specializations on $child(v)$ will always be invalid. This property significantly reduces the partitioning space, while guaranteeing that the output is suboptimal.

6.5 Performance Evaluation

In this section we evaluate the performance of Fusion. First, we discuss the implementation details, and then we present the experimental results that include mashup utility, attribute scalability and record scalability.

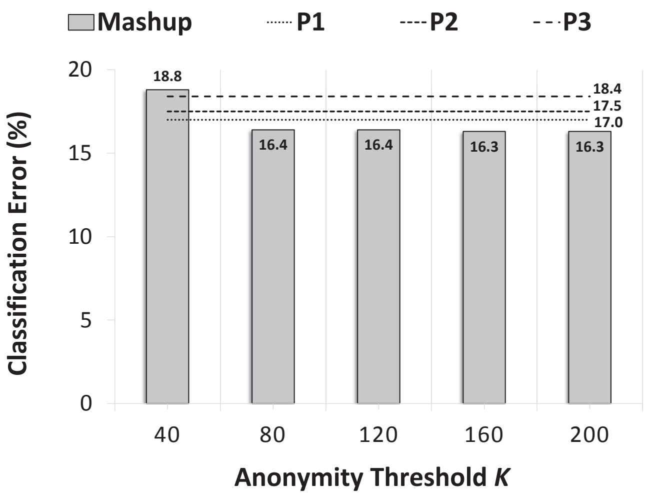


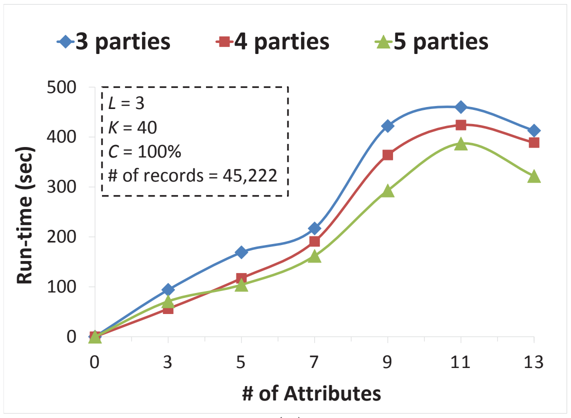
Figure 21: Utility of mashup w.r.t. anonymity threshold K.

6.5.1 Implementation and Setup

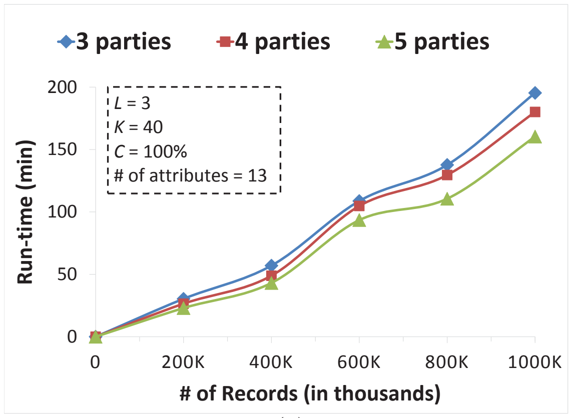
Fusion is implemented using SCAPI[1], an open-source Java library for implementing secure multiparty computation protocols. We utilize queue-based channels in the communication layer to allow for asynchronous transfer of ciphertexts between parties, where ActiveMQ[2] is used as the messaging broker. The experiments were conducted on a machine equipped with an Intel Core i7 3.8GHz CPU and 16GB RAM, running 64-bit Windows 7.

We utilize a real-life *adult* data set [BL13] in our experiments to illustrate the performance of Fusion. The adult data set consists of 45,222 census records containing six numerical attributes, eight categorical attributes, and a *class* attribute. Table 16 lists all the attributes and their types. In our experiments, we model three different distributed settings: 3 parties, 4 parties and 5 parties. We consider attribute *Marital-status* as the sensitive attribute, while we consider the remaining attributes as quasi-identifiers. Table 16 illustrates the distribution of each attribute between parties in each of the distributed settings.

[1] SCAPI: https://scapi.readthedocs.org/
[2] ActiveMQ: https://activemq.apache.org/



(a)



(b)

Figure 22: Scalability with respect to (a) the number of attributes and (b) the number of records.

6.5.2 Mashup Utility

Rather than releasing an anonymous mashup data for classification analysis, each data provider could release a *classifier* of its data. To determine the usefulness of our approach with respect to classification analysis, we utilize C4.5 classifier [Sal94] to compare the classification error of the mashup data with the classification error of the classifier of each party. We use 30,160 records (2/3) to build (train) the classifiers, and 15,062 records (1/3) for testing.

Figure 21 depicts the classification error for each individual party, as well as for the mashup data. The classification error is measured w.r.t. the anonymity threshold K, where K linearly increases from 40 and 200. Our approach is robust w.r.t. L, since we found out that increasing the prior knowledge of the adversary does not impact the data quality. The classification error for $\mathcal{P}_1$ is 17%, $\mathcal{P}_2$ is 17.5% and $\mathcal{P}_3$ is 18.4%. On the other hand, the mashup classification error decreases from 18.8% to 16.3% as K increases from 40 to 200. Except when $K = 40$, we observe that all data providers benefits from participating in integration process, where the maximum benefit is as much as 2.1% and the minimum benefit is as low as 0.6%.

6.5.3 Scalability

We measure the scalability of Fusion with respect to the number of attributes (*Attribute Scalability*) and the number of records (*Record Scalability*) in three distributed settings: 3 parties, 4 parties and

5 parties.

Attribute Scalability

Figure 22a depicts the runtime from 3 to 13 attributes, for $L = 3$, $K = 40$, $C = 100\%$ and 45,222 records. We observe that the runtime grows sub-linearly when the number of attributes linearly increases, regardless of the number of parties in the setting. We also observe that runtime decreases as the number of parties increases. This is because adding more parties reduces the load on each individual party.

Record Scalability

Figure 22b depicts the runtime from 200,000 to 1,000,000, for $L = 3$, $K = 40$, $C = 100\%$ and 13 attributes. We observe that it takes up to 195 minutes to run Fusion on a dataset with 1,000,000 records in a 3-party setting. This is mainly due to the fact that we perform a modular exponentiation operation every time a ciphertext is blinded in Sub-Protocol 1.3. However, we also observe that the runtime is still scalable w.r.t. the linear increase in the number of records, regardless of the number of parties in the setting. Similar to Section 6.5.3, we observe that runtime decreases as the number of parties increases.

6.6 Summary

In this chapter, we present a secure protocol for data integration in a distributed setting. The protocol is privacy-preserving, while the output is a mashup data for data mining that satisfy LKC-privacy. We empirically show that the mashup data contains higher information utility, and that the protocol is scalable *w.r.t.* the number of records as well as the number of attributes in the mashup data.

Chapter 7

Secure and Privacy-preserving Data Auditing in Bitcoin

7.1 Introduction

Digital currencies enable transactions that are electronically authorized, cleared and settled. After decades of research [Cha82, CHL05, Bel11, Par11] and failed business ventures attempting to establish a digital currency, Bitcoin [Nak08] was proposed and deployed in 2009. While still in its infancy, Bitcoin has achieved unprecedented success, enjoying a multi-billion dollar market capitalization and deployment by large retailers. Bitcoin transactions can be executed at any time by any device in the world with low (sometimes zero) fees.

Users can maintain security of their assets by managing the private keys used to control them. However, managing cryptographic keys is difficult for many users [EBSC15]. Equipment failure, lost or stolen devices, or Bitcoin-specific malware [LS14] could all result in the loss of one's holdings. Many users prefer to keep their holdings with online *exchanges* for a simple user experience similar to online banking—*e.g.,* with passwords, account recovery, velocity limits and customer support. Exchanges, as their name suggest, also provide conversion services between bitcoin[1] and other currencies. Customers can 'withdraw' by instructing the exchange to send the stored bitcoin to a Bitcoin address for which they manage the private key.

Unfortunately, storing assets with an exchange leaves users vulnerable to the exchange being hacked and losing its assets. One of the most notorious events in Bitcoin's short but storied history is the collapse and ongoing bankruptcy of the oldest and largest exchange, Mt. Gox, which lost over US$450M in customer assets. A number of other exchanges have lost their customers' Bitcoin holdings and declared bankruptcy due to external theft, internal theft, or technical mistakes [MC13].

While the vulnerability of an exchange to catastrophic loss can never be fully mitigated, a sensible safeguard is periodic demonstrations that an exchange controls enough bitcoins to settle all of its customers' accounts. Otherwise, an exchange which has (secretly) suffered losses can continue operating until the net withdrawal of Bitcoin exceeds their holdings. Note that while conventional banks typically implement *fractional reserve banking* in which they only retain enough assets to cover a fraction of their liabilities, the Bitcoin community is skeptical of this approach and exchanges are generally expected to be fully solvent at all times.

A rudimentary approach to demonstrating assets is simply to transfer them to a fresh public key. Mt. Gox did so once in 2011 in the face of customer skepticism, moving over ₿420k (then worth over US$7 M) in a single large transaction. However, this demonstration undermined Mt. Gox's privacy by revealing which Bitcoin addresses they controlled. It was never repeated.

More importantly, a *proof of reserves* without a corresponding *proof of liabilities* is not sufficient to prove solvency. A proof of liabilities might consist of an audit by a trusted accountant, as done for example by Coinbase[2] and Bitstamp[3]. This might be improved by allowing users to independently verify they are in the dataset seen by the auditor, a step taken by Kraken[4] and OKCoin[5].

[1]Following convention, we refer to the protocol as 'Bitcoin' and the units of currency as 'bitcoin' or ₿.

[2]A. Antonopoulos, "Coinbase Review," *antonopoulos.com* (Blog), 25 Feb 2014.

[3]E. Spaven, "Bitstamp Passes Audit Overseen by Bitcoin Developer Mike Hearn," *CoinDesk*, 27 May 2014.

[4]N. Hajdarbegovic. "Kraken Bitcoin Exchange Passes 'Proof of Reserves' Cryptographic Audit," *CoinDesk*, 24 Mar 2014.

[5]J. Southurst, "OKCoin Reveals BTC Reserves of 104% as China's Exchanges Undergo Audits," *CoinDesk*, 22

The notion of a cryptographic proof of liabilities, verifiable by any party with no trusted auditor, was first proposed by Maxwell [Wil14], although this initial proposal leaks information about the number and size of customer accounts (see Section 7.2.2). These privacy issues (as well as those inherent to a simple public proof of assets) have been cited by some exchanges (*e.g.,* Kraken[6]) as a reason to use a trusted auditor instead.

In this chapter we propose Provisions, a cryptographic proof of solvency scheme with the following properties:

- no information is revealed about customer holdings
- the value of the exchange's total total holdings is kept secret
- the exchange maintains unlinkability from its Bitcoin address(es) through an anonymity set of arbitrary size
- multiple exchanges performing Provisions contemporaneously can prove they are not colluding

While the Maxwell proof of reserves is a straightforward use of a Merkle tree, a data structure well known by Bitcoin community, Provisions employs somewhat heavier cryptography not found in Bitcoin itself—*e.g.,* homomorphic commitments and zero knowledge proofs. However, we demonstrate that Provisions is efficient enough in practice even for the largest of today's exchanges to conduct a daily proof of solvency, being computable by a single server in a few hours and requiring proofs which are less than 20 GB in size. Given this practicality and the strong privacy guarantees, we hope it will become the norm for exchanges to regularly compute a Provisions proof of solvency which might go a long way to restoring confidence in the Bitcoin ecosystem.

Limitations It is important to recognize that no proof of solvency (or any other type of audit) is future proof, as exchanges can still be hacked at any time. Likewise, proving control of a quantity of bitcoin does not guarantee the exchange itself will behave honestly in the future. It may simply abscond with all of its customers funds after completing a Provisions proof. The best we can hope

Aug 2014.

[6]"Kraken Proof-of-Reserves Audit Process," https://www.kraken.com/security/audit

for is efficient enough proofs to enable frequent and continual monitoring of the financial health of exchanges to quickly detect the loss of funds, which Provisions enables.

Provisions also requires customers to check individually that their balance has been included in the proof of liabilities. This appears to be a fundamental limitation given our privacy goals that a user's account balance is not revealed to any other party. On the positive side, as long as *some* users check and the exchange cannot predict confidently which users will check, it runs a high risk of detection if it cheats (see Section 7.5.3).

Provisions is also limited to proving ownership of accounts with a full public key on the blockchain (not unused pay-to-pub-key-hash or pay-to-script-hash addresses which haven't yet be been used or multi-sig addresses). Removing this limitation is an interesting challenge for future work.

The work in this chapter is a collaborative effort with a team from the computer science department at Stanford University. The results of this chapter have been published in [DBB+15b].

7.2 Background

We assume the reader is familiar with Bitcoin [Nak08]. Bonneau et al. [BMC+15] provide an extensive survey of Bitcoin, although a deep understanding is not needed for understanding Provisions. The pertinent features are that each unit of bitcoin is usually redeemable by a specified public key[7] and this information is maintained in a public data structure called the blockchain.

Note that the blockchain is an ever-growing log of transactions. Any proof of solvency will be inherent to a single block, representing one snapshot of the state of the system. In the remainder of the chapter we leave implicit the proof will be valid for a specific block number t. It is also possible for the blockchain to fork (or "re-org") in which case an apparently-valid proof at block t may not be valid in the final block number t. As is standard with Bitcoin transactions, the defense against this is to wait until a block is *confirmed* with high probability, typically after observing that 6 followup

[7]Technically, bitcoins are redeemable by a specific transaction script which can encode various spending conditions, though in the vast majority of cases this is simply a public key signature and we will discuss Bitcoin as if this is the only method.

blocks have been published.

Bitcoin public keys which hold funds are interchangeably called *accounts* or *addresses*. We note here that while we designed Provisions with Bitcoin in mind as it is the dominant cryptocurrency today, it could easily be ported to similar cryptocurrencies which have the above properties.

A proof of solvency consists of two components. In the first, the *proof of liabilities*, the exchange proves the total value of bitcoin it owes to each of its users. In the second, the *proof of assets*, the exchange proves the total value of bitcoin it has signing authority over. If the latter amount is greater than or equal to the former, the exchange is considered solvent.

7.2.1 Exchange Structure and Holdings

Nearly all large Bitcoin exchanges operate by pooling customers' funds into a small number of large accounts. Typically for security reasons the keys for some of these accounts are kept on offline computers or in hardware security modules, requiring human action to authorize transactions (commonly called *cold storage*).

One might ask why an exchange does not simply maintain a separate Bitcoin address for each customer, enabling *direct monitoring* by each user of their funds on the public blockchain; a simple mechanism that eschews the need for a more complicated cryptographic proof of solvency. By itself, this scheme is not secure, as a malicious exchange might attempt to convince two users with the same balance that a single address is holding funds for both of them (a variation of the *clash attack* [VTK12] discussed later).

This model also has several key practical shortcomings. First, it prevents simple division of money into hot and cold storage. Current exchanges can exist with a limited amount of money in more vulnerable hot storage because, on aggregate, the number of withdrawals in a given day is typically only a small amount of total holdings. This is similar to a large offline bank which does not carry enough cash in ATMs to cover all customer accounts, keeping substantial assets in secure (but less accessible) storage.[8]

[8]Executing Provisions will require computation using all of an exchange's private keys, including those for assets in

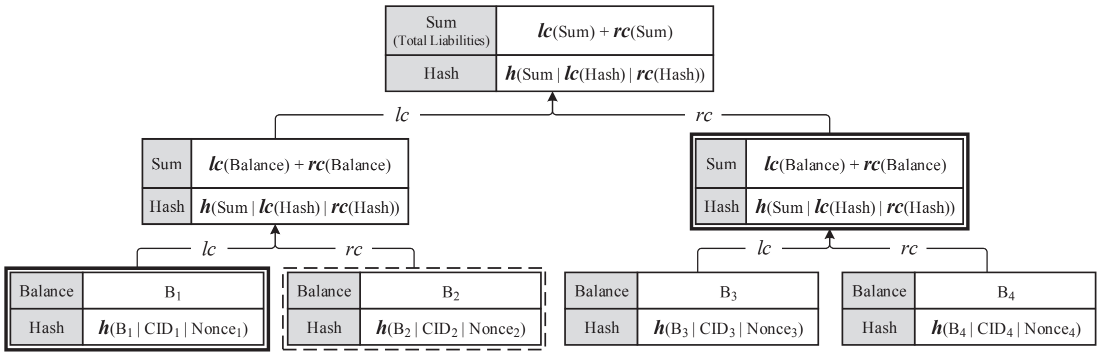


Figure 23: The Merkle tree from the Maxwell protocol [Wil14] for proof of solvency. When a customer desires to verify their account (e.g. dashed line node), only two nodes need to be sent to the customer (bold line nodes).

Second, pooling assets means that transfers between customers can be efficiently settled by changing each customers' account balance without executing a transaction on the Bitcoin blockchain (incurring a transaction fee and a wait of around an hour for confirmation). Similarly, two exchanges can aggregate multiple transactions between pairs of their customers into a single settlement payment (referred to as *netting*). Minimizing reliance on the blockchain (especially for small transfers) is a key benefit of exchanges. By contrast, maintaining a separate Bitcoin account for each customer requires "hitting the blockchain" with every transaction.

Finally, although it is not typically advertised, exchanges offer a significant privacy benefit to users as as pooling funds ensures that it is not easy for outside observers to link deposits and withdrawals to the same individual [MPJ+13].

Thus, we consider the pooled assets model likely to persist and we have designed Provisions to work in this model. If we combine these factors with maintaining the privacy of an exchange's addresses—proving that one owns (*i.e.,* knows) a private key without disclosing which—zero knowledge proofs appear inescapable.

cold storage. However, this can be done with human intervention at a predictable time and does not require network access to the cold storage.

7.2.2 Maxwell's Proof of Liabilities

Maxwell proposed a protocol (summarized by Wilcox [Wil14]) that enables an exchange to prove its total liabilities while allowing users to verify that their accounts are included in this total. The exchange constructs a binary Merkle hash tree [Mer79] where each leaf node contains a customer's balance, as well as the hash of the balance concatenated with the customer id and a fresh nonce (*i.e.,* a hash-based commitment). Each internal node stores the aggregate balance of its left child (lc) and right child (rc), as well as the hash of its aggregate balance concatenated with the hash of its left and right children. The root node stores the aggregate of all customers' balances, representing the total liabilities, and the exchange broadcasts the root node. This is illustrated in Figure 23.

When a customer wants to verify that their balance is included in the total liabilities declared by the exchange, it is sufficient to send to the customer only part of the hash tree in order to perform the verification. Specifically, the exchange sends to the customer her nonce and the sibling node of each node on the unique path from the customer's leaf node to the root node. The other nodes on the path, including the leaf node itself, do not need to be sent to the customer because they will have sufficient information to reconstruct them. The customer eventually accepts that their balance is included iff their path terminates with the same root broadcast by the exchange.

While elegant, this protocol does not hide the value of the exchange's total liabilities which is published in the root node. While a rough sense of this value may be public knowledge, the exact value may be sensitive commercial data. Furthermore, regular proofs will reveal precise changes in the exchange's holdings.

This protocol also leaks partial information about other customers' balances. For example, if a simple balanced tree is used then each customer's proof reveals the exact balance of the sibling account in the tree (although the account holder remains anonymous). More generally, each sibling node revealed in a given users' path to the root node reveals the total holdings of each customer in that neighboring subtree. This could be mitigated somewhat by using an unbalanced tree so it is not immediately clear how many customers are in any neighboring subtree, but the protocol inherently leaks some information. Provisions removes this problem entirely, revealing no information about any users' assets beyond the fact that the total is less than the exchange's proven reserves.

7.2.3 Proof of Assets

Once an exchange establishes its total liabilities, it must prove it owns sufficient bitcoin to match (or exceed) its liabilities. This proof of assets together with the proof of liabilities forms a proof of solvency. Maxwell's proof of assets does not preserve privacy. Instead, the exchange publicly demonstrates control of a set of addresses holding at least as much bitcoin as the exchange's total liabilities. This demonstration of control might involve moving a challenge amount of bitcoin from each account or signing a challenge message with the private key associated with each address. Exchanges may be reluctant to do so for privacy and security concerns (revealing their internal division of funds between accounts).

In Provisions, we enable the exchange to prove ownership of an anonymous subset of addresses pulled from the blockchain. The total quantity of bitcoin across these addresses can then be determined, without being revealed, and proved to be equal or greater than the exchange's total liabilities.

Control vs. Ownership

Any proof of assets, including Provisions, faces the inherent problem that the ability to use the signing key of an address does not necessarily imply ownership of it. A malicious exchange may collude with one or more bitcoin holders who agree to use their accounts to cover the exchange's liabilities. However, these partners may have no intention of ever making their holdings available to the exchange's customers.

An exchange might try consolidating its holdings into a single address to demonstrate that either exchange or the colluder is risking their bitcoin by placing it under the other's control. However, there is no guarantee that the single address does not implement a shared access structure by a threshold signature scheme [Gol14].

This problem is fundamental, as no system can cryptographically prove its intentions to return something of value to a given user if requested. This customer request will be made without cryptographic authentication (*e.g.,* password-authenticated) because by assumption exchange customers are unwilling or unable to manage cryptographic keys. Otherwise, assets could be proved by sending each customer's bitcoins to a 1-out-of-2 multisig address redeemable by either the exchange or the user [Wil14], providing a window for each customer to redeem their coins if desired. Again, we assume this is impractical for most exchange customers.

Collusion Attacks

Another potential vulnerability is that a cabal of two or more malicious exchanges might collude by using their own assets to participate in each other's proof of assets, making each exchange appear to control the total amount controlled by the cabal. With a public proof of assets, this would be detected if done simultaneously (because the same addresses would appear in multiple exchanges' proofs) while the transaction graph might reveal if assets are simply being moved around in a shell game.

In Provisions, because the exchange's addresses are kept confidential, detection of this attack becomes more challenging. However, in Section 7.4.6 we show an extension to the basic Provisions protocol which enables exchanges to prove that they are not using the same assets as other exchanges running the protocol. To do so, they publish an additional value which is unlinkable to their real Bitcoin address, yet is a deterministic function of (and requires knowledge of) their private key. Thus, if any two exchanges attempt to use the same bitcoin address in separate executions of Provisions, they can be detected.

This extension imposes a small performance cost (see Section 7.6.4) and a small impact on the exchange's privacy as it reveals the number of addresses to which the exchange knows the private key (see Section 7.5.2). Thus we leave it as an extension for now, as it will only become beneficial when multiple exchanges are implementing Provisions and are willing to synchronize their proofs.

7.3 Preliminaries

7.3.1 Public Parameters

We let g and h be fixed public generators of a group G of prime order q. Our implementation uses the elliptic curve secp256k1 [Cer00] as the group G; this is the group used for Bitcoin ECDSA signatures. Note that this allows us to work with existing Bitcoin public and private keys, although we do not actually perform any ECDSA signatures. While implemented over elliptic curves, we use the more conventional multiplicative notation (*e.g.*, $y = g^x$ instead of $Y = xG$).

7.3.2 Bitcoin Balance Lookups

We assume that the Bitcoin blockchain is universally agreed upon and all parties can use it to compute the quantity of bitcoin owned by each address. More precisely, for a Bitcoin public key $y \in G$ we use $\text{bal}(y)$ to denote the balance associated with y. We assume $\text{bal}(y)$ is an integer between 0 and MaxBTC for all y. We can represent any bitcoin account with $\mathsf{MaxBTC} = 2^{51}$—the rules of Bitcoin limit the total currency supply to 21M ₿, each divisible into a maximum of 10^{-8} atomic units called *satoshis*. Note that satoshis are the true units of currency in Bitcoin, with ₿$1 = 10^8$ satoshis simply a convention to provide more human-friendly accounting units. In the remainder of this chapter when we speak of account balances we will always be working with satoshis.

7.3.3 Commitments

Provisions makes heavy use of Pedersen commitments [Ped92]. Recall that the commitment to a message $m \in \mathbb{Z}_q$ is defined as $\mathsf{com} = g^m \cdot h^r$ where g and h are fixed public elements of G and the quantity r is chosen at random in $\mathbb{Z}_q$. We use the standard g from secp256k1 and derive h deterministically by hashing the string `Provisions`. Recall that Pedersen commitments are perfectly hiding so that com reveals no information about m.

7.4 Solution: **Provisions** Protocol

7.4.1 Protocol Overview

The objective of Provisions is to enable an exchange $\mathcal{E}$ to publicly prove that it owns enough bitcoin to cover all its customers' balances such that (1) all customer accounts remain fully confidential, (2) no account contains a negative balance, (3) the exchange does not reveal its total liabilities or total assets, and (4) the exchange does not reveal its Bitcoin addresses. Provisions consists of three main protocols:

Protocol 7.1 - Proof of assets. In this protocol, the exchange selects a large set of public keys $\mathbf{PK}$ from the blockchain that hold bitcoin to serve as an anonymity set for its own keys. The exchange possesses the private keys to a subset of the public keys in $\mathbf{PK}$. Next, the exchange creates a commitment to its total assets and proves in zero-knowledge that the sum of balances held by the public keys it owns (i.e. public keys for which it knows the secret key) is equal to the committed value. This is done without revealing which public keys it owns.

Protocol 7.2 - Proof of liabilities. In this protocol, the exchange publishes a commitment to each user's account balance, revealing to each user individually the random factors used to commit to the balance for their verification. For each committed balance, it also proves it is a small positive integer. These committed values are summed homomorphically to produce a commitment to the exchange's total liabilities.

Protocol 7.3 - Proof of solvency. Using the commitments to its total assets and liabilities produced by the above two protocols, the exchange will homomorphically compute a commitment to their difference and prove in zero-knowledge that this final commitment is a commitment to zero. This will prove that the total liabilities is exactly equal to the total assets (or, via a minor modification, that it is strictly less than the total assets).

7.4.2 Proof of Assets

We begin with Protocol 7.1 which lets the exchange $\mathcal{E}$ generate a commitment to its total assets along with a zero-knowledge proof that the exchange knows the private keys for a set of Bitcoin addresses whose total value is equal to the committed value.

The exchange $\mathcal{E}$ chooses a set of Bitcoin public keys

$$\mathbf{PK} = \{y_1, \ldots, y_n\} \subseteq G$$

that will serve as an anonymity set (we will discuss choosing this in Section 7.6). We let $x_1, \ldots, x_n \in \mathbb{Z}_q$ be the corresponding secret keys so that $y_i = g^{x_i}$ for $i = 1, \ldots, n$.

Let $\mathbf{S}$ be the exchange's own set of Bitcoin addresses for which it knows the private keys. The anonymity set $\mathbf{PK}$ must of course be a superset of the exchange's own Bitcoin addresses so that $\mathbf{S} \subseteq \mathbf{PK}$.

We use the booleans $s_i \in \{0, 1\}$ to indicate which accounts the exchange controls in $\mathbf{PK}$. We set $s_i = 1$ whenever the exchange knows the private key x_i for Bitcoin public key $y_i \in \mathbf{PK}$. The exchange's total assets can then be expressed as

$$\mathsf{Assets} = \sum_{i=1}^{n} s_i \cdot \mathrm{bal}(y_i)$$

Finally, it will be convenient to define

$$b_i = g^{\mathrm{bal}(y_i)} \quad \text{for } i = 1, \ldots, n.$$

Given the set $\mathbf{PK}$, a verifier can easily compute all the b_i for itself using information in the Bitcoin blockchain.

Proof of Assets Σ-Protocol

The exchange constructs Pedersen commitments to each $s_i \cdot bal(y_i)$ for $i \in [1, n]$ by choosing a random $v_i \in \mathbb{Z}_q$ and computing

$$p_i = h^{v_i} \cdot b_i^{s_i} \ . \tag{10}$$

A homomorphic addition of these commitments yields a Pedersen commitment $Z_{\textsf{Assets}}$ to $\textsf{Assets}$:

$$Z_{\textsf{Assets}} = \prod_{i=1}^{n} p_i = \prod_{i=1}^{n} h^{v_i} \cdot b_i^{s_i} = h^{(\sum_{i=1}^{n} v_i)} g^{\textsf{Assets}} . \tag{11}$$

It remains to prove in zero-knowledge that $Z_{\textsf{Assets}}$ is valid. To do so the exchange publishes a few additional auxiliary values. For each $i \in [1, n]$ the exchange chooses a random $t_i \in \mathbb{Z}_q$ and publishes

$$l_i = y_i^{s_i} h^{t_i} \quad \in G \tag{12}$$

which is a Pedersen commitment for s_i. Equivalently, these l_i can be written as

$$l_i = g^{x_i \cdot s_i} h^{t_i}$$

which is a Pedersen commitment to the quantity $x_i \cdot s_i \in \mathbb{Z}_q$. By setting $\hat{x}_i = x_i \cdot s_i$ the equation can be written as

$$l_i = g^{\hat{x}_i} h^{t_i} \tag{13}$$

Now, to prove that $Z_{\textsf{Assets}}$ is a commitment to the exchange's assets the exchange needs to prove that for every $i \in [1, n]$ it knows $s_i \in \{0, 1\}, \; v_i, t_i, \hat{x}_i \in \mathbb{Z}_q$ satisfying conditions (10), (12), and (13). $Z_{\textsf{Assets}}$ can then be computed according to (11).

The exchange proves knowledge of the required values using the Σ-protocol presented in Protocol 7.1 along with a Σ-protocol to prove that each s_i is binary and known to the exchange. Proving in zero-knowledge that a Pedersen commitment l_i is a commitment to a binary value is a standard zero-knowledge proof.

The protocol can be made non-interactive using the standard Fiat-Shamir heuristic. It therefore suffices to prove that the protocol is honest-verifier zero knowledge. This is captured in the following theorem:

Theorem 1 *The Σ-protocol in Protocol 7.1 is a honest-verifier zero knowledge proof of knowledge of quantities*

$$\textsf{Assets} \; \textit{and} \; (s_i \in \{0, 1\}, \quad v_i, t_i, \hat{x}_i \in \mathbb{Z}_q) \;\; \textit{for} \; i \in [1, n]$$

that satisfy conditions (10),(11), (12) and (13) for all $i \in [1, n]$.

1. For $i \in [1, n]$
   (a) $\mathcal{E}$ chooses $u_i^{(1)}, u_i^{(2)}, u_i^{(3)}, u_i^{(4)} \xleftarrow{\$} \mathbb{Z}_q$.
   (b) The exchange $\mathcal{E}$ sends to the verifier:

$$a_i^{(1)} = b_i^{u_i^{(1)}} h^{u_i^{(4)}}, \qquad a_i^{(2)} = y_i^{u_i^{(1)}}$$
$$a_i^{(3)} = h^{u_i^{(2)}}, \quad a_i^{(4)} = g^{u_i^{(3)}}$$

   (c) The verifier replies with a challenge $c_i \xleftarrow{\$} Z_q$
   (d) $\mathcal{E}$ replies with:

$$\begin{aligned} r_{s_i} &= u_i^{(1)} + c_i \cdot s_i, && \text{Response for } s_i \\ r_{t_i} &= u_i^{(2)} + c_i \cdot t_i, && \text{Response for } t_i \\ r_{\hat{x}_i} &= u_i^{(3)} + c_i \cdot \hat{x}_i, && \text{Response for } \hat{x}_i \\ r_{v_i} &= u_i^{(4)} + c_i \cdot v_i, && \text{Response for } v_i \end{aligned}$$

   (e) The verifier accepts if:

$$\begin{aligned} b_i^{r_{s_i}} h^{r_{v_i}} &\stackrel{?}{=} p_i^{c_i} a_i^{(1)} && \text{Verify statement (10)} \\ y_i^{r_{s_i}} h^{r_{t_i}} &\stackrel{?}{=} l_i^{c_i} a_i^{(2)} a_i^{(3)} && \text{Verify statement (12)} \\ g^{r_{\hat{x}_i}} h^{r_{t_i}} &\stackrel{?}{=} l_i^{c_i} a_i^{(3)} a_i^{(4)} && \text{Verify statement (13)} \end{aligned}$$

   (f) Run a zero knowledge proof on l_i to prove knowledge of $s_i \in \{0, 1\}$
2. The verifier computes $Z_{\mathsf{Assets}} = \prod_{i=1}^{n} p_i$ Statement (11)

Protocol 7.1: Privacy-preserving proof of assets

(The proof of Theorem 1 is presented in the technical report [DBB+15a]).

The proof of knowledge convinces the verifier that Z_{Assets} is a commitment to the exchange's total assets. More precisely, the verifier is convinced that

- Z_{Assets} is a commitment to $\sum_{i=1}^{n} s_i \cdot \mathrm{bal}(y_i) \in \mathbb{Z}_q$ (by equation (11)), where $s_i \in \{0, 1\}$, and
- whenever $s_i = 1$ the exchange knows the corresponding private key $x_i \in \mathbb{Z}_q$. To see why observe that dividing equation (12) by (13) proves that when $s_i = 1$ the exchange knows $\hat{x}_i \in \mathbb{Z}_q$ such that $g^{\hat{x}_i} = y_i$, as required.

That the proof is honest-verifier zero knowledge implies that nothing is revealed about the total assets, the s_i, or the x_i, as required.

Proof length. The proof is linear in the anonymity set size n, requiring about $13n$ elements in $\mathbb{Z}_q$. This is feasible even for large anonymity sets. We will discuss practical parameters in Section 7.6.

7.4.3 Proof of Liabilities

Protocol 7.2 enables the exchange $\mathcal{E}$ to verifiably commit to its total liabilities and convince all clients that their balances were included in the commitment.

To provide some intuition behind the design of Protocol 7.2, consider the mapping of real customers to entries on $\mathsf{LiabList}$. Each real customer should have an entry in $\mathsf{LiabList}$ (*i.e.,* the mapping is a function) and no distinct customers should be given the same entry (*i.e.,* the mapping should be injective). Perhaps it would be ideal if all entries would correspond to customers (*i.e.,* the mapping were surjective) however this property cannot be enforced—$\mathcal{E}$ can always add fake users to the list, but we ensure that doing so can only increase $\mathcal{E}$'s apparent liabilities.[9]

If two users have the same balance, a malicious $\mathcal{E}$ might try to point both users to the same entry—in the voting literature, this is called a clash attack [VTK12]. To ensure an injective mapping, customers are provided an ID in line 1e which commits[10] to unique information about the customer $\mathsf{username}_i$ (which may include their username, email address, and/or account number). The commitment is binding, preventing the exchange from opening a CID to distinct data for different users. It is also hiding, preventing an adversary who knows the email address of a potential customer from determining if that customer is in $\mathsf{LiabList}$ (or if a user is known to be a customer, which CID they correspond to).

The exchange can add arbitrary accounts to the list. However, as long as accounts can only add to the total liabilities (*e.g.,* $\mathcal{E}$ cannot commit to a negative balance and assign it to a fake

[9] It might be in $\mathcal{E}$'s interest to include fake users with a zero (or tiny) balance to obscure the total number of customers it truly has.

[10] Unlike the other commitments used in $\mathsf{Provisions}$, the commitment scheme used to produce CID_i need only be binding and hiding, not additively homomorphic. We use a simpler hash-based commitment scheme instead of Pedersen commitments.

To verifiably compute its liabilities, $\mathcal{E}$ does:

1. For each customer $\mathcal{C}_i : 1 \leq i \leq c$:
   (a) Represent each $\mathcal{C}_i$'s balance $\mathsf{Balance}_i$ as an m-bit binary number (where $m = \lceil \lg_2 \mathsf{MaxBTC} \rceil$):
   $$\mathsf{BinBalance}_i = \langle x_{i,0}, x_{i,1}, \ldots, x_{i,m-1} \rangle, \qquad \text{(then } \mathsf{Balance}_i = \textstyle\sum_{k=0}^{m-1} x_{i,k} \cdot 2^k\text{)}$$
   (b) Compute and publish a Pedersen commitment to each $x_{i,k}$ in the group G using generators g and h:
   $$y_{i,k} = g^{x_{i,k}} h^{r_{i,k}}, \quad r_{i,k} \xleftarrow{\$} \mathbb{Z}_q$$
   (c) Compute a non-interactive proof of knowledge Π_i of all $r_{i,k}$ and $x_{i,k}$, and that every $x_{i,k}$ is binary.
   (d) Compute a commitment to $\mathcal{C}_i$'s balance as $y_i = \prod_{k=0}^{m-1} (y_{i,k})^{(2^k)} \in G$.
   Then y_i is a Pedersen commitment to $\mathsf{Balance}_i$ because $y_i = g^{\mathsf{balance}_i} h^{r_i}$ where $r_i = \sum_{k=0}^{m-1} r_{i,k} \cdot 2^k$.
   (e) Compute a fresh customer identifier CID_i by picking a random nonce n_i and committing $\mathcal{C}_i$'s username: $\mathsf{CID}_i \xleftarrow{\$} \mathsf{commit}\left(\mathsf{username}_i, n_i\right)$
2. Homomorphically add the commitments to all customers balance into a single commitment to the total liabilities:
   $$Z_{\mathsf{Liabilities}} = \prod_{i=1}^{c} y_i$$
3. Publish the commitment to total liabilities $Z_{\mathsf{Liabilities}}$ and the list $\mathsf{LiabList}$ of all customers' tuples:
   $$\mathsf{LiabList} = \langle \mathsf{CID}_i, y_{i,0}, \ldots, y_{i,m-1}, \Pi_i \rangle \text{ for } i = 1, \ldots, c.$$
4. Every client $\mathcal{C}_i$, upon login, is privately given $\mathsf{username}_i, r_i$ and a string n_i' to open the commitment CID_i. The client verifies that $\mathsf{username}_i = \mathsf{open}\left(\mathsf{CID}_i, n_i'\right)$ and locates it in $\mathsf{LiabList}$. The client then verifies that its balance is included as follows:
   (a) compute $y_i = \prod_{k=0}^{m-1} (y_{i,k})^{(2^k)}$ and verify that $y_i = g^{\mathsf{balance}_i} h^{r_i}$,
   (b) verify that $Z_{\mathsf{Liabilities}} = \prod_{i=1}^{c} y_i$, and
   (c) verify the proof Π_i for $i = 1, \ldots, c$.

   Note that steps (b) and (c) can be carried out by any public auditor and need not be done by every client.

Protocol 7.2: Privacy-preserving proof of liabilities user account), adding accounts is detrimental to a malicious $\mathcal{E}$'s goal as it could only increase its apparent liabilities. Since negative numbers do not technically exist in modular arithmetic, the precise requirement is that when added together, the sum will never cause a reduction $\bmod q$ where $q \approx 2^{256}$ for our group $G =$secp256k1.

To enforce this, $\mathcal{E}$ provides a range proof (adapted from [Mao98]) for each committed balance showing it is from a 'small' interval between 0 and $\mathsf{MaxBTC} = 2^{51}$. This makes it easy to ensure a modular reduction will never occur, as long as the exchange has fewer than 2^{205} accounts.

The range proof works by providing a bit-by-bit commitment of the account balance in binary representation, proving each bit is a 0 or 1 (using the proof of knowledge, mentioned above, twice with conjunctive logic [CDS94]), and showing how many bits the number contains (an upper-bound on its maximum value). This committed binary representation is homomorphically converted into an integer and homomorphically summed.

Theorem 2 *Protocol 7.2 is a honest-verifier zero knowledge proof of knowledge of quantities* Liabilities *and*

$$(x_{i,k} \in \{0,1\}, \quad r_{i,k} \in \mathbb{Z}_q) \quad \text{for } i \in [1,c] \text{ and } k \in [0,m-1]$$

that satisfy the condition

$$Z_{\mathsf{Liabilities}} = \prod_{i=1}^{c} y_i = \prod_{i=1}^{c}\prod_{k=0}^{m-1} (y_{i,k})^{(2^k)} = \prod_{i=1}^{c}\prod_{k=0}^{m-1} (g^{x_{i,k}} h^{r_{i,k}})^{(2^k)}$$

for all $i \in [1,c]$ *and* $k \in [0,m-1]$.

(The proof of Theorem 2 is presented in the technical report [DBB+15a]).

This step leads to the bulk of the proof size (see Section 7.6). Zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs) [BSCG+13] can be used as an alternate version of this protocol. The proof generated by this protocol, however, is significantly shorter (constant in the number of users) at the expense of a large common reference string, the use of heavier cryptographic tools and a trusted setup step.

7.4.4 Customer Verification

We assume that customers each verify $\mathsf{LiabList}$ to confirm the existence of their accounts and the correctness of their balances y_i and ID commitments CID_i. A malicious $\mathcal{E}$ which omits some customers will only be detected if at least one of those customers checks, although this is an inherent limitation given our privacy goals which require that only customers themselves can tell if their balance has been included or not. This limit applies equally, for example, to Maxwell's protocol.

The required checks from individual customers are fortunately quite lightweight. Each customer $\mathcal{C}_i$ receives from $\mathcal{E}$ their $\mathsf{username}_i$, r_i and n_i. They then locate in $\mathsf{LiabList}$, with a hint from $\mathcal{E}$, their tuple:

$$\langle \mathsf{CID}_i, y_{i,0}, \ldots, y_{i,m-1}, \Pi_i \rangle$$

Using n_i, they can open their commitment CID_i and verify that it commits to $\mathsf{username}_i$. Next, using r_i the customer checks that y_i is indeed a commitment to their true account balance $\mathsf{Balance}_i$. This is shown in Step (4a) and is a simple calculation.

The other two verification steps, (4b) and (4c), can be carried out by any party—we assume a public auditor will do so on behalf of most customers, so that individuals will typically not verify the entire proof (though they are free do to so). We discuss the cost of verifying the entire proof further in Section 7.6.

7.4.5 Proof of Solvency

1. $\mathcal{E}$ runs Protocol 7.1 to verifiably generate a commitment Z_{Assets} to its total assets.
2. $\mathcal{E}$ runs Run Protocol 7.2 to verifiably generate a commitment $Z_{\mathsf{Liabilities}}$ to its total assets and a list $\mathsf{LiabList}$ of its liabilities.
3. $\mathcal{E}$ computes $Z_{\mathsf{Assets}} \cdot {Z_{\mathsf{Liabilities}}}^{-1} = Z_{\mathsf{Assets-Liabilities}}$.
4. $\mathcal{E}$ proves in zero-knowledge that $Z_{\mathsf{Assets-Liabilities}}$ is a commitment to the value 0.

Protocol 7.3: Complete privacy-preserving proof of solvency

Protocol 7.3 specifies how $\mathcal{E}$ can complete the proof of solvency given commitments to total assets and liabilities from Protocols 7.1 and 7.2. The proof that $Z_{\mathsf{Assets-Liabilities}}$ is a commitment to 0 (line 4) is a simple Schnorr ZK proof of knowledge of the discrete log of $Z_{\mathsf{Assets-Liabilities}}$ to the base h, since $Z_{\mathsf{Assets-Liabilities}} = g^0 h^k$ for a value k known to the exchange and if $Z_{\mathsf{Assets-Liabilities}}$ were a commitment to any other value then computing its discrete log to the base h would reveal the discrete log of h relative to g.

Variation for exchanges with a surplus If the exchange is actually running a surplus (total assets are greater than total liabilities), this can easily be handled with a simple modification—the exchange can create a commitment to its surplus, Z_{Surplus}, and apply the same range proof used for customer balances to prove that this is a small positive number. It then replaces line 3 in Protocol 7.3 with:

$$Z_{\mathsf{Assets}} \cdot Z_{\mathsf{Liabilities}}{}^{-1} \cdot Z_{\mathsf{Surplus}}{}^{-1}$$

This approach reveals that a surplus exists. The exchange can also prove the magnitude of its surplus if desired by opening the commitment Z_{Surplus}. Alternatively, to hide even the existence of any surplus, the exchange could simply move its surplus into a separate address which is not included in the addresses $\mathbf{S}$ used in its proof of assets, or include the value of the surplus in a number of fake customers' accounts which will add to its apparent liabilities.

Variation for fractional-reserve exchanges Fractional reserve banking, in which an exchange promises to keep assets equal to only a fraction ρ of its total liabilities instead of all of them, has been frowned upon by many in the Bitcoin community and not seen significant deployment. However if this approach becomes more popular in the future, it is easy to modify Provisions to handle this case by modifying Protocol 7.3 to commit to a modified balance $f_i(\mathsf{Balance}_i)$ instead of the customer's true balance $\mathsf{Balance}_i$. Each user can then check during verification that f_i was computed correctly on their true balance. Simple fractional reserves could be implemented by defining $f_i(x) = \rho \cdot x$ for all users. It would also be straightforward to define $f_i(x) = \rho_i \cdot x$ with a different ρ_i for each user if, for example, some users' accounts are fully-guaranteed ($\rho_i = 1$) while others are only fractionally-guaranteed ($\rho_i < 1$). Arbitrary other functions are possible, with a natural example from traditional finance being guaranteeing a user's assets up to some maximum value.

Finally, an exchange can also prove that it is running a surplus of proportion ρ by setting $f_i(x) = (1 + \rho) \cdot x$, with a "fractional surplus" effectively being the inverse of a fractional reserve.

7.4.6 Proof of Non-Collusion

Recall from Section 7.2.3 that the privacy guarantees of Provisions introduce the risk that a cabal of insolvent exchanges colluding by covering each exchanges' individual liabilities with their collective assets. In effect, the assets of a single Bitcoin address can be used in the proof of solvency for multiple exchanges. This can be done by having the exchanges contribute to a set of joint NIZKPs of their keys (*e.g.,* using divertable ZK [BFP$^+$01]).

The simplest defense is for each exchange to choose an anonymity set **PK** which is smaller than the set of all public keys and where each exchange's set is disjoint from the anonymity set of all other exchanges. This ensures that each exchange is proving solvency using assets it owns and without the help of other exchanges. The difficulty with this approach is that there may not be sufficiently many addresses on the Bitcoin blockchain to accommodate strong privacy for all the exchanges. In the long run, if exchanges come to collectively control the majority of all bitcoins, we would like them to be able to use each other as an anonymity set.

Extension to Proof of Assets We can obtain a stronger defense by extending Protocol 7.1 with a few additional steps. Our goal is to ensure that the assets of every Bitcoin address is used in at most one proof of solvency. Recall that the exchange has a set of Bitcoin signing keys $\mathbf{PK} = \{y_1, \ldots, y_n\}$ where $y_i = g^{x_i}$ for $i \in [1, n]$. The exchange knows the secret keys x_i for some subset of these public keys. We use indicator variables $s_1, \ldots, s_n \in \{0, 1\}$ such that $s_i = 1$ when the exchange knows the secret key x_i and $s_i = 0$ otherwise.

We extend Protocol 7.1 to force every exchange to also compute the list $\mathbf{L} = \{h^{\hat{x}_i} =$

$h^{x_i \cdot s_i}$ for $i \in [1, n]\}$ which is randomly permuted and published. Note that when $s_i = 1$ the corresponding element in $\mathbf{L}$ is h^{x_i} and when $s_i = 0$ the corresponding element is simply $1 \in G$, the identity element. Thus $\mathbf{L}$ is a random permutation of the exchange's Bitcoin public keys, but using the base h instead of g.

We require the exchange to prove that $\mathbf{L}$ is correctly constructed (*i.e.,* a permutation of $h^{\hat{x}_1}, \ldots, h^{\hat{x}_n}$) using a zero knowledge proof used as a component of the Neff mix net [Nef01]. That zero-knowledge proof is used to prove that a given list $\ell_2 = \{h^{z_1}, \ldots, h^{z_n}\}$ is a permutation and base change of another given list $\ell_1 = \{g^{z_1}, \ldots, g^{z_n}\}$. This Neff proof thus proves that the published list $\mathbf{L}$ is constructed correctly. It is a simple and efficient proof, requiring $8n$ group elements (8 for each account) and $4n$ additional exponentiations during construction and verification.

We show below that the list $\mathbf{L}$ reveals no information about the $\mathcal{E}$'s Bitcoin addresses beyond the number of addresses ν controlled by $\mathcal{E}$. Note that ν is not revealed by the basic protocol (Protocol 7.1). We'll return to the implications of making this information public in Section 7.5.2 but this is one reason (in addition to added complexity) why we present this as an optional protocol extension.

Now, suppose two exchanges collude and use the same Bitcoin address $y = g^x$ in their proof of solvency. Then h^x will appear in the $\mathbf{L}$ list of both exchanges. In other words, the $\mathbf{L}$ lists of these two exchanges will have a non-trivial intersection.

Since every exchange is required to publish its list $\mathbf{L}$, an auditor can simply check that these lists are mutually disjoint (ignoring the elements $1 \in G$). If so, then the auditor is assured that every Bitcoin address is used in at most one proof of solvency and this holds even if all the exchanges use *the same* anonymity set $\mathbf{PK}$.

An important security requirement is that all exchanges run the extension at the same time—barring this, a simple attack is for exchanges to move bitcoins from one address to another in between runs of the protocol so that the same funds can be used but with a different value for $h^{\hat{x}_i} = h^{x_i \cdot s_i}$ in each $\mathbf{L}$ (since x_i will have changed). Fortunately, the blockchain already provides an easy method of synchronization. Exchanges simply need to agree on a common block number (say, every 240th block to run the protocol daily) and all run the protocol based on the state of the blockchain up to that block. No further synchronization is required; all exchanges can run the protocol and publish their proofs independently and any assets used by more than one exchange will be detectable.

It remains to argue that the list $\mathbf{L}$ reveals no information about the exchange's Bitcoin addresses beyond the number of addresses. This follows directly from the Decision Diffie-Hellman (DDH) assumption which is believed to hold in the secp256k1 group. DDH states that given the tuple $\langle g, h, h^x \rangle$, the quantity g^x is computationally indistinguishable from a random element of G. Therefore, given the list $\mathbf{L}$ it is not possible to distinguish the n-bit string $(s_1, \ldots, s_n) \in \{0,1\}^n$ from a random bit string of the same length.

7.5 Security Discussion

7.5.1 Security Definition

We now present a general definition of a privacy-preserving proof of solvency. We say a function $\nu(k)$ is negligible if for all positive polynomials $p(\cdot)$, there is a sufficiently large k such that $\nu(k) < 1/p(k)$.

Let $\mathcal{A}$ and $\mathcal{A}'$ denote mappings $(y = g^x) \mapsto \mathrm{bal}(y)$ where $\mathcal{A} \subseteq \mathcal{A}'$, y is the public key corresponding to a Bitcoin address with private key x and $\mathrm{bal}(y)$ is the amount of currency, or assets, observably spendable by this key on the blockchain.

Let $\mathcal{L}$ denote a mapping $\mathrm{ID} \mapsto \ell$ where ℓ is the amount of currency, or liabilities, owed by the exchange to each user identified by the unique identity ID.

Definition 19 (Valid Pair) We say that $\mathcal{A}$ and $\mathcal{L}$ are a *valid pair* with respect to a positive integer MaxBTC iff $\forall \mathrm{ID} \in \mathcal{L}$,

1. $\sum_{y \in \mathcal{A}} \mathcal{A}[y] - \sum_{\mathrm{ID} \in \mathcal{L}} \mathcal{L}[\mathrm{ID}] \geq 0$ and

2. $0 \leq \mathcal{L}[\mathrm{ID}] \leq \mathsf{MaxBTC}$. $\square$

Consider an interactive protocol $\mathsf{ProveSolvency}$ run between an exchange $\mathcal{E}$ and user $\mathcal{U}$ such that

1. $\mathsf{output}_{\mathcal{E}}^{\mathsf{ProveSolvency}}(1^k, \mathsf{MaxBTC}, \mathcal{A}, \mathcal{L}, \mathcal{A}') = \emptyset$

2. $\mathsf{output}_{\mathcal{U}}^{\mathsf{ProveSolvency}}(1^k, \mathsf{MaxBTC}, \mathcal{A}', \mathrm{ID}, \ell) \in \{\textsc{Accept}, \textsc{Reject}\}$

For brevity, we refer to these as $\mathsf{out}_{\mathcal{E}}$ and $\mathsf{out}_{\mathcal{U}}$ respectively.

Definition 20 (Privacy-Preserving Proof of Solvency) A privacy-preserving proof of solvency is a probabilistic polynomial-time interactive protocol $\mathsf{ProveSolvency}$, with inputs/outputs as above, such that the following properties hold:

1. **Correctness.** If $\mathcal{A}$ and $\mathcal{L}$ are a valid pair and $\mathcal{L}[\mathrm{ID}] = \ell$, then $Pr[\mathsf{out}_{\mathcal{U}} = \textsc{Accept}] = 1$.

2. **Soundness.** If $\mathcal{A}$ and $\mathcal{L}$ are instead not a valid pair, or if $\mathcal{L}[\mathrm{ID}] \neq \ell$, then $Pr[\mathsf{out}_{\mathcal{U}} = \textsc{Reject}] \geq 1 - \nu(k)$.

3. **Ownership.** For all valid pairs $\mathcal{A}$ and $\mathcal{L}$, if $Pr[\mathsf{out}_{\mathcal{U}} = \textsc{Accept}] = 1$, then the exchange must have 'known' the private keys associated with the public keys in $\mathcal{A}$; i.e., there exists an extractor that, given $\mathcal{A}$, $\mathcal{L}$, and rewindable black-box access to $\mathcal{E}$, can produce x for all $y \in \mathcal{A}$.

4. **Privacy.** A potentially dishonest user interacting with an honest exchange cannot learn anything about a valid pair $\mathcal{A}$ and $\mathcal{L}$ beyond its validity and $\mathcal{L}[\mathrm{ID}]$ (and possibly $|\mathcal{A}|$ and $|\mathcal{L}|$); i.e., even a cheating user cannot distinguish between an interaction using the real pair $\mathcal{A}$ and $\mathcal{L}$ and any other (equally sized) valid pair $\hat{\mathcal{A}}$ and $\hat{\mathcal{L}}$ such that $\hat{\mathcal{L}}[\mathrm{ID}] = \mathcal{L}[\mathrm{ID}]$. □

Theorem 3 *Provisions, as specified in Protocol 7.3, is a privacy-preserving proof of solvency.*

(The proof of Theorem 3 is presented in the technical report [DBB+15a]).

7.5.2 Anonymity Sets

Although Theorem 3 is true, in the case that the protocol extension of Section 7.4.6 is used, the number of Bitcoin addresses ν controlled by the exchange is revealed as well as the size of the anonymity set $n = |\mathbf{PK}|$ (which includes the ν addresses). For efficiency reasons, exchanges may opt to use smaller anonymity sets than the set of all public keys on the blockchain; in particular, if the number of keys grows unexpectedly in the future. In such a case, the exchange must be aware that this might leak some meaningful information about what $\mathcal{E}$'s total assets are.

Specifically, the adversary can determine that $\mathcal{E}$'s assets consist of one of the $\binom{n}{\nu}$ subsets of the anonymity set $\mathbf{PK}$. We remark that $\mathcal{E}$ can easily control n and can also control ν (by splitting accounts up or by padding ν with zero balance accounts). For practical instances, $\binom{n}{\nu}$ grows quickly—*e.g.,* $\nu = 25$ and $n = 250$ already yields $\approx 2^{114}$ candidates. That said, we have no idea what types of external information might be useful for eliminating unlikely or impossible totals from this set (*e.g.,* the adversary's corruption of customers may provides them with a lower bound on the total assets), or for whittling n down by eliminating addresses known or suspected not to be controlled by the exchange. Research on deanonymizing Bitcoin addresses, *e.g.,* through clustering and reidentification [MPJ+13], has demonstrated that Bitcoin's anonymity is limited (see [BMC+15] for a survey).

If an exchange conducts proofs of solvency on a regular basis (or more than once), each anonymity set should be based closely on the anonymity set used previously—choosing independent anonymity sets could reveal the exchange's addresses by intersecting the sets. Exchanges can remove addresses from their anonymity set if the criteria for doing so is independent of whether the exchange owns the address or not. For example, it might remove addresses once the balance is under a certain threshold. However, generally, anonymity sets should grow over time with new addresses (some owned by the exchange and some as cover) being added to the set.

We leave the process of developing and analyzing a heuristic for forming an anonymity set (in terms of size of n and ν and the distribution of amounts across the ν accounts) as future work. For the current state of Bitcoin at the time of writing, we show in Section 7.6 that it is reasonable for all exchanges to choose an anonymity set equal to most available accounts, sieving out tiny "dust" accounts.

7.5.3 User Verification

Although Theorem 3 is true, it may fall short of an ideal level of user verification. Specifically, a proof of solvency *enables* user verification, but it does not guarantee that users actually perform the verification. Consider a malicious $\mathcal{E}$ that does not correctly include some set of users accounts—by either omitting them or zeroing their balances. Assume the exchange has U users, F (for fraudulent) entries, and that a *random* subset $A \subset U$ of users choose to audit the correctness of $\mathsf{LiabList}$. In this case, the probability that an adversary will go undetected is $\binom{U-F}{A}/\binom{U}{A}$, which is closely bounded from above by $\min[(1 - A/U)^F, (1 - F/U)^A]$ (*cf.* the probability of a malicious election authority being caught modifying ballot receipts in a cryptographic voting system [CCC+08]). This probability decreases close-to-exponentially in F and A. Due to the approximation, we conservatively conclude the probability of being caught is high, instead of overwhelming.

Next, one might question the assumption that each customer is equally likely to verify $\mathsf{LiabList}$. However, it is reasonable that the distribution skews in the direction of customers with high balances (and thus more at stake) being more likely to check. This is actually beneficial, because the probability of catching a malicious exchange does not depend on the amount of bitcoin zeroed out. In other words, zeroing out the largest account is equivalent to zeroing out the smallest in terms of being caught, yet the former action better benefits the adversary's goal of lowering its liabilities.

We also note that Provisions as described does not provide dispute resolution. If a user finds their account missing or balance incorrect, they do not have sufficient cryptographic evidence that this is the case [KTV10]. The issue appears unsolvable cryptographically. Recall that the primary motivation for users keeping funds with an exchange is to avoid needing to remember long-term cryptographic secrets, therefore exchanges must be able to execute user orders and change their balance without cryptographic authentication from the user (*e.g.,* password authentication). Resolving this will likely require legal regulation. Users who dislike an exchange may also falsely claim that verification of their accounts failed, and it is not possible to judge if the user or the exchange is correct in this case based on a Provisions transcript alone.

Lastly, we note that if a user does verify their account, they should use a verification tool other than one provided by the exchange itself; such a tool could be automated to increase participation. All of the issues discussed in this remark deserve followup work to ensure that Provisions is implemented in practice in such a way that users are likely to perform auditing and to do so correctly.

7.6 Performance Evaluation

7.6.1 Asymptotic Performance

Provisions scales linearly in proof size, construction and verification time with respect to its inputs: the proof of assets scales with the size of the anonymity set and the proof of liabilities scales with the number of customer accounts. The final proof of solvency given an encryption of the total assets and an encryption of the total liabilities is constant and in practice is negligible. All of the linear parts of the protocol can be run in parallel and require only associative aggregations to compute homomorphic sums, meaning the protocol is straightforward to parallelize.

Specifically the proof of assets is linear in n, the number of public keys in the anonymity set, regardless of the size of $\mathbf{S}$, the total number of accounts actually owned by $\mathcal{E}$, requiring $13n$ integers from $\mathbb{Z}_q$ in total. The proof of liabilities is linear with respect to the number of customers c. It is dominated by $m+1$ elements from $\mathbb{Z}_q$ used to commit to each bit of each customer's balance, where $m = \lceil \lg_2 \mathsf{MaxBTC} \rceil = 51$. If needed, an exchange could slightly reduce proof sizes by capping the size of assets below or reducing precision. For example, with $m = 32$ the exchange could still include accounts worth up to US\$1 billion with precision to the nearest penny. However, we'll assume full precision is desired in our implementation.

Full verification of the protocol requires approximately equal time to the construction of the proof. For customers opting to only validate their own balance's correct inclusion in the proof and trust a third party to run the full verification, verification is much simpler, the customer to check their CID value with a single hash and check that y_i is a correct commitment their balance which requires only $m + 2$ group operations.

7.6.2 Incremental Updates

As described in Section 7.1 the protocol is intended to be run often (e.g. daily) to give continued proof of solvency. A natural question is whether it is possible to update the proof incrementally. We will consider updates to the anonymity set, to the assets proof and to the liabilities proof separately.

The full set of addresses (anonymity set + owned addresses) used in the proof is public. As such any newly created addresses by the exchange need to be published. To hide these new addresses it is important to additionally add addresses to the anonymity set. As with the anonymity set in general and discussed in Section 7.5.2 it is important to choose in such a way that the actual addresses are indistinguishable from it. A proper implementation would for example add addresses deterministically (e.g. all addresses with balances over X bitcoin).

The asset proof is almost perfectly separable, in that there is a separate and independent component for each address in the full set of addresses. The components for new addresses and addresses with changed balance need to be updated. However, it is not necessary to update the components of all other addresses. This is especially useful for cold addresses, which do not have a private key easily accessible. The set of addresses which are new or have changed balances is public on the blockchain anyways and thus no additional information is leaked.

The liabilities proof mainly consists of a commitment to each customer's balance and a proof that said balance is within a range. For all new users and users whose balance changed the commitment the proof needs to be redone. For the other users it is not technically necessary to redo the proof. However, not changing the proofs for customers whose balance remained unchanged will leak how many users were actively using their account between the two proofs. If the complete proof were redone then this information would remain private. If an exchange were to accept this privacy leak it could drastically reduce the size of the proof updates.

7.6.3 Implementation and Setup

To test the performance of our protocol in practice we created a prototype implementation of our protocol in Java 1.8. All cryptographic operations are performed using BouncyCastle,[11] a standard cryptographic library for Java which is also used by the popular bitcoinj implementation of Bitcoin in Java. We performed tests on a commodity server with 2 E5-2680 v2 Xenon processors and 128GB RAM. The max heap size of the JVM was set to the default 256MB. Our implementation assumes a previously downloaded and verified blockchain, to enable efficient balance lookups and selection of an appropriate anonymity set.

An exchange could achieve optimum anonymity by choosing the anonymity set **PK** to be the entire set of unclaimed transaction outputs (called the *UTXO set*) which represents all potentially active Bitcoin accounts. The size of the UTXO set has steadily increased throughout Bitcoin's history [BMC+15] and at the time of this writing contains approximately 17M addresses. However, the vast majority of these are "dust" addresses holding only a tiny value. There are fewer than 500,000 addresses with a balance of more than 0.1 BTC, which collectively control 99.8% of all bitcoin.[12] Some of these addresses are unusable for the protocol because they do not have public keys available (*i.e.,* they are pay-to-pub-key-hash addresses with only a hash of the public key visible in the block chain), others have questionable anonymity value as they have never been moved since being mined and exchanges are not expected to be mining their own bitcoin directly. Thus, we expect that fewer than a million addresses are available to be used in the anonymity set in practice We tested our implementation with anonymity sets up to 500,000.

On the proof of liabilities side, Coinbase is thought to be one the largest exchanges and currently claims roughly 2 million customers.[13] We take as our goal supporting this number of users.

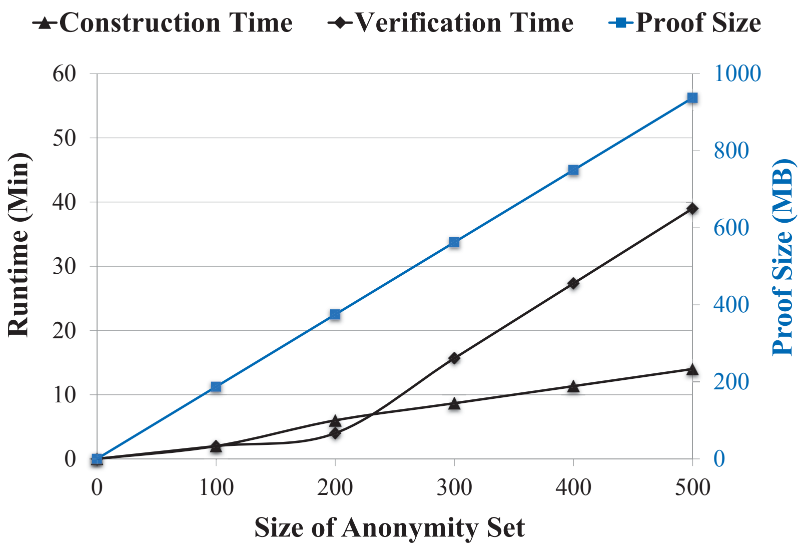


Figure 24: Performance of Protocol 7.1 (proof of assets).

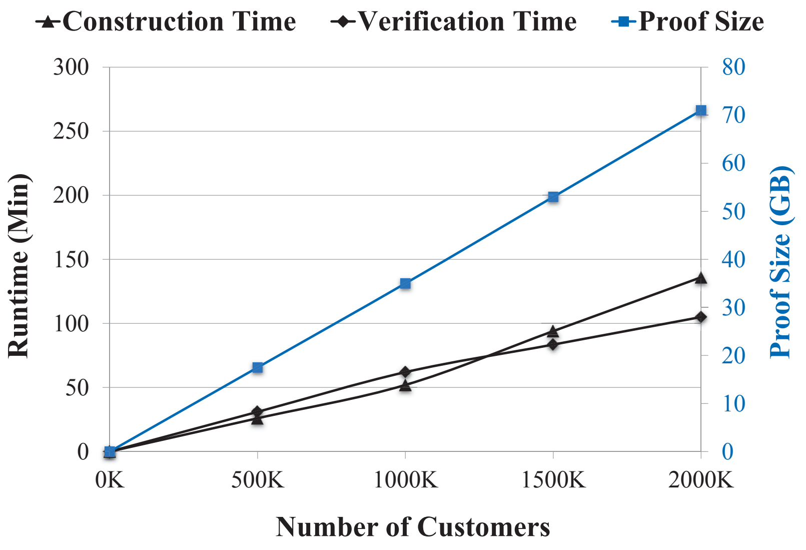


Figure 25: Performance of Protocol 7.2 (proof of liabilities).

7.6.4 Experimental Results

Our experiments confirm that Provisions should be practical even for large exchanges desiring strong anonymity and full precision to represent customer accounts. Figure 24 shows proof sizes and computation times for Protocol 7.1, the proof of assets, varying the anonymity set size n from 10 to 500,000. Figure 25 shows proof sizes and computation times for Protocol 7.2, the proof of liabilities, varying the number of customers c from 1,000 to 2,000,000. We tested with $m = 51$, supporting full precision of account balances. Reducing m would lead to proportional reductions in proof sizes and construction times. Note that, given realistic parameters today, it appears that the proof of liabilities is the more expensive protocol today for a large exchange.

We report numbers without the protocol extension from Section 7.4.6 to ensure assets are not shared between colluding exchanges executing the protocol contemporaneously. This extensions would increase the size and construction time of the proof of assets by about $\frac{4}{13} \approx 30\%$. Because the proof of liabilities is likely much larger, this extension makes only a minor impact on performance.

We omit performance figures for Protocol 7.3 as this protocol is constant size and negligible compared to Protocols 7.1 and 7.2. Similarly, verification time for individual clients depends only m and not the anonymity set or number of other customers. In our implementation it took fewer than 10 ms.

7.7 Summary

Stu Feldman has outlined a roadmap for technical maturity (as quoted in [Gee01]):

1. You have a good idea;
2. You can make your idea work;
3. You can convince a (gullible) friend to try it;
4. People stop asking why you are doing it; and

[11] https://www.bouncycastle.org/
[12] https://bitinfocharts.com/top-100-richest-bitcoin-addresses.html
[13] https://www.coinbase.com/about

5. Other people are asked why they are not doing it.

Given the shaky track record of Bitcoin exchanges, the onus upon an exchange to perform some kind of audit is nearing level 5. However, cryptographic solvency proofs, like the Maxwell protocol, are lagging behind around level 3. Our belief is that the privacy implications of Maxwell are hindering it—there are good reasons for an exchange not to reveal which addresses it controls, the scale of its total holdings, or potentially leak information about large customers' account sizes. Provisions removes these barriers. While cryptographic proofs of solvency still have inherent limits, namely that control of an address' key at present does not guarantee the future ability to use that key to refund customers, we believe that with Provisions there are no longer good reasons for an exchange *not* to provide regular proofs of solvency to increase customer confidence.

Chapter 8

Conclusion

In this thesis, we address the problem of secure data sharing while protecting the sensitive information about the individuals referenced in the data. Inspired by real-life scenarios, we develop multi-party protocols for sharing, integrating, and auditing relational and set-valued data for different application scenarios. The proposed protocols consider security in the semi-honest and malicious threat models, guarantee two privacy models, namely differential privacy and *LKC*-privacy, preserve data utility for data mining, and support public verifiability for set-valued data integration and Bitcoin exchange solvency.

8.1 Summary

We begin by studying the problem of secure and privacy-preserving data outsourcing. Motivated by the process followed by Population Data BC (PopData) for sharing patient-specific health data received from several hospitals, health organizations and government agencies, we propose a secure cloud-based data outsourcing and query processing framework that simultaneously preserves the confidentiality of the data and the query requests, while providing differential privacy guarantee on query outputs. The framework is secure in the semi-honest adversarial model (Chapter 4). Next, we study the problem of secure and privacy-preserving set-valued data integration with public verifiability. We propose a protocol for integrating person-specific data from two or more data owners, while providing differential privacy guarantee and maintaining an effective level of utility on the released data for the purpose of data mining. The protocol is secure in the malicious adversarial model with dishonest majority while supporting public verifiability (Chapter 5). Next, we study the problem of secure and privacy-preserving relational data integration. We propose a protocol for high-dimensional data integration from three or more data owners, with guaranteed LKC-privacy on the output mashup data. The protocol is secure in the semi-honest adversarial model (Chapter 6). Finally, we study the problem of secure and privacy-preserving data auditing in Bitcoin. Motivated by the collapse and ongoing bankruptcy of the oldest and largest exchange, Mt. Gox, which lost over US$450M in customer assets, we propose a cryptographic proof of solvency scheme for Bitcoin exchanges such that no information is revealed about the exchanges customer holdings, the value of the exchanges total holdings is kept secret, and multiple exchanges performing the same proof of solvency can contemporaneously prove they are not colluding. The protocol is secure in the malicious adversarial model with public verifiability (Chapter 7).

In a nutshell, the main contribution of this thesis is to develop secure multi-party protocols for different data sharing scenarios while ensuring different notions of privacy.

8.2 Looking Ahead

Due to the large size and high complexity of the data being collected everyday, the demand for secure and privacy-preserving protocols for data sharing will continue to increase. We believe in the future more researchers will focus on achieving public verifiability in the context of data integration and outsourcing, so processes become more transparent and individuals gain more confidence about the handling of their personal data. We also believe that technological solutions cannot fully solve the problem of secure and privacy preserving data sharing, and there is a need for regulations and policies that define frameworks and guidelines for data sharing.

Weitzner *et al.* [WABL+08] advocated that our society should focus on holding parties responsible for their data usage to offset the problem of security and privacy:

> *"Hide-it-or-lose-it perspective dominates technical and public-policy approaches to fundamental social questions of online privacy, copyright, and surveillance. Yet it is increasingly inadequate for a connected world where information is easily copied and aggregated, and automated correlations and inferences across multiple databases uncover information even when it is not revealed explicitly. As an alternative, accountability must become a primary means through which society addresses appropriate use."*

Future research in data sharing will also focus on *information accountability*, not as an alternative, however, but as a complement to the technical and legislative solutions.

Bibliography

[ABC$^+$08] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. *Journal of Cryptology (JC)*, 21(3):350–391, March 2008.

[ABCK09] M. Adjedj, J. Bringer, H. Chabanne, and B. Kindarji. Biometric identification over encrypted data made feasible. In *Proceedings of the 5th International Conference on Information Systems Security (ICISS)*, pages 86–100, 2009.

[AD74] J. H. Ahrens and U. Dieter. Computer methods for sampling from gamma, beta, poisson and bionomial distributions. *Computing*, 12(3):223–246, 1974.

[ADFH14] M. Arafati, G. G. Dagher, B. C. M. Fung, and P. C. K. Hung. D-mash: A framework for privacy-preserving data-as-a-service mashups. In *Proceedings of the 8th IEEE International Conference on Cloud Computing (CLOUD)*, page 8, June 2014.

[AKSX04] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 563–574, 2004.

[AMFD12] D. Alhadidi, N. Mohammed, B. C. M. Fung, and M. Debbabi. Secure distributed framework for achieving ε-differential privacy. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2012.

[BAAD14] S. Barouti, F. Aljumah, D. Alhadidi, and M. Debbabi. Secure and privacy-preserving querying of personal health records in the cloud. In *Data and Applications Security and Privacy XXVIII (LNCS)*, volume 8566, pages 82–97. 2014.

[BBO07] M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In *Proceedings of the 27th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO)*, pages 535–552, 2007.

[BBS98] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In *Advances in Cryptology - EUROCRYPT (LNCS)*, volume 1403, pages 127–144. 1998.

[BBS04] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Advances in Cryptology - CRYPTO (LNCS)*, volume 3152, pages 41–55. 2004.

[BCK09] J. Bringer, H. Chabanne, and B. Kindarji. Error-tolerant searchable encryption. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 768–773, 2009.

[BDCOP04] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Advances in Cryptology - EUROCRYPT (LNCS)*, volume 3027, pages 506–522. 2004.

[BDDY08] F. Bao, R. H. Deng, X. Ding, and Y. Yang. Private query on encrypted data in multi-user settings. In *Proceedings of the 4th International Conference on Information Security Practice and Experience (ISPEC)*, pages 71–85, 2008.

[BDMN05] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the SuLQ framework. In *Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 128–138, 2005.

[Bel11] M. Belenkiy. E-Cash. In *Handbook of Financial Cryptography and Security*. CRC, 2011.

[Ben75] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[Ben87] J. D. C. Benaloh. *Verifiable Secret-ballot Elections*. PhD thesis, Yale University, 1987.

[BF03] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.

[BFP$^+$01] O. Baudron, P.-A. Fouque, D. Pointcheval, J. Stern, and G. Poupard. Practical multi-candidate election system. In *Proceedings of the 20th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 274–283, 2001.

[BGV11] S. Benabbas, R. Gennaro, and Y. Vahlis. Verifiable delegation of computation over large datasets. In *Proceedings of the 31st annual conference on Advances in cryptology (CRYPTO)*, pages 111–131, 2011.

[BKM05] L. Ballard, S. Kamara, and F. Monrose. Achieving efficient conjunctive keyword searches over encrypted data. In *Proceedings of the 7th International Conference on Information and Communications Security (ICICS)*, pages 414–426, 2005.

[BKOSI07] D. Boneh, E. Kushilevitz, R. Ostrovsky, and W. E. Skeith III. Public key encryption that allows pir queries. In *Advances in Cryptology - CRYPTO (LNCS)*, volume 4622, pages 50–67. 2007.

[BL13] K. Bache and M. Lichman. *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences, 2013.

[BLL06] J. Byun, D. Lee, and J. Lim. Efficient conjunctive keyword search on encrypted data storage system. In *Public Key Infrastructure (LNCS)*, volume 4043, pages 184–196. 2006.

[Blo70] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, July 1970.

[BLS01] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT)*, pages 514–532, 2001.

[BLST10] R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta. Discovering frequent patterns in sensitive data. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2010.

[BM70] R. Bayer and E. McCreight. Organization and maintenance of large ordered indices. In *Proceedings of the ACM SIGFIDET Workshop on Data Description, Access and Control (DDAC)*, pages 107–141, 1970.

[BMC$^+$15] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten. Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. *Proceedings of the IEEE Symposium on Security and Privacy (S & P)*, 2015.

[BNVH12] E.-O. Blass, G. Noubir, and T. D. Vo-Huu. Epic: Efficient privacy-preserving counting for mapreduce. Cryptology ePrint Archive, Report 2012/452, 2012. http://eprint.iacr.org/2003/216/.

[BOO10] A. Beimel, E. Omri, and I. Orlov. Protocols for multiparty coin toss with dishonest majority. In *Proceedings of the 30th Annual Conference on Advances in Cryptology (CRYPTO)*, pages 538–557, 2010.

[BR11] E. Barker and A. Roginsky. Transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths. Technical report, Department of Commerce, 2011.

[Bra06] F. Brandt. Efficient cryptographic protocol design based on distributed el gamal encryption. In *Proceedings of the 8th International Conference on Information Security and Cryptology (ICISC)*, pages 32–47, 2006.

[BSCG+13] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza. Snarks for c: Verifying program executions succinctly and in zero knowledge. In *Advances in Cryptology - CRYPTO (LNCS)*, volume 8043, pages 90–108. 2013.

[BSNS08] J. Baek, R. Safavi-Naini, and W. Susilo. Public key encryption with keyword search revisited. In *Computational Science and Its Applications - ICCSA (LNCS)*, volume 5072, pages 1249–1259. 2008.

[BSW07] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *Proceedings of the IEEE Symposium on Security and Privacy (S & P)*, pages 321–334, 2007.

[BSW11] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *Proceedings of the 8th Conference on Theory of Cryptography (CTC)*, pages 253–273, 2011.

[BTHJ12] C. Bösch, Q. Tang, P. Hartel, and W. Jonker. Selective document retrieval from encrypted database. In *Proceedings of the 15th international conference on Information Security (ISC)*, pages 224–241, 2012.

[BW07] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *Proceedings of the 4th Conference on Theory of Cryptography (CTC)*, pages 535–554, 2007.

[BZ06] M. Barbaro and T. Jr. Zeller. A face is exposed for aol searcher no. 4417749, Aug 2006.

[CCC+08] D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R. L. Rivest, P. Y. A.

Ryan, E. Shen, and A. T. Sherman. Scantegrity II: end-to-end verifiability for optical scan election systems using invisible ink confirmation codes. In *EVTProceedings of the Conference on Electronic Voting Technology (EVT)*, pages 14:1–14:13, 2008.

[CCM09] A. Chakrabarti, G. Cormode, and A. Mcgregor. Annotations in data streams. In *ICALP*, 2009.

[CDN01] R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology (EUROCRYPT)*, pages 280–299, 2001.

[CDS94] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO)*, pages 174–187, 1994.

[Cer00] Certicom Research. SEC 2: Recommended Elliptic Curve Domain Parameters, Version 1.0., 2000.

[CGKO06] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)*, pages 79–88, 2006.

[CGS97] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Proceedings of the 16th annual International Conference on Theory and Application of Cryptographic Techniques (EUROCRYPT)*, pages 103–118, 1997.

[CH10] J. Clark and U. Hengartner. On the use of financial data as a random beacon. In *EVT/WOTE*, 2010.

[Cha82] D. Chaum. Blind signatures for untraceable payments. In *CRYPTO*, 1982.

[CHL05] J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash. In *Advances in Cryptology - EUROCRYPT (LNCS)*, volume 3494, pages 302–321. 2005.

[CJJ+13] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. In *Advances in Cryptology - CRYPTO (LNCS)*, volume 8042, pages 353–373. 2013.

[CK10] M. Chase and S. Kamara. Structured encryption and controlled disclosure. In *Advances in Cryptology - ASIACRYPT (LNCS)*, volume 6477 of *Lecture Notes in Computer Science*, pages 577–594. 2010.

[CKLR11] K.-M. Chung, Y. T. Kalai, F.-H. Liu, and R. Raz. Memory delegation. In *Proceedings of the 31st Annual Conference on Advances in Cryptology (CRYPTO)*, pages 151–165, 2011.

[CKV10] K.-M. Chung, Y. Kalai, and S. Vadhan. Improved delegation of computation using fully homomorphic encryption. In *Proceedings of the 30th annual conference on Advances in cryptology (CRYPTO)*, pages 483–501, 2010.

[CM05] Y-C. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *Proceedings of the 3rd International Conference on Applied Cryptography and Network Security (ACNS)*, pages 442–455, 2005.

[CMF+11] R. Chen, N. Mohammed, B. C. M. Fung, B. C. Desai, and L. Xiong. Publishing set-valued data via differential privacy. *Proceedings of the 37th International Conference on Very Large Data Bases (PVLDB)*, 4(11), 2011.

[Coc01] C. Cocks. An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding (IMACC)*, pages 360–363, 2001.

[Com79] D. Comer. Ubiquitous B-Tree. *ACM Computing Surveys*, 11(2):121–137, 1979.

[CP92] D. Chaum and T. P. Pedersen. Wallet databases with observers. In *CRYPTO*, 1992.

[CPS$^+$12] G. Cormode, C. Procopiuc, D. Srivastava, E. Shen, and T. Yu. Differentially private spatial decompositions. In *Proceedings of the IEEE 28th International Conference on Data Engineering (ICDE)*, pages 20–31, 2012.

[Dal77] T. Dalenius. Towards a methodology for statistical disclosure control. *Statistik Tidskrift*, 15, 1977.

[DBB$^+$15a] G. G. Dagher, B. Bünz, J. Bonneau, J. Clark, and D. Boneh. Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges (full version). Technical report, IACR Cryptology ePrint Archive, 2015.

[DBB$^+$15b] G. G. Dagher, B. Bünz, J. Bonneau, J. Clark, and D. Boneh. Provisions: Private proofs of solvency for bitcoin exchanges. In *Proceedings of the 22nd ACM Conference on Computer and Communications Security (CCS)*, October 2015.

[dBCvKO08] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, 3rd edition, 2008.

[DBP07] G. Di Battista and B. Palazzi. Authenticated relational tables and authenticated skip lists. In *Proceedings of the 21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSec)*, pages 31–46, 2007.

[DCF] G. G. Dagher, J. Clark, and B. C. M. Fung. Publicly verifiable protocol for set-valued data integration with differential privacy. *Proceedings on Privacy Enhancing Technologies (PoPETs)*, page 14. Under review.

[DCFG$^+$14] G. Di Crescenzo, J. Feigenbaum, D. Gupta, E. Panagos, J. Perry, and R. Wright. Practical and privacy-preserving policy compliance for outsourced data. In *Financial Cryptography and Data Security (LNCS)*, volume 8438, pages 181–194. 2014.

[DFMC] G. G. Dagher, B. C. M. Fung, N. Mohammed, and J. Clark. SecDM: A privacy-preserving framework for confidential query processing on the cloud. *IEEE Transactions on Cloud Computing (TCC)*, page 14. Under review.

[DIAF15] G. G. Dagher, F. Iqbal, M. Arafati, and B. C. M. Fung. Fusion: Privacy-preserving distributed protocol for high-dimensional data mashup. In *Proceedings of the 21st IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, page 10, 2015. Accepted.

[DKM$^+$06] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *Proceedings of the 24th Annual International Conference on The Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 486–503, 2006.

[DLL13] J. Dreier, P. Lafourcade, and Y. Lakhnech. Formal verification of e-auction protocols. In *Principles of Security and Trust (LNCS)*, volume 7796, pages 247–266. 2013.

[DMNS06] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd conference on Theory of Cryptography (TCC)*, 2006.

[Don00] D. L. Donoho. Aide-memoire. high-dimensional data analysis: The curses and blessings of dimensionality, 2000.

[DRD08] C. Dong, G. Russello, and N. Dulay. Shared and searchable encrypted data for untrusted servers. In *Proceedings of the 22nd annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy (DBSec)*, pages 127–143, 2008.

[DVJ$^+$03] E. Damiani, S. D. C. Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. Balancing confidentiality and efficiency in untrusted relational dbmss. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS)*, pages 93–102, 2003.

[DWC10] T. Dillon, C. Wu, and E. Chang. Cloud computing: issues and challenges. In *Proceedings of the 24th IEEE Conference on Advanced Information Networking and Applications (AINA)*, pages 27–33, 2010.

[Dwo06] C. Dwork. Differential privacy. In *Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 1–12, 2006.

[EAAG06] F. Emekci, D. Agrawal, A.E. Abbadi, and A. Gulbeden. Privacy preserving query processing using third parties. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE)*, pages 27–36, 2006.

[EBSC15] S. Eskandari, D. Barrera, E. Stobert, and J. Clark. A first look at the usability of bitcoin key management. In *Proceedings of the NDSS Workshop on Usable Security (USEC)*, 2015.

[EG85] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO on Advances in Cryptology*, pages 10–18, 1985.

[EVK05] M. Eirinaki, M. Vazirgiannis, and D. Kapogiannis. Web path recommendations based on page ranking and markov models. In *Proceedings of the ACM Workshop on Web Information and Data Management (WIDM)*, pages 2–9, 2005.

[FH94] M. K. Franklin and S. Haber. Joint encryption and message-efficient secure computation. In *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO)*, pages 266–277, 1994.

[FS87] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings of the 7th Annual Conference on Advances in Cryptology (CRYPTO)*, pages 186–194, 1987.

[FS90] A. Fiat and A. Shamir. Witness indistinguishable and witness hiding protocols. In

Proceedings of the 22st annual ACM Symposium on Theory of Computing (STOC), pages 416–426, 1990.

[FWCY10] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys*, 42(4):14:1–14:53, 2010.

[FWY07a] B. C. M. Fung, K. Wang, and P. S. Yu. Anonymizing classification data for privacy preservation. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 19(5):711–725, 2007.

[FWY07b] B. C. M. Fung, K. Wang, and P. S. Yu. Anonymizing classification data for privacy preservation. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 19(5):711–725, May 2007.

[Gee01] D. Geer. Technical maturity, reliability, implicit taxes, and wealth creation. *login: The magazine of Usenix & Sage*, 26(8), 2001.

[Gen09] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM Symposium on Theory of Computing (STOC)*, pages 169–178, 2009.

[GGP10] R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Proceedings of the 30th annual conference on Advances in cryptology (CRYPTO)*, pages 465–482, 2010.

[GJKR07] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology (JC)*, 20(1):51–83, 2007.

[GKR08] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. Delegating computation: Interactive proofs for muggles. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 113–122, 2008.

[GLM$^+$13a] F. Giannotti, L.V.S. Lakshmanan, A. Monreale, D. Pedreschi, and H. Wang. Privacy-preserving mining of association rules from outsourced transaction databases. *IEEE Systems Journal (ISJ)*, 7(3):385–395, 2013.

[GLM$^+$13b] F. Giannotti, L.V.S. Lakshmanan, A. Monreale, D. Pedreschi, and H. Wang. Privacy-preserving mining of association rules from outsourced transaction databases. *ISJ*, 7(3), 2013.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 218–229, 1987.

[Goh03] E.-J. Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003. `http://eprint.iacr.org/2003/216/`.

[Gol04] O. Goldreich. *Foundations of Cryptography*, volume 2. Cambridge University Press, 2004.

[Gol14] S. Goldfeder. Better wallet security for bitcoin. Technical report, Princeton, March 2014.

[GPSW06] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)*, pages 89–98, 2006.

[GSW04] P. Golle, J. Staddon, and B. Waters. Secure conjunctive keyword search over encrypted data. In *Applied Cryptography and Network Security (LNCS)*, volume 3089, pages 31–45. 2004.

[Gut84] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 47–57, 1984.

[GZ07a] T. Ge and S. Zdonik. Answering aggregation queries in a secure system model. In *Proceedings of the 33rd International Conference on Very Large Data Bases (PVLDB)*, pages 519–530, 2007.

[GZ07b] T. Ge and S. Zdonik. Fast, secure encryption for indexing in a column-oriented DBMS. In *Proceedings of the IEEE 23rd International Conference on Data Engineering (ICDE)*, pages 676–685, 2007.

[HILM02] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra. Executing sql over encrypted data in the database-service-provider model. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 216–227, 2002.

[HIM04] H. Hacigümüş, B. Iyer, and S. Mehrotra. Efficient execution of aggregation queries over encrypted relational databases. In *Proceedings of the Database Systems for Advanced Applications (DASFAA)*, pages 125–136, 2004.

[HL07] Y. H. Hwang and P. J. Lee. Public key encryption with conjunctive keyword search and its extension to a multi-user system. In *Proceedings of the 1st International Conference on Pairing-Based Cryptography (Pairing)*, pages 2–22, 2007.

[HL10] C. Hazay and Y. Lindell. *Efficient Secure Two-Party Protocols.* Springer, 2010.

[HMT04] B. Hore, S. Mehrotra, and G. Tsudik. A privacy-preserving index for range queries. In *Proceedings of the 13th International Conference on Very Large Data Bases (PVLDB)*, pages 720–731, 2004.

[HN09] Y. He and J. F. Naughton. Anonymization of set-valued data via top-down, local generalization. *Proceedings of the 35th International Conference on Very Large Data Bases (PVLDB)*, 2(1), 2009.

[HSM$^+$14] J. Han, W. Susilo, Y. Mu, J. Zhou, and M. Au. Ppdcp-abe: Privacy-preserving decentralized ciphertext-policy attribute-based encryption. In *Computer Security - ESORICS (LNCS)*, volume 8713, pages 73–90. 2014.

[HXL13] Y. Hua, B. Xiao, and X. Liu. Nest: Locality-aware approximate query service for cloud computing. In *Proceedings of the 32nd IEEE Conference on Computer Communications (INFOCOM)*, 2013.

[HXRC11] H. Hu, J. Xu, C. Ren, and B. Choi. Processing private queries over untrusted data cloud through privacy homomorphism. In *Proceedings of the IEEE 27th International Conference on Data Engineering (ICDE)*, pages 601–612, 2011.

[JC06] W. Jiang and C. Clifton. A secure distributed framework for achieving k-anonymity. *The International Journal on Very Large Data Bases (VLDBJ)*, 15(4), 2006.

[JGWY10] X. Jiang, J. Gao, T. Wang, and D. Yang. Multiple sensitive association protection in the outsourced database. In *Proceedings of the 15th International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 123–137, 2010.

[JJ00] M. Jakobsson and A. Juels. Mix and match: Secure function evaluation via ciphertexts. In *Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT)*, pages 162–177, 2000.

[JJK$^+$13] S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. Outsourced symmetric private information retrieval. In *Proceedings of the ACM SIGSAC conference on Computer & Communications Security (CCS)*, pages 875–888, 2013.

[JJR] M. Jakobsson, A. Juels, and R. L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proceedings of the 11th USENIX Security Symposium (USENIX)*, pages 339–353.

[Jou00] A. Joux. A one round protocol for tripartite diffie-hellman. In *Proceedings of the 4th*

International Symposium on Algorithmic Number Theory (ANTS), pages 385–394, 2000.

[JX09] P. Jurczyk and L. Xiong. Distributed anonymization: Achieving privacy for both data subjects and data providers. In *Proceedings of the 23rd annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy (DBSec)*, 2009.

[KC04] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 16(9), 2004.

[KK07] M. Kantarcioglu and O. Kardes. Privacy-preserving data mining applications in the malicious model. In *Proceedings of the IEEE ICDM Workshop on Data Mining (ICDMW)*, pages 717–722, Oct 2007.

[KKP01] S. Kotz, T. Kozubowski, and K. Podgorski. *The Laplace Distribution and Generalizations: A Revisit With Applications to Communications, Exonomics, Engineering, and Finance*. Springer, 2001.

[KL10] D. Kifer and B.-R. Lin. Towards an axiomatization of statistical privacy and utility. In *Proceedings of the 29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 147–158, 2010.

[KMR] S. Kamara, P. Mohassel, and M. Raykova. Outsourcing multi-party computation. *IACR Cryptology ePrint Archive*, 2011:272.

[KPR12] S. Kamara, C. Papamanthou, and T. Roeder. Dynamic searchable symmetric encryption. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS)*, pages 965–976, 2012.

[KSW08] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Proceedings of the 27th annual International Conference on Advances in Cryptology (EUROCRYPT)*, pages 146–162, 2008.

[KTV10] R. Kusters, T. Truderung, and A. Vogt. Accountability: Definition and relationship to verifiability. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS)*, 2010.

[LDLW14] J. Lai, R. H. Deng, Y. Li, and J. Weng. Fully secure key-policy attribute-based encryption with constant-size ciphertexts and fast decryption. In *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security (ASIA CCS)*, pages 239–248, 2014.

[LKM01] B. Lee, K. Kim, and J. Ma. Efficient public auction with one-time registration and public verifiability. In *Progress in Cryptology - INDOCRYPT (LNCS)*, volume 2247, pages 162–174. 2001.

[LQSC12] N. Li, W. Qardaji, D. Su, and J. Cao. Privbasis: Frequent itemset mining with differential privacy. *Proceedings of the 38th International Conference on Very Large Data Bases (PVLDB)*, 5(11):1340–1351, 2012.

[LS08] D. Lubicz and T. Sirvent. Attribute-based broadcast encryption scheme made efficient. In *Proceedings of the 1st International Conference on Progress in Cryptology (AFRICACRYPT)*, pages 325–342, 2008.

[LS14] P. Litke and J. Stewart. Cryptocurrency-stealing malware landscape. Technical report, Dell SecureWorks Counter Threat Unit, 2014.

[LWW$^+$10] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou. Fuzzy keyword search over encrypted data in cloud computing. In *Proceedings of the 29th Conference on Information Communications (INFOCOM)*, pages 441–445, 2010.

[MAFD14] N. Mohammed, D. Alhadidi, B. C. M. Fung, and M. Debbabi. Secure two-party differentially private data release for vertically-partitioned data. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 11(1), 2014.

[Mao98] W. Mao. Guaranteed correct sharing of integer factorization with off-line shareholders. In *Proceedings of the 1st International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography (PKC)*, pages 60–71, 1998.

[MC13] T. Moore and N. Christin. Beware the middleman: Empirical analysis of bitcoin-exchange risk. In *Financial Cryptography and Data Security (LNCS)*, volume 7859, pages 25–33. 2013.

[MCFY11] N. Mohammed, R. Chen, B. C. M. Fung, and P. S. Yu. Differentially private data release for data mining. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 493–501, 2011.

[McS09] F. D. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2009.

[Mer79] R. C. Merkle. *Secrecy, Authentication, and Public Key Systems.* PhD thesis, Stanford, 1979.

[MFD11] N. Mohammed, B. C. M. Fung, and M. Debbabi. Anonymity meets game theory: secure data integration with malicious participants. *The International Journal on Very Large Data Bases (VLDBJ)*, 20(4), 2011.

[MFHL09] N. Mohammed, B. C. M. Fung, P. C. K. Hung, and C-k Lee. Anonymizing health-care data: a case study on the blood transfusion service. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1285–1294, 2009.

[MKGV07] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1), 2007.

[MPJ$^+$13] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 Conference on Internet Measurement Conference (IMC)*, pages 127–140, 2013.

[MT00] G. Marsaglia and W. W. Tsang. A simple method for generating gamma variables. *ACM Transactions on Mathematical Software (TOMS)*, 26(3):363–372, 2000.

[Nak08] S. Nakamoto. Bitcoin: A peer-to-peer electionic cash system. Unpublished, 2008.

[NC11] A. E. Nergiz and C. Clifton. Query processing in private data outsourcing using anonymization. In *Proceedings of the 25th annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy (DBSec)*, pages 138–153, 2011.

[Nef01] C. A. Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS)*, 2001.

[NH12] A. Narayan and A. Haeberlen. Djoin: Differentially private join queries over distributed databases. In *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation (OSDI)*, pages 149–162, 2012.

[NS08] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *Proceedings of the IEEE Symposium on Security and Privacy (S & P)*, pages 111–125, 2008.

[NYO08] T. Nishide, K. Yoneyama, and K. Ohta. Attribute-based encryption with partially hidden encryptor-specified access structures. In *Proceedings of the 6th International*

Conference on Applied Cryptography and Network Security (ACNS), pages 111–129, 2008.

[OSW07] R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS)*, pages 195–203, 2007.

[Par11] R. Parhonyi. Micropayment Systems. In *Handbook of Financial Cryptography and Security*. CRC, 2011.

[PCL05] D. J. Park, J. Cha, and P. J. Lee. Searchable keyword-based encryption. Cryptology ePrint Archive, Report, 2005. http://eprint.iacr.org/2005/367.

[Ped91] T. P. Pedersen. A threshold cryptosystem without a trusted party. In *Proceedings of the 10th Annual International Conference on Theory and Application of Cryptographic Techniques (EUROCRYPT)*, pages 522–526, 1991.

[Ped92] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO)*, pages 129–140, 1992.

[PRZB11] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. Cryptdb: Protecting confidentiality with encrypted query processing. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP)*, pages 85–100, 2011.

[PZ12] K. Peng and Y. Zhang. A secure mix network with an efficient validity verification mechanism. In *Internet and Distributed Computing Systems (LNCS)*, volume 7646, pages 85–96. 2012.

[Qui93] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[Rab79] M. O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, Massachusetts Institute of Technology (MIT), 1979.

[RN10] V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 735–746, 2010.

[RVBM09] M. Raykova, B. Vo, S. M. Bellovin, and T. Malkin. Secure anonymous database search. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security (CCSW)*, pages 115–126, 2009.

[Sal94] S. L. Salzberg. C4.5: Programs for machine learning by j. ross quinlan. *Machine Learning (ML)*, 16(3):235–240, 1994.

[Sam01a] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 13(6):1010–1027, November 2001.

[Sam01b] P. Samarati. Protecting respondents identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 13(6):1010–1027, 2001.

[Sch91] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptography (JC)*, 4(3):161–174, 1991.

[SCR$^+$11] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song. Privacy-preserving aggregation of time-series data. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2011.

[SER12] A. Shabtai, Y. Elovici, and L. Rokach. *A Survey of Data Leakage Detection and Prevention Solutions*. SpringerBriefs in Computer Science (SBCS). Springer, 2012.

[SHKS12] M. Schläpfer, R. Haenni, R. Koenig, and O. Spycher. Efficient vote authorization in coercion-resistant internet voting. In *Proceedings of the 3rd International Conference on E-Voting and Identity (VoteID)*, pages 71–88, 2012.

[SKK09] M. Shaneck, Y. Kim, and V. Kumar. Privacy preserving nearest neighbor search. In *Machine Learning in Cyber Trust (MLCT)*, pages 247–276. 2009.

[SR92] A. Skowron and C. Rauszer. The discernibility matrices and functions in information systems. In *Intelligent Decision Support*, volume 11 of *Theory and Decision Library*, pages 331–362. 1992.

[ST04] B. Schoenmakers and P. Tuyls. Practical two-party computation based on the conditional gate. In *Advances in Cryptology - ASIACRYPT (LNCS)*, volume 3329, pages 119–136. 2004.

[STW$^+$12] E. Shmueli, T. Tassa, R. Wasserstein, B. Shapira, and L. Rokach. Limiting disclosure of sensitive data in sequential releases of databases. *Information Science (IS)*, 191:98–127, 2012.

[SVLN$^+$10] S. Sedghi, P. Van Liesdonk, S. Nikova, P. Hartel, and W. Jonker. Searching keywords with wildcards on encrypted data. In *Proceedings of the 7th International Conference on Security and Cryptography for Networks (SCN)*, pages 138–153, 2010.

[SW05] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Proceedings of the 24th annual International Conference on Advances in Cryptology (EUROCRYPT)*, pages 457–473, 2005.

[Swe02a] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (IJUFKS)*, 10(5):571–588, 2002.

[Swe02b] L. Sweeney. K-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (UFKS)*, 10(5):557–570, October 2002.

[SWP00] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *Proceedings of the IEEE Symposium on Security and Privacy (S & P)*, 2000.

[Tam03] R. Tamassia. Authenticated data structures. In *Algorithms - ESA (LNCS)*, volume 2832, pages 2–5. 2003.

[Tas14] T. Tassa. Secure mining of association rules in horizontally distributed databases. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 26(4), 2014.

[TH13] P.K. Tysowski and M.A. Hasan. Hybrid attribute- and re-encryption-based key management for secure and scalable mobile applications in clouds. *IEEE Transactions on Cloud Computing (TCC)*, 1(2):172–186, July 2013.

[TMK08] M. Terrovitis, N. Mamoulis, and P. Kalnis. Privacy-preserving anonymization of set-valued data. *Proceedings of the 34th International Conference on Very Large Data Bases (PVLDB)*, 1(1), 2008.

[TMK11] M. Terrovitis, N. Mamoulis, and P. Kalnis. Local and global recoding methods for anonymizing set-valued data. *The International Journal on Very Large Data Bases (VLDBJ)*, 20(1), 2011.

[TRN08] V. Teague, K. Ramchen, and L. Naish. Coercion-resistant tallying for stv voting. In *Proceedings of the Conference on Electronic Voting Technology (EVT)*, pages 15:1–15:14, 2008.

[VC05] J. Vaidya and C. Clifton. Privacy-preserving top-k queries. In *Proceedings of the 21st International Conference on Data Engineering (ICDE)*, pages 545–546, 2005.

[VLSD$^+$10] P. Van Liesdonk, S. Sedghi, J. Doumen, P. Hartel, and W. Jonker. Computationally efficient searchable symmetric encryption. In *Proceedings of the 7th VLDB Conference on Secure Data Management (SDM)*, pages 87–100, 2010.

[VTK12] A. Vogt, T. Truderung, and R. Kusters. Clash attacks on the verifiability of e-voting systems. In *Proceedings of the IEEE Symposium on Security and Privacy (S & P)*, 2012.

[WABL$^+$08] D. J. Weitzner, H. Abelson, T. Berners-Lee, J. Feigenbaum, J. Hendler, and G. J. Sussman. Information accountability. *Communications of the ACM*, 51(6):82–87, 2008.

[WAEA11] S. Wang, D. Agrawal, and A. El Abbadi. A comprehensive framework for secure query processing on relational data in the cloud. In *Proceedings of the 8th VLDB International Conference on Secure Data Management (SDM)*, pages 52–69, 2011.

[Wat11] B. Waters. Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In *Proceedings of the 14th International Conference on Practice and Theory in Public Key Cryptography (PKC)*, pages 53–70, 2011.

[WCH$^+$07] W. K. Wong, D. W. Cheung, E. Hung, B. Kao, and N. Mamoulis. Security in outsourcing of association rule mining. In *Proceedings of the 33rd International Conference on Very Large Data Bases (PVLDB)*, 2007.

[WCKM09] W. K. Wong, D. W.-L. Cheung, B. Kao, and N. Mamoulis. Secure knn computation on encrypted databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 139–152, 2009.

[WCL$^+$10] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou. Secure ranked keyword search over encrypted cloud data. In *Proceedings of the IEEE 30th International Conference on Distributed Computing Systems (ICDCS)*, pages 253–262, 2010.

[WCW$^+$13] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou. Privacy-preserving public auditing for secure cloud storage. *IEEE Transactions on Computers (TC)*, 62(2):362–375, 2013.

[WDWS04] Z.-F. Wang, J. Dai, W. Wang, and B.-L. Shi. Fast query over encrypted character data in database. In *Proceedings of the 1st International Conference on Computational and Information Science (CIS)*, pages 1027–1033, 2004.

[WFY07] K. Wang, B. C. M. Fung, and P. S. Yu. Handicapping attacker's confidence: An alternative to k-anonymization. *Knowledge and Information Systems (KAIS)*, 11(3):345–368, 2007.

[WHZ$^+$14] O. A. Wahab, M. O. Hachami, A. Zaffari, M. Vivas, and G. G. Dagher. Darm: A privacy-preserving approach for distributed association rules mining on horizontally-partitioned data. In *Proceedings of the 18th International Database Engineering & Applications Symposium (IDEAS)*, pages 1–8, 2014.

[Wil14] Z. Wilcox. Proving your bitcoin reserves. https://iwilcox.me.uk/2014/proving-bitcoin-reserves, Feb. 2014.

[WL06] H. Wang and L. V. S. Lakshmanan. Efficient secure query evaluation over encrypted xml databases. In *Proceedings of the 32nd International Conference on Very Large Data Bases (PVLDB)*, pages 127–138, 2006.

[WLFW06] R. C.-W. Wong, J. Li, A. W.-C. Fu, and K. Wang. (α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 754–759, 2006.

[WLL12] B. Wang, B. Li, and H. Li. Oruta: Privacy-preserving public auditing for shared data in the cloud. In *Proceedings of the IEEE 5th International Conference on Cloud Computing (CLOUD)*, pages 295–302, 2012.

[WR13] P. Wang and C.V. Ravishankar. Secure and efficient range queries on outsourced databases using $\widehat{R}$-trees. In *Proceedings of the IEEE 29th International Conference on Data Engineering (ICDE)*, pages 314–325, 2013.

[WSC08] P. Williams, R. Sion, and B. Carbunar. Building castles out of mud: Practical access pattern privacy and correctness on untrusted storage. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS)*, pages 139–148, 2008.

[WWP08a] P. Wang, H. Wang, and J. Pieprzyk. Keyword field-free conjunctive keyword searches on encrypted data and extension for dynamic groups. In *Cryptology and Network Security (LNCS)*, volume 5339, pages 178–195. 2008.

[WWP08b] P. Wang, H. Wang, and J. Pieprzyk. Threshold privacy preserving keyword searches. In *SOFSEM 2008: Theory and Practice of Computer Science (LNCS)*, volume 4910, pages 646–658. 2008.

[WWP09] P. Wang, H. Wang, and J. Pieprzyk. An efficient scheme of common secure indices for conjunctive keyword-based retrieval on encrypted data. In *Information Security Applications (LNCS)*, volume 5379, pages 145–159. 2009.

[WWS05] Z.-F. Wang, W. Wang, and B.-L. Shi. Storage and query over encrypted character and numerical data in database. In *Proceedings of the 5th International Conference on Computer and Information Technology (CIT)*, pages 77–81, 2005.

[XT06] X. Xiao and Y. Tao. Anatomy: simple and effective privacy preservation. In *Proceedings of the 32nd International Conference on Very Large Data Bases (PVLDB)*, pages 139–150, 2006.

[XXY10] Y. Xiao, L. Xiong, and C. Yuan. Differentially private data release through multidimensional partitioning. In *Proceedings of the 7th VLDB Conference on Secure Data Management (SDM)*, pages 150–168, 2010.

[Yao82] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (SFCS)*, pages 160–164, 1982.

[Yao86] A. C.-C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science (SFCS)*, pages 162–167, 1986.

[YLW11] Y. Yang, H. Lu, and J. Weng. Multi-user private keyword search for cloud computing. In *Proceedings of the IEEE 3rd International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 264–271, Nov 2011.

[Yon14] W. Yongge. Privacy-preserving data storage in cloud using array bp-xor codes. *IEEE Transactions on Cloud Computing (TCC)*, 99, 2014.

[ZRZ$^+$13] F. Zhang, C.-M. Rong, G. Zhao, J. Wu, and X. Wu. Privacy-preserving two-party distributed association rules mining on horizontally partitioned data. In *Proceedings of the International Conference on Cloud Computing and Big Data (CLOUDCOM-ASIA)*, 2013.