University Library



A gateway to Melbourne's research publications

Minerva Access is the Institutional Repository of The University of Melbourne

Author/s:
Roohi, Leyla

Title:
Privacy-preserving computation on graph-structured data

Date:
2019

Persistent Link:
https://hdl.handle.net/11343/235879

Terms and Conditions:
Terms and Conditions: Copyright in works deposited in Minerva Access is retained by the copyright owner. The work may not be altered without permission from the copyright owner. Readers may only download, print and save electronic copies of whole works for their own personal non-commercial use. Any use that exceeds these limits requires permission from the copyright owner. Attribution is essential when quoting or paraphrasing from these works.

School of Computing and Information Systems
The University of Melbourne

Privacy-Preserving Computation on Graph-Structured Data

Leyla Roohi

Submitted in total fulfilment of the requirements of the degree of Doctor of Philosophy

Produced on archival quality paper

October, 2019

Abstract

There are many examples of graph-structured data, like records of friendships in social networks, travel patterns in transport networks and communications meta-data in telecommunication networks, and many more. In such data, people are represented as nodes and their interactions as edges. Graphs can provide valuable information about people and their connections, however privacy disclosure of performing or releasing graph analysis is a major open challenge. Naive techniques such as anonymisation on edges and nodes have been shown to fall short: heuristically anonymised graphs still leak significant structural information that can be used to match individuals and recover sensitive information. In this thesis, we address important privacy disclosure scenarios while accessing graph databases in a cloud or in-house storage system. First, we propose a method for privately storing data on the cloud by leveraging secure multi-party computation that does not rely on trusting the cloud/processing servers. Second, we introduce a differentially private framework for joint computation by two parties on graph databases in performing efficient and accurate computation of node importance. Finally we contribute more efficient protocol design that overcomes the challenges of computation and communication complexity to compute egocentric betweenness centrality under differential privacy in multi-party settings. Remarkably our protocol incurs negligible utility loss as the number of parties scale and achieves strong privacy protections with practical runtime requirements.

Declaration

This is to certify that

1. the thesis comprises only my original work towards the degree of Doctor of Philosophy except where indicated in the Preface,
2. due acknowledgement has been made in the text to all other material used,
3. the thesis is fewer than 80,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Leyla Roohi

Preface

This thesis has been written at the School of Computing and Information Systems, The University of Melbourne. The core results of the thesis are Chapters 3, 4 and 5. They are based on published proceeding papers and I declare that I am the primary author and have contributed $> 50\%$ in the following papers.

- Leyla Roohi and Vanessa Teague. Privacy-preserving queries over secret-shared graph-structured data. In 2017 IEEE Trustcom/BigDataSE/ICESS, Sydney, Australia, August 1-4, 2017, pages 955–960, 2017.

- Leyla Roohi, Benjamin I. P. Rubinstein, and Vanessa Teague. Differentially private two-party egocentric betweenness centrality. In 2019 IEEE Conference on Computer Communications, INFOCOM 2019, Paris, France, April 29 - May 2, 2019, pages 2233–2241, 2019.

- Leyla Roohi, Benjamin I. P. Rubinstein and Vanessa Teague. Assessing Centrality Without Knowing the Connections. The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20) (under review)

Dedicated to my family

Rezza, Ghazal, Maryam, Jahangir

Acknowledgements

First and foremost, I would like to express my appreciation and gratitude to my supervisors, Associate Professors Vanessa Teague and Benjamin Rubinstein, for their help, support and allowing me to explore my areas of interest. Their trust and patience during my PhD made this journey possible.

I would also like to thank my committee chair, Professor Karin Verspoor for her valuable guidance during my PhD program. Furthermore I thank CIS administrators, Ms. Rhonda Smithies and Ms. Julie Ireland, the university postgraduate and support team and all my friends in CIS that I would not have completed this journey without their love and support.

Last but not least, I would like to sincerely thank my husband and daughter, Rezza and Ghazal. There are no words to explain their kindness. Especially I am grateful to my husband for selflessly providing me with his love and emotional support.

Contents

List of Figures **xv**

List of Tables **xvii**

1 Introduction **3**
1.1 Motivation of this Work 4
1.2 Research Contributions 6

2 Background **9**
2.1 Secure Multi-Party Computation 9
2.1.1 SPDZ 12
2.2 SPDZ Protocol 13
2.2.1 Somewhat Homomorphic Encryption 13
2.2.2 Additive Secret Sharing 14
2.2.3 Sub-Protocols 15
2.2.4 Multiplication 15
2.3 SMC on Graphs 17
2.4 Data Privacy Concerns 18
2.5 Differential Privacy 19
2.5.1 Interactive and Non-Interactive Settings 19
2.5.2 Formal Differential Privacy Definition 20
2.5.3 Generic Mechanisms for Privacy 21
2.5.3.1 Global Sensitivity 21
2.5.4 Laplace Mechanism 22
2.5.5 Exponential Mechanism 22

2.5.6 Privacy Budget and the Compositional Calculus 22
2.6 Differential Privacy on Graphs 23
2.7 Graph Statistics 26
2.7.1 Betweeness Centrality 26
2.7.2 Ego Network Betweenness Centrality 28

3 Privacy-Preserving Queries over Secret-Shared Graph-Structured Data 29
3.1 Introduction 29
3.2 SPDZ vs. SPDZ2 30
3.3 Client-Server Model in SPDZ 32
3.4 Using SPDZ2 Client-Server Model in Graph Secret Sharing (GSS) 34
3.4.1 GSS Protocol 35
3.4.2 Neighbourhood and Second-Hop Query 36
3.5 Implementation 38
3.6 Experimental Results 39
3.7 Concluding Remarks 40

4 Differentially Private Two-Party Egocentric Betweenness Centrality 43
4.1 Introduction 44
4.2 Two-Party Protocol 45
4.3 Warm-Up: A Non-Private Protocol 47
4.4 A Privacy-Preserving Protocol 49
4.4.1 Forward Message 49
4.4.1.1 Symmetric Set Difference 50
4.4.1.2 Equi-Quality Responses 51
4.4.1.3 Two-Stage Sampling 52
4.4.1.4 Linear-Complexity Sampling 55
4.4.1.5 Quality Function Global Sensitivity 57
4.4.2 Backward Message 58
4.4.2.1 Privately Counting Paths 59
4.4.2.2 Private Partial EBC 60
4.4.3 PRIVATEEBC: Putting it All Together 61
4.5 Experiments 65

4.6 Results . . . 65
4.7 Concluding Remarks . . . 69

5 Differentially Private Multi-Party Egocentric Betweenness Centrality 71
5.1 Introduction . . . 71
5.2 Another View of the Two-Party Exponential Mechanism . . . 73
5.3 Problem Statement . . . 76
5.4 Non-Private Multi-Party Protocol . . . 77
5.4.1 Privacy Disclosure . . . 78
5.5 Multi-Party Private EBC . . . 79
5.5.1 Private Egonetwork Broadcast . . . 79
5.5.2 Private Path Count . . . 81
5.5.3 Private Reciprocate and Sum . . . 82
5.5.4 PRIVATEEBC: Putting it All Together . . . 84
5.6 Utility Bound . . . 84
5.7 Experimental Setup . . . 94
5.8 Results . . . 95
5.9 Concluding Remarks . . . 96

6 Conclusion 97
6.1 Summary of Contributions . . . 97
6.1.1 Privacy-Preserving Queries over Secret-Shared Graph-Structured Data . . . 97
6.1.2 Differentially Private Two-Party Egocentric Betweenness Centrality . . . 98
6.1.3 Differentially Private Multi-Party Egocentric Betweenness Centrality . . . 99
6.2 Discussion . . . 100
6.3 Future Research . . . 100

7 Bibliography 105

List of Figures

2.1 Multi-party computation between 5 parties. 12
2.2 Examples of node 'a' with low and high EBC. 27

3.1 Input supply protocol 33
3.2 Output Delivery protocol 33
3.3 GSS protocol. 36

4.1 EBC two-party computation protocol, comprising one forward and two backward messages. 48
4.2 Average relative error of the 60 random nodes with $\epsilon = 0.1$ to 7, Facebook dataset. 62
4.3 Average relative error of 60 random nodes with $\epsilon = 0.1$ to 7, Enron dataset. 62
4.4 Average relative error of 60 random nodes with $\epsilon = 0.1$ to 7, PGP dataset. 63
4.5 Average relative error of 60 random nodes in three different mechanisms, $\epsilon = 1$, Facebook dataset. 63
4.6 Relative error of 100 random nodes with different degrees for $\epsilon = 1$, Facebook dataset. 64
4.7 Time of computing 20 random nodes with $\epsilon = 0.1$ to 7, Enron dataset. 67
4.8 Time of computing 20 random nodes with $\epsilon = 0.1$ to 7, PGP dataset. 67
4.9 Time of computing 20 random nodes with $\epsilon = 0.1$ to 7, Facebook dataset. 68

5.1 Ego network of node 'a' spanning between three parties. 75
5.2 EBC multi-party computation protocol for party α, comprising three messages per party. Visualised for three parties. 78
5.3 Median relative error of the 60 random nodes with ϵ =0.1 to 7, Facebook, Enron and PGP dataset, for three parties. 88
5.4 Median Relative error of 120 nodes with $\epsilon = 1$, for different number of parties for PGP. 89
5.5 Time of computing 60 random nodes with ϵ =0.1 to 7, Facebook data , for three parties. 90
5.6 Time of computing 60 random nodes with ϵ =0.1 to 7, Enron dataset , for three parties. 90
5.7 Time of computing 60 random nodes with ϵ =0.1 to 7, PGP dataset, for three parties. 91
5.8 Relative error of 60 nodes with different degrees for ϵ =1, Facebook dataset. 91
5.9 Relative error of 60 nodes with different degrees for ϵ =1, degree, Enron dataset. 92
5.10 Relative error of 60 nodes with different degrees for ϵ =1, PGP dataset. 93

List of Tables

2.1 Comparison of different SMC Frameworks. 12
2.2 Taxonomy of differential privacy methods on graphs 25

3.1 Neighbourhood and Second-hop Query Computation Time for Dataset of 64 Nodes 40

4.1 Glossary of symbols used in this chapter. 46

5.1 Glossary of symbols used in this Chapter. 74
5.2 Dataset Characteristics 94

List of Algorithms

2.1 RESHARE 16
2.2 PBRACKET 16
2.3 PANGLE 17
3.1 SPDZ MAC Checking 31
3.2 SPDZ2 MAC Checking 31
3.3 Neighbourhood Query 38
3.4 Second-hop Query 39
4.1 FORWARDMESSAGE Two-Stage Sampler 55
4.2 INVERSETRANSFORMSAMPLER 55
4.3 PICKANDFLIPSAMPLER 56
4.4 BACKWARDMESSAGE 58
4.5 PRIVATEEBC 59
5.1 PRIVATEPATHCOUNT 81
5.2 PRIVATERECIPROCATEANDSUM 82
5.3 PRIVATEEBC in Multi-Party 83

Chapter 1

Introduction

The significant growth of social networks, transport networks, peer to peer file sharing and telecommunication networks has created large and complex graphs. These graphs attract many researchers in different application domains such as marketing, psychology, energy industry and many more. Research in these domains has revealed interesting aspects of graph-structured data and delivered new and effective ways of analysing, querying and mining them.

Although much useful information can be extracted from graphs, privacy disclosure of unintentionally too much information about graphs can occur even with graph anonymisation techniques, in three main categories of leaked information: edge, node and content. De-identification or the process of anonymising datasets before sharing them has been used in many paradigms in order to hide the identity of the entities associated to the node, and also relations between the entities [9,49,108]. However, there are numerous examples of re-identification scenarios [30,51,72,90], such as revealing the name and email address of 25.6 million Uber riders and drivers in the USA or the health records belonging to citizens by the Australian Department of Health [22]. Hence, accessing stored graph-structured data or querying such data can reveal both sensitive and valuable information about people or their connections.

1.1 Motivation of this Work

Every day a huge volume of graph data is collected by many companies and organisations. Such data can provide useful information and insights to business and government. However privacy of individuals can be compromised while storing and making queries of data, or sharing derived statistics with untrusted parties; preserving utility and scalability together with protecting privacy are challenges that have gained considerable attention.

In this thesis, three different privacy issues and different adversary models are addressed while accessing graph-structured data. The first privacy challenge is outsourcing of graph data storage to public cloud servers and querying them, where the primary adversary of concern is the untrusted cloud authority. The second and third threats occur when keeping data in house and conducting joint computation with other databases to answer a query. Different adversary models can be considered here due to revealing the query answer to the untrusted querier or any third party. Moreover, the number of the databases or parties attending in the computation can vary and potentially affect privacy, accuracy and computation or communication complexity.

Storing and querying big data securely is a challenge. The expense of storing data means that some companies may be forced to store it remotely. As the NSA's director of targeted access operations put it, "cloud storage is just a fancy name for putting your files on someone else's computer.[1]" Therefore, many companies are interested in keeping their valuable data on cloud storage.The cloud authority can access this data and can learn about the data through observation of query answers.

Our interest here is in cloud storage applications for which encryption alone is insufficient for maintaining stored data and query response privacy. The reason being a single point of failure that inevitably comes from storing a decryption key.

Therefore, the goal is instead of just using cryptographic methods, applying

[1]Rob Joyce, Chief, Tailored Access Operations, National Security Agency, `https://www.youtube.com/watch?v=bDJb8WOJYdA`

a technique that forces the adversary who seeks to gain unauthorised access, to subvert multiple nodes on which data is stored. This is more appropriate for applications in which a powerful adversary may make a few, but so too is arbitrary, queries to highly sensitive data.

Not only is privacy of outsourced data a challenge for companies, but also keeping data in-house while executing joint computation with other companies' databases. This is a fast growing problem due to huge volumes of data generated by companies and also the reluctance of companies to share their valuable data with others. Social licence, commercial competition, or legislative requirements, might prevent sharing data. Database owners might be interested to perform some statistical computations on the graph data such as finding the node importance in the network or even mining the graph to extract node's clusters, therefore they need to share some information about their nodes and edges with other database curators potentially compromising the privacy of the data under their charge.

Although there are many real-world applications of joint computation on graphs, there is limited research on privacy-preserving two-party or multi-party computation techniques on them; particularly with differential privacy as a strong privacy guarantee. Some of these challenges are exacerbated from the complexity of graph structures:

Privacy-preserving techniques in graph-structured data is intuitively more challenging compared to the same techniques on tabular data. Graph- structured data contains significant identifying information that is and it is difficult to measure them [4]. In other words, finding a technique to quantify this information is a challenging question because by perturbing a node or an edge within some privacy-preserving techniques, it is hard to even say how much data is lost. Also, complexity of structures in graph data can cause the difficulty in modelling of the background knowledge and the capability of an attacker. Even some elements of the graph are privatised with some structural metrics, there is a possibility that a release is not private with other metrics. Therefore, it is not clear what is the best privacy model for graphs and how to measure privacy breach. In this thesis we consider threat models for which secure multi-party computation and differential privacy are appropriate.

Moreover, the correlated structure of a graph can affect privacy-preserving techniques such as differential privacy because addition or removal of just a single node or edge can spread through structure spanning the whole graph. This can be a main challenge in such techniques as differential privacy which make extensive use of sensitivity of the query function in order to calibrate the randomisation in the system.For example consider releasing the diameter of a graph while protecting edge privacy. Removal of just one edge could significantly affect this global property.

Apart from structural barriers in graph techniques, multi-party scenarios and 'big data' both are additional challenges for privacy-preserving techniques since they add more computation and communication complexity to algorithms even without privacy considerations.

1.2 Research Contributions

Each chapter of this thesis addresses a particular aspect of the previously described goals. Here we give a short overview of each of them.

In Chapter 2, we present a comprehensive overview of secure multi-party computation (SMC) and differential privacy (DP) techniques with a focus on graph-structured data to provide enough background of knowledge for understanding the main contributions of the thesis. We also provide related theoretical explanation of the methods and techniques and give a comparative analysis in this chapter.

In Chapter 3, we investigate the use of the SPDZ multi- party computation platform to facilitate secure cloud storage of graph-structured data such as telecommunications metadata. We report on an implementation of a simple scheme for answering adjacency, nearest- neighbour and second-hop queries. Our solution hides the data, the query and the answer from the cloud servers unless they all collude to recover them. We use secret sharing to support a trust model without a single point of failure, so that an adversary who seeks to gain unauthorised access must subvert multiple nodes on which data is stored. This is slower than encrypted storage, but more appropriate for applications in

which a powerful adversary may make a few, but not arbitrary, queries to highly sensitive data. The protocol is examined on a graph of 62 nodes for two queries: neighbourhood query and second-hop query.

In Chapter 4, we propose a novel protocol for computing the egocentric betweenness centrality (EBC) of a node when relevant edge information is spread between two mutually distrusting parties such as two telecommunications providers. While each node belongs to one network or the other, its ego network (the subgraph induced by its neighbourhood) might include edges unknown to its network provider. We develop a protocol of differentially-private mechanisms [35] to hide each network's internal edge structure from the other network and contribute a new two-stage stratified sampler for exponential improvement to time and space efficiency. Empirical results on several open graph datasets demonstrate practical relative error rates, such as 16% error on a Facebook dataset, while delivering strong privacy guarantees.

In Chapter 5, the idea of joint computation of EBC between two-parties is generalised and extended to multi-party scenario. As modern communication networks span multiple providers, we show for the first time how multiple mutually-distrusting parties can successfully compute node EBC while revealing only differentially private information about their internal network connections. A theoretical utility analysis upper bounds a primary source of private EBC error—private release of ego networks—with high probability. Empirical results further demonstrate practical applicability with a low 1.07 relative error achievable at strong privacy budget $\epsilon = 0.1$ on a Facebook graph, and insignificant performance degradation as the number of network provider parties grows.

Chapter 2

Background

Privacy-preserving computation techniques with distributed settings have become on-demand methods in today's modern world. Every day a huge amount of data is collected by different companies who need to extract information from data or answer queries in a collaborative manner. In order to avoid unwanted privacy disclosure to third parties, such organisations need to apply privacy-preserving techniques to protect their data, unfortunately there are no single turn-key solutions to maintaining privacy. Fortunately there are many available primitives for privacy-preserving data analysis.

In this chapter, the background for this study and definitions of two privacy-preserving frameworks are discussed with a main focus on graph structured data: secure multi-party computation and differential privacy. For the interested, more details on secure multi-party computation and differential privacy can be found in the books of Cramer *et al.* [20] and Dwork *et al.* [36] respectively.

2.1 Secure Multi-Party Computation

Secure multi-party computation (SMC) is a method in which different parties can compute an agreed function on their data and reveal nothing about their individual inputs except what is implied by the function's output. An example of an SMC application is a group of millionaires wanting to compute how much money they own collectively without revealing the amount of money they each

individually have to one another or other parties (see Figure 2.1).

SMC must defend against a subset of players who deviate from the protocol with the aim of distorting the outcome or learning others' secrets. These are definitions of active and passive adversary models respectively. This method as a computation platform requires not only security against these specific attacks, but enough efficiency in terms of round and communication complexity, to be practically applicable. Round complexity is the maximum number of rounds necessary to execute the protocol. Communication complexity is the maximum number of bits transmitted during the execution of the computation.

Theoretical study on SMC started in 1982 with pioneering work of Andrew Yao [103] and since then there have been numerous foundational results in the area [3, 7, 18, 44, 45, 89]. The literature shows that this approach can solve distributed computing problems while guarantying privacy and security of the inputs. However, the goal of making SMC an efficient and practical platform to use in real world scenarios emerged 10-15 years ago, by some interesting platforms such as SEPIA [14], VIFF [24], SPDZ [28] and SPDZ2 [25]. In early work on SMC the main focus was on a passive adversary model—known as the honest but curious model. This model can be applied to two-party [65] as well as multi-party computation [24]. SEPIA (Security through Private Information Aggregation) [14] is an SMC implementation based on this security model. It is based on Shamir secret sharing [89] and can compute 82,730 multiplication/sec for five players with the use of multi-threaded programing in Java. Recently however, practical interest has arisen on protocols with active security in which adversaries are corrupted and try to eventually change the result. VIFF (Virtual Ideal Functionality Framework) [24] is another implemented SMC protocol which is based on Shamir secret sharing that follows the security model of active adversaries. This framework allows the corruption at a rate less than n/3 and can compute 326 64-bit multiplication/sec for five players, which is significantly less than the number of multiplications/sec in SEPIA. This demonstrates the cost of the higher security model: efficiency can deteriorate, especially for scenarios which allow more than half of the players to be corrupted.

In general, SMC protocols can tolerate some dishonest participants. In other words, if t out of n players (n is the number of players and t is the number of

corrupted players) deviate from the protocol, the privacy of inputs and accuracy of outputs can still be guaranteed.

However, SMC protocols can be very inefficient, particularly for multiplication. A practical breakthrough in SPDZ [28] is the pre-processing model, which allows much of the computational work to be done in advance. The SPDZ protocol is based on additive secret sharing that is secure against active and adaptive corruption of $n-1$ players. It consists of offline and online phases while most of the computation load is shifted to the offline phase. 20000 64-bit multiplications per second can be performed for three players by this protocol while pre-processing takes 13ms to prepare one multiplication.

SPDZ2 [25] makes a significant improvement to the offline phase. SPDZ2 generates square pairs and shared bits in the offline phase that leads to faster performance of specific operations in the online phase. It also introduces a key generation protocol for generating the somewhat homomorphic encryption (SHE) secret shared key. The SPDZ2 protocol was examined with three players in a sequential single thread and 50 in parallel, single thread and four threads and resulted in 4700, 98000 and 260000 64-bit multiplication /sec respectively. The results show the performance of the online phase is comparable to passively secure protocols based on Shamir secret sharing.

A major improvement of the SPDZ2 protocol is the authentication part, which solved a problem of the original SPDZ requiring the MAC key to be revealed at the end. Players need to know the global key α to check the correctness of the publicly opened values in original SPDZ, while in the new protocol, the players do not need the global key to authenticate so the MAC checking can be done before the end of the protocol.

The big advantage of this protocol is the pre-processed data MACed under the same key can still be used after a MAC check. This is very useful for the client-server approach that needs to compute lots of functions on the same shared data, without changing the shares after each computation. Further in Section 2.1.1, we explain some more details of SPDZ.

Table 2.1: Comparison of different SMC Frameworks.

	SEPIA	VIFF	SPDZ	SPDZ2
Technique	Shamir.sh	Shamir.sh	Additive.sh	Additive.sh
Platform	Java	Python	Python,C++	Python,C++
Multiply/S	82730	326	20000	98000
Adversary model	Passive	Active	Active	Active

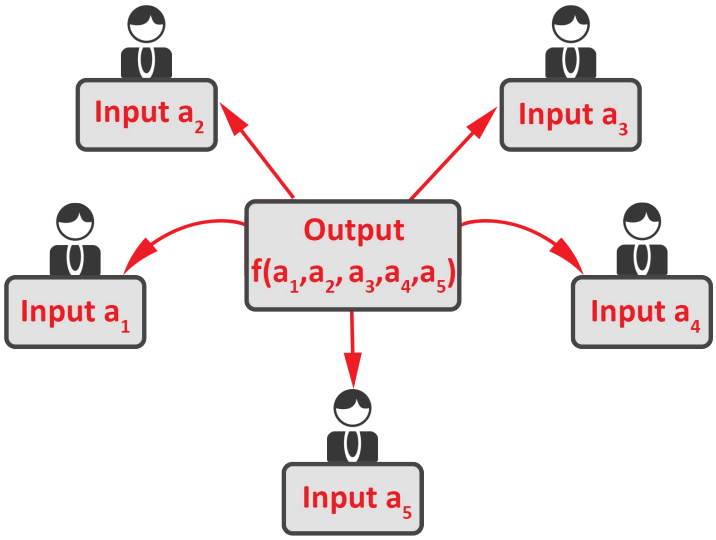


Figure 2.1: Multi-party computation between 5 parties.

2.1.1 SPDZ

SPDZ is a secure multi-party computation (SMC) technique in which two or more parties wish to compute a given function $f(x_1, x_2, ...x_n)$ where data x_i is known by party i. In this technique the honest parties should receive the result of the computation; even if a subset of parties are corrupted. This subset is also called a threshold in this technique and depends on the method of SMC. In the SPDZ technique the threshold is $n-1$; it means if $n-1$ parties are controlled by the adversaries the honest parties get the correct answer and this is the only new information that is released. In the case that more than half of parties are

corrupted, it is very hard to maintain the unconditional security [15]. However, there are some new techniques that make it feasible practically [8,27,77].

One of these techniques is designing a SMC protocol with offline and online phases. Basically in offline or pre-processing phase, there is a trusted dealer that supplies all the raw materials before the protocol is started. The trusted dealer does not need to know about the function and inputs and can provide raw material to the protocol securely. This allows the online phase to work efficiently. Although SPDZ leverages two-phase protocol, the raw materials are generated as a random shares by players in the pre-processing phase and there is no central party to supply the random materials. SPDZ uses a two-phase protocol as well in order to provide an unconditional security. Pre-processing phase in SPDZ allows the protocol to perform an optimal online phase that has better performance in terms of computation complexity compared to the previous techniques.

2.2 SPDZ Protocol

As mentioned in the previous section, the SPDZ protocol consists of offline and online phases. In the offline phase some shared random values are generated and in the online phase the actual computation is performed. The protocol leverages additive secret sharing and Somewhat homomorphic encryption (SHE) in order to generate the shares of the secret values and assign them to the players. In Section 2.2.2, there are details of how these two tools work together to secretly share the values.

2.2.1 Somewhat Homomorphic Encryption

In the pre-processing phase, SPDZ uses somewhat homomorphic encryption (SHE) to help players privately compute the share of secret inputs and the related message authentication code (MAC).

The advantage of using SHE compared to the previous model that uses fully homomorphic encryption (FHE) is that it does not need bootstrapping. However, SHE supports a limited number of homomorphic operations. Additions are

basically free for known SHE, but multiplications can be really costly. FHE definition for arbitrary number of V where V is the depth of the circuits is:

$$\text{ENC}(m_1) \boxplus \ldots \boxplus \text{ENC}(m_V) = \text{ENC}(m_1 + \ldots + m_V)$$

$$\text{ENC}(m_1) \boxtimes \ldots \boxtimes \text{ENC}(m_V) = \text{ENC}(m_1 * \ldots * m_V)$$

In other words, the main difference between FHE and SHE is the depth of multiplication circuits. The depth of multiplication circuits in FHE is an arbitrary number while SHE has a limited depth of multiplication circuits.

In SPDZ the SHE single instruction, multiple data (SIMD) approach of [93] is used to handle many values in parallel in a single ciphertext. It leads to an efficient pre-processing phase and allows computation of circuits of multiplicative in depth one on ciphertexts.

In Algorithm 2.1, 2.2 and 2.3, the SHE addition and multiplication of encrypted inputs are denoted by $\boxplus$ and $\boxtimes$. Addition is allowed on V ciphertexts where V is an arbitrary value.

$$\text{ENC}(m_1) \boxplus \ldots \boxplus \text{ENC}(m_V) = \text{ENC}(m_1 + \ldots + m_V)$$

$$\text{ENC}(m_1) \boxtimes \text{ENC}(m_2) = \text{ENC}(m_1 * m_2)$$

2.2.2 Additive Secret Sharing

In SPDZ protocol, each player P_i keeps share x_i of secret $x = x_1 + x_2 + \ldots + x_n$, where $x \in \mathbb{F}_p$ for some large prime p, and x_is are randomly chosen in such a way that if $n-1$ of the players collude, x remains unknown. Each player P_i also holds $\gamma(x_i)$-MAC- for authenticating the value of x, where $\gamma(x_i)$ is the additive secret sharing of $\gamma(x) = \alpha * x$, that is $\gamma(x) = \gamma(x_1) + \gamma(x_2) + \ldots + \gamma(x_n)$ and α is the global MAC key. Therefore, we say that player P_i holds tuple $(x_i, \gamma(x_i))$ which is denoted as $< x >$ when it is used in computation.

More specifically, let $[[r]]$ denote an encrypted version of r. To share a value x, each player P_i takes an available pair of $< r >$ and $[[r]]$ which were generated in pre-processing phase and broadcasts $\epsilon = x + r$ to the other players, then

players compute $< x >=< r > +\epsilon$ by letting a particular player add ϵ to its share. In other words, one player P_i holds the share $x + r - r_i$ and other players hold just the $-r_i$ where $r = r_1 + r_2 + \ldots + r_n$. For generating $\gamma(x_i)$ in tuple $(x_i, \gamma(x_i))$, the MAC key that was generated in the offline phase is used. α is not known by players but the encryption $[[\alpha]]$ and shared $< \alpha >$ is. SHE (Somewhat Homomorphic Encryption) helps each player privately compute $x * \alpha$ without knowing x and α, and generate the shares of the MAC. The checking of all the shares and also global MAC key α is postponed to the end of protocol. α can be checked by any player P_i that holds $[[\alpha]]$ when any player p_i uses sub-protocol 2.3 to share $\alpha.\beta_i$ as $\gamma(\alpha)_i^j$. Therefore, at the end of protocol player p_i can check the integrity of the α by receiving the shares of α and $\gamma(\alpha)_i^j$ and checking $\sum_j \gamma(\alpha)_i^j = \alpha.\beta_i$,

2.2.3 Sub-Protocols

In the pre-processing phase, SPDZ uses three sub-protocols:

RESHARE, PBRACKET, PANGLE. These protocols generate additive secret shares of plain text x and their related MAC.

RESHARE sub-protocol 2.1 additively secret shares a plain text x; the input of this protocol is encrypted version of the plain text x as a public value, and the output is a share x_i of x to each player i, where there are n players in the protocol.

PBRACKET sub-protocol 2.2 re-shares the share of values with leveraging key β_i. This sub-protocol can be used for integrity checking of the global MAC key α.

PANGLE sub-protocol 2.3 shares any values in the SPDZ protocol by using global MAC key α in order to generate the shares of the MAC.

2.2.4 Multiplication

The number of multiplications in an arithmetic circuit is the dominant cost in SMC protocols. Since SPDZ protocol uses the classic circuit randomisation technique of Beaver in the pre-processing phase to generate the triples, multiplication can be done by evaluating many small circuits of multiplicative with depth 1 in

Algorithm 2.1 RESHARE

Input: $e_x \longleftarrow \text{ENC}(x)$; e_x is a public cipher text

`//Each player` P_i `does the following`

1: $r_i \sim \text{Unif}\left(\mathbb{F}_{p^k}\right)^s$

2: Broadcasts $e_{r_i} \longleftarrow \text{ENC}(r_i)$

`//Players compute` e_r `and` e_{x+r} `by SHE`

3: $e_r \longleftarrow e_{r_1} \boxplus \ldots \boxplus e_{r_n}$

4: $e_{r+x} \longleftarrow e_x \boxplus e_r$

`//Players decrypt` $e_r + x$ `and obtain` $x + r$

5: P_1: $x_1 \longleftarrow x + r - r_1$

6: $P_i (i \neq 1)$ $x_i \longleftarrow -r_i$

7: **return** $x_1 \ldots x_n$

Algorithm 2.2 PBRACKET

Input: $e_t \longleftarrow \text{ENC}(t)$; e_t is a public cipher text, shares of the t; $t - 1, t_2, \ldots t_n$; $e_{\beta_i} \longleftarrow \text{ENC}(\beta_i)$ is generated in initialisation process by every player p_i

`//This protocol generates [[t]]`

1: **for** $i \in \{1, 2, \ldots n\}$ **do**

2: P_i generates: $e_{\gamma_i} \longleftarrow e_t \boxtimes e_{\beta_i}$ `//Each player` p_j `gets a share of` γ_j^i `of` $t.\beta_i$

3: $(\gamma_i^1, \gamma_i^2, \ldots, \gamma_i^n) \longleftarrow \text{RESHARE}(e_{\gamma_i})$

4: **end for**

5: **return** $[[t]] = (0, t_1, t_2, \ldots, t_n, (\beta_i, \gamma_i^1, \gamma_i^2, \ldots, \gamma_i^n)_{i=1,2,\ldots,n})$

parallel and it can increase the rate of multiplication/s to 20000 in SPDZ protocol. In order to multiply $< x >$ and $< y >$ the parties do the following:

In the pre-processing phase, they take two available triples $< a >, < b >, < c >$ and $< d >, < e >, < f >$ and check if $a.b = c$. The first triple is used for multiplication, which is based on a technique by Beaver [5]. The second triple is sacrificed to check the first triple and ensures that the first triple is not manipulated by the adversary.

In the online phase, for computing $< x > . < y >$, parties use the first triple $< a >, < b >, < c >$ which were checked in the pre-processing phase and apply Beaver technique [5] as following:

First, parties compute $< x > - < a >= \epsilon$, and $< y > - < b >= \delta$; then

Algorithm 2.3 PANGLE

Input: $e_t \longleftarrow \text{ENC}(t); e_t$ is a public cipher text, shares of the t; $t-1, t_2, \ldots t_n$; $e_\alpha \longleftarrow \text{ENC}(\alpha)$ is generated in initialisation process by every player p_i
`//This protocol generates <t>`
1: P_i generates: $e_{t.\alpha} \longleftarrow e_t \boxtimes e_\alpha$
2: $(\gamma_1, \gamma_2, \ldots, \gamma_n) \longleftarrow \text{RESHARE}(e_{t.\alpha})$
`//Each player` p_i `gets a share of` γ_i `of` $t.\alpha$
3: **return** $< t >= (0, t_1, t_2, \ldots, t_n, (\gamma_1, \gamma_2, \ldots, \gamma_n)$

$< x > . < y >$ results in:

$$< x > . < y >=< c > +\epsilon < b > +\delta < a > +\epsilon\delta$$

.

2.3 SMC on Graphs

Since emerging SMC in 1980, in the seminal work by Yao [103] as a two-party protocol, numerous SMC protocols have been proposed in different contexts such as, mining [97], voting [21], face recognition [87], self-organising map [92] and decision tree [53]; however, very little research has been done on SMC in graphs due to complexity of graph data.

Do *et al.* [31] proposed a two-party and multi-party protocols to count the triangles in the graph privately, the protocol uses homomorphic encryption and privacy model is passive; honest but curious. Brickell and Shmatikov [13] also presented a privacy-preserving two-party scenario by leveraging Yao's method [103] and homomorphic encryption. They computed the all-pair shortest path, single source shortest-path and minimum spanning tree in the honest but curious privacy model. Another SMC based on two-party protocol and honest but curious privacy model presented by Failla and Pierluigi [39]; they used A^* algorithm [47] to search the shortest path on an encrypted graph. A^* algorithm is based on the application of heuristic function [50].

A multi-party setting introduced by Aley *et al.* [2] to compute the shortest path and maximum flow. The algorithm works in presence of an honest majority

for active or malicious, and passive adversaries. It is based on secure arithmetic black-box functionality, which was proposed by Damgard and Nielsen [26]. This method uses addition, multiplication and comparison functions from arithmetic black-box functionality.

The proposed protocol in Chapter 3 uses arithmetic functions of SPDZ2 protocol [25] as black-boxes. Many other work on two-party and multi-party settings also called the black-box functionality, such as Fairplay [6], Sharemind [12], VIFF [24] and SEPIA [14].

2.4 Data Privacy Concerns

During the last decade a huge amount of data has been generated and collected by companies, organisations and governments, and the pace of collection is significantly increasing, due to rapid growth of technology. The data collectors are often interested to publish data for further analysis. As a result the privacy of the collected data can be compromised in two key ways during this procedure: first, when the collector receives the data from different resources and second, when the collector sends the aggregate information extracted from collected data to the users. In the first scenario the adversary is the data collector, while the user is the main adversary in the second; in both scenarios private and sensitive information about individuals can be revealed. In both stages the naive approach of data anonymisation can be applied to the data in order to attempt to provide data privacy. Popular techniques in this category are: K-anonymity [60,88,95,96], l-diversity [64,100,102], t-closeness [61,62,85] and δ-presence [74,75]; however, these methods are vulnerable and can be compromised by linkage attack to background information; background information could come from databases or could come from a privatise database [19,54,101].

The prevalence of privacy violations from released "anonymised" data shows at these synthetic technique do not work: they are synthetic in that they do not guarantee any particular privacy property. In 1998, the de-anonymisation demonstration on a published medical dataset [88,96] that showed how the link between fields of two databases can link the privacy information of individuals

in anonymised database with their identities. Other examples of linkage attack include Netflix dataset de-anonymisation [72] and re-identification of individuals in an Australian Department of Health dataset [22]. Moreover, research on different privacy models based on anonymisation shows different possible attacks on databases; *i.e.*, minimality attack [19, 54, 101] in K-anonymity techniques or attacks based on composition [42], and foreground knowledge [99] which provide yet more evidence of the insufficienncy of privacy guarantees provided by techniques.

All of these real world scenarios and examples show that anonymisation techniques do not suffice to preserve the privacy of data and more reliable techniques are required to protect privacy. Differential privacy has emerged as a robust solution to overcome these problems by introducing a provable privacy guarantee against background knowledge and against harm not only caused by re-identification.

2.5 Differential Privacy

Differential privacy (DP) provides an opportunity for the data collector to collect and share aggregate data while maintaining the privacy of individuals. According to the privacy model in differential privacy, the privacy of an individual in the database is preserved even if the adversary knows all information about the database from background knowledge such as auxiliary data, except the information about that particular individual in the database.

Differential privacy guarantees for the data collector that even with knowledge of $t-1$ records of a database of t records that the adversary can not learn about any one unknown record. In the next Section 2.5.2, the formal definition of differential privacy is presented.

2.5.1 Interactive and Non-Interactive Settings

In publishing aggregate information in differential privacy, there are two main settings: the interactive and non-interactive settings. In the interactive setting, data collector receives queries one by one and randomises the answer before

sending it to the user. However, in the non-interactive setting a single release is made which is not per se a response to a specific query. This could be a release a machine learned model trained on sensitive data for example [86].

2.5.2 Formal Differential Privacy Definition

Differential privacy is proposed to quantify the indistinguishability of input databases when observing the output of data analysis [35]. With careful selection of which databases are to be indistinguishable—through the so-called neighbouring relation—the protective semantics of differential privacy may be controlled.

Formally, two databases $D, D' \in \mathcal{D}^n$ are termed *neighbouring* (denoted $D \sim D'$) if there exists exactly one $i \in [n]$ such that $D_i \neq D'_i$ and $D_j = D'_j$ for all $j \in [n] \backslash \{i\}$. For example over scalar binary data $\mathcal{D} = \{0, 1\}$, neighbouring databases D, D' are such that $\|D - D'\|_1 = 1$.

Definition 1. *For $\epsilon > 0$, a randomised algorithm on databases or* mechanism $\mathcal{A}$ *is said to preserve* ϵ-differential privacy *if for any two neighbouring databases D, D', and for any measurable set $R \subseteq \text{Range}(\mathcal{A})$,*

$$\Pr\left(\mathcal{A}(D) \in R\right) \leq \quad \exp(\epsilon) \cdot \Pr\left(\mathcal{A}(D') \in R\right) \quad . \tag{2.1}$$

We consider a powerful attacker that might know $\mathcal{A}$ up to its source of randomness, the true input database D up to one record (as in, the attacker knows a ball of databases neighbouring D but not D exactly, has access to releases of $\mathcal{A}$, and has unbounded computational power).

DP guarantees that even then, the attacker cannot significantly distinguish input databases. Using their knowledge they could form this ball of databases, they could form a histogram of observed releases $A(D)$ that approximates $\Pr\left(\mathcal{A}(D) \in R\right)$ but that this histogram also would resemble all the $\Pr\left(\mathcal{A}(D') \in R\right)$ owing to the DP guarantee. And as such as long as the histogram is far enough from $\Pr\left(\mathcal{A}(D) \in R\right)$ due to limiting the number of releases of $\mathcal{A}(D)$ to be linear in the size of D, then the histogram is no closer to the true distribution than the other distributions in the ball. Finally, level of privacy protection can be

controlled by varying epsilon-smaller epsilon means more privacy.

2.5.3 Generic Mechanisms for Privacy

There are two important DP mechanisms used in this thesis: the Laplace mechanism [35] which applies additive noise to numeric vector-valued analyses, and the exponential mechanism [67] which privately optimises a real-valued objective function bivariate in the database and the decision variable which need not be numeric. Common to most generic mechanisms, and the Laplace and exponential in particular, is the concept of sensitivity-calibrated randomisation: the more sensitive a target function is to input perturbation, the more randomisation is required to attain a level of differential privacy. Both mechanisms leveraged in this thesis are calibrated via the same measure of sensitivity. Roughly speaking, the sensitivity of any function f reflects how much this function could possibly change while the function input changes between D and D'. In other words, sensitivity is a key parameter controlling how much noise should be added in the processes of answering a query; the more sensitivity the more noise should be added as intuitively differential privacy requires that the randomised mechanism *not* be sensitive to perturbation. In this way, sensitivity-calibrated randomisation smooths out sensitive mechanisms. In the next Section 2.5.3.1, the formal definition of global sensitivity is given.

2.5.3.1 Global Sensitivity

Definition 2. *The* L_1-global sensitivity *of any function* $f : \mathcal{D}^n \to \mathbb{R}^d$ *for any* $d \in \mathbb{N}$, *is defined as*

$$\Delta f \geq \sup_{D,D' \in \mathcal{D}^n, D \sim D'} \|f(D) - f(D')\|_1 \ . \tag{2.2}$$

For functions of additional variables $f : \{0,1\}^n \times \Theta \to \mathbb{R}^d$ *we extend this definition naturally as*

$$\Delta f \geq \sup_{\theta \in \Theta} \sup_{D,D' \in \mathcal{D}^n, D \sim D'} \|f(D,\theta) - f(D',\theta)\|_1 \ . \tag{2.3}$$

2.5.4 Laplace Mechanism

The Laplace mechanism achieves differential privacy for any non-private numeric vector release that is of bounded sensitive. It operates by additive perturbation: noise is added to the non-private analysis result.

Lemma 3. *Consider any Euclidean vector-valued deterministic function $f : \mathcal{D}^n \to \mathbb{R}^d$ for any $d \in \mathbb{N}$, and any scalar $\epsilon > 0$. Given input $D \in \mathcal{D}^n$, the Laplace mechanism releases responses in $\mathbb{R}^d$ distributed as $f(D) + \mathbf{X}$ where $\mathbf{X}$ is d i.i.d. zero-mean Laplace*[1] *r.v.'s with scale $\Delta f / \epsilon$. Then the Laplace mechanism preserves ϵ-differential privacy.*

2.5.5 Exponential Mechanism

The exponential mechanism generalises the Laplace mechanism, and permits private optimisation of any scalar-valued objective function, framed as a quality function. It operates by replacing optimisation by sampling close to optimising solutions.

Lemma 4 ([35]). *Consider any real-valued bivariate* quality function $q : \mathcal{D}^n \times \Theta \to \mathbb{R}$, *which assigns quality score $q(D, \theta)$ to candidate response $\theta \in \Theta$, on input database $D \in \mathcal{D}^n$. The exponential mechanism approximately maximises $q(D, \cdot)$ by releasing randomised response θ with likelihood proportional to* $\exp(q(D, \theta) \cdot \epsilon / (2 \cdot \Delta q))$. *Then the exponential mechanism preserves ϵ-differential privacy.*

2.5.6 Privacy Budget and the Compositional Calculus

In Definition 1, parameter ϵ is privacy budget and shows the privacy level of mechanism M. The smaller privacy budget achieves stronger level of privacy. The compositional calculus shows how the privacy budget changes when the differential privacy mechanisms runs more than once on a database.

In order to build up more complex privacy-preserving computations, it is necessary to be able to quantify the privacy loss of compositions. Fortunately, differential privacy satisfies sequential composition and transformation invariance [35, 58, 68] among other compositions.

[1] The zero-mean scalar Laplace with scale $\lambda > 0$ has PDF $(2\lambda)^{-1} \exp(-|x|/\lambda)$.

Lemma 5 (Sequential composition)**.** *For any sequence of randomised mechanisms $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$, if each $\mathcal{A}_i$ preserves ϵ_i-differential privacy then the compound response on a database D, $(\mathcal{A}_1(D), \ldots, \mathcal{A}_k(D))$, preserves $\left(\sum_{i=1}^{k} \epsilon_i\right)$-differential privacy.*

Lemma 6 (Transformation invariance)**.** *For any mechanism $\mathcal{A}_1$ that is ϵ-differentially-private, and any (possibly randomised) mapping $\mathcal{A}_2$ with domain containing the co-domain of $\mathcal{A}_1$, the randomised mechanism $\mathcal{A} = \mathcal{A}_2 \circ \mathcal{A}_1$ preserves ϵ-differential privacy.*

2.6 Differential Privacy on Graphs

Graph-structured data research and applications have enjoyed significant growth in the past two decades due to rapid evolution in technologies, such as online social networks, sensor networks and telecommunication networks which at their core involve graphs. The graph is represented by a pair $G = (V, E)$ where V represents the set of nodes and E is the set of edges (each edge a set of two nodes in the case of undirected graphs; or an ordered pair of nodes in the case of directed graphs). The node can represent the individual in an online social network, sensor networks or mobile phones networks among other examples, while edges reflect the connection between the entities, such as friendship connections or sent/received signals between two sensors or even a phone call between two individuals in the network.

Although a graph can provide a large amount of useful information to the users, it could be a source of privacy disclosure also. Three privacy disclosures can occur on graph-structured data: edge, node and content disclosure. In edge disclosure the connection between entities is revealed, *e.g.*, who calls whom in a telecommunication network. Node privacy is compromised when the identity of an individual or entity who is associated with a node is disclosed and content privacy breach in graphs is due to revealing private information which is related to a sensitive node. Re-identification is a potential privacy threat to nodes and edges of a sensitive graph.

K-anonymity [96] represents a major, early attempt at preventing re-identification in databases, but it has been proven to be insufficient [1]. Some anonymi-

sation methods are proposed particularly for preventing node and edge re-identification threat in graphs such as, *K*-candidate [49], *K*-degree anonymity [63] and *K*-neighbourhood anonymity [107] for node privacy, and privacy preserving link analysis [32] and random perturbation [104] for edge privacy.

Differential privacy for graph processing was first introduced in [78] and was followed up by further work [29,70,91,106]. Two main privacy models exist when publishing graph-based information under differential privacy; node [29] and edge differential privacy [78]. Hay *et al.* [48] introduced an algorithm for publishing degree distributions under node and edge privacy, implicitly permitting private *K*-star counting as well. Projection-based techniques have been proposed to answer degree distribution queries under node differential privacy [57,83].

Other statistics have been approximated under differential privacy such as frequent patterns of given sub graphs [10,55,106]. Bhaskar *et al.* [10] used the exponential mechanism to publish the (approximately) most frequent patterns with high probability, and the Laplace mechanism to release the noisy frequency of maximising patterns. Karawa *et al.* [55] proposed a differentially private algorithm to output answers to sub-graph counting queries for *K*-star, *K*-triangle sub graphs while using local sensitivity [78] to overcome high global sensitivity in sub-graph counting queries. An approach to finding arbitrary frequent patterns was proposed by Shen & Yu [91]. They utilise the exponential mechanism and Markov chain Monte Carlo sampling to output frequent patterns on graph datasets.

Finding node clusters in a single graph under differential privacy was first proposed by [70] and followed by [76]. These techniques try to find the group of nodes sharing many links with other nodes in the same group, but relatively few outside the group. They maintain the privacy of the output clusters under node or edge differential privacy. Wang *et al.* [98] proposed a divide and conquer approach to enforce edge differential privacy to graph structured data. They decomposed the target computation f to the several less complex unit computations which are connected by less complex units and then added Laplace noise to the output of each sub-query or basic unit computation by distributed privacy budget and smooth sensitivity. They applied the proposed technique on different

Table 2.2: Taxonomy of differential privacy methods on graphs

Author	Node/Edge DP	Interactive/Non-Interactive	Query Type
Hay *et al.* [48]	Node	Interactive	Degree distribution
Kasiviswanathan *et al.* [57]	Node	Interactive	Degree distribution
Blocki *et al.* [11]	Node	Interactive	General
Day *et al.* [29]	Node	Interactive	Degree Distribution
Raskhodnikova and Smith [83]	Node	Interactive	Degree distribution
Chen *et al.* [17]	Node	Non-Interactive	Sub-graph counting
Karwa *et al.* [55]	Edge	Interactive	K-triangle, K-star
Rastogi *et al.* [84]	Edge	Interactive	General Sub-graph counting
Nisim *et al.* [78]	Edge	Interactive	Sub-graph counting
Hay *et al.* [48]	Edge	Interactive	Frequent Pattern, K-star
Zhang *et al.* [106]	Edge	Interactive	Sub-graph
Zhu *et al.* [109]	Edge	Non-Interactive	General
Shen and Yu [91]	Node	Non-Interactive	Frequent Pattern
Mulle *et al.* [70]	Node	Interactive	Node-clustering
Nguyen *et al.* [76]	Edge	Non-Interactive	Node clustering
Wang *et al.* [98]	Edge	Interactive	clustering coefficient

synthetic and real graphs to compute clustering coefficient.

Table 2.2 shows the taxonomy of differential privacy methods on graphs. The proposed methods in Chapter 4 and 5 differ from previous studies reviewed in this section, in two key ways. First we focus on the problem of node influence, through the study of ego betweenness centrality. This particular task poses significant technical challenges, made efficient in those chapters by adopting two-stage stratified and accept-reject sampling. Second we consider a core graph processing task in a distributed two-party and multi-party settings. While most existing work on graph mining under differential privacy can adopt a model of trusted computation, there is some work on privacy for distributed systems [16, 34], these are based on distributed queries that are decomposed into sub-queries, each answered per database. The methods in this thesis require untrusting parties to cooperate on computation without revealing one another's privacy-sensitive data.

Node differential privacy guarantees indistinguishability of answer which of queries on neighbouring graphs which just differ on one node. This is ac-

complished by ensuring the response distribution of running a differentially private mechanism on two such graphs undergoes only bounded changes. The sensitivity of of graph analyses in the context of node differential privacy can be high as a node could have common edges with all other nodes in the graph and so node removal could affect global graph structure. Edge differential privacy ensures that for any two neighbouring graphs that just differ in one edge, the response distribution on the first graph is very close to the response distribution on the second graph. For some analyses such as degree distributions, this doesn't have large sensitivity [48]. But for analyses involving connectedness or shortest paths, removal of an edge could greatly affect sensitivity.

2.7 Graph Statistics

Several sub-graph queries have presented under edge and node differential privacy such as, *K*-triangle, *K*-star counting and degree distribution, Table 2.2. Other useful statistics can be measured on graphs such as betweeness centrality [40]. Betweenness centrality is a measure of centrality in a graph based on shortest paths that has a wide range of applications from biology to transport and telecommunication. For instance, in a telecommunications network, a node with higher betweenness centrality would have more control over the network, because more information will pass through that node. Another example is preventing the spread of fake news in social networks by controlling the most central node.

2.7.1 Betweeness Centrality

Measures of centrality enable identification of the most important nodes within a network, with betweenness centrality based on shortest paths. Popular among these measures is betweenness centrality, which represents the extent to which nodes stand between each other. Betweenness centrality was devised as a general measure of centrality: it applies to a wide range of problems in complex networks, including analysis of social networks [43], biological regulatory networks [105], transport and scientific cooperation [82]. Although earlier authors

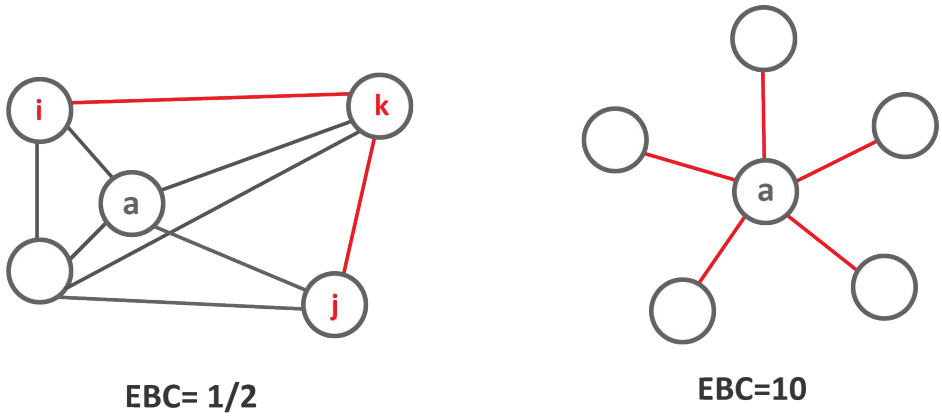


Figure 2.2: Examples of node 'a' with low and high EBC.

have intuitively described centrality as based on betweenness, Freeman [40] gave the first formal definition of betweenness centrality.

Definition 7. *The* betweenness centrality (BC) *of node a in simple undirected graph (V, E) is defined as*

$$\mathrm{BC}(a) \quad = \quad \sum_{\{i,j\} \subseteq V \setminus \{a\}} \frac{\delta_{ij}(a)}{\delta_{ij}} \ ,$$

where δ_{ij} denotes the total number of shortest paths between the nodes i and j, and $\delta_{ij}(a)$ is the number of such shortest paths that also pass through node a.

For an undirected graph, for each pair of the nodes in the graph there is a path which is minimized in the number of edges. Betweeness centrality of a node is the number of these shortest paths passing through the node.

Unfortunately betweeness centrality is not the most scalable tool for analysing large networks, as computing the measure is dependant on evaluating all shortest paths between a node and all other nodes in the network, as well as between all nodes. This costs $O(|V| \cdot |E|)$.

2.7.2 Ego Network Betweenness Centrality

First proposed by Everett & Borgatti [37] as an approximation to betweenness centrality [40], egocentric betweenness centrality (EBC) has gained recognition in its own right as a natural measure of a node's importance as a network bridge [66]. The EBC of a node a is the sum, for all pairs of neighbours of a that are not directly connected, of the fraction of 2-edge paths between them that pass through a.

EBC is the computation of the centrality for a node while only the neighbours or *ego network* of the node is considered for endpoints of shortest paths. The runtime cost of computing the EBC is potentially far less than that of betweenness centrality, only worst-case cubic in the size of the chosen node's ego network. Figure 2.2 shows examples of high EBC and low EBC nodes.

Definition 8. Egocentric betweenness centrality (EBC) *of node a in simple undirected graph (V, E) is defined as*

$$\mathrm{EBC}(a) \quad = \quad \sum_{i,j \in N_a : A_{ij}=0, j>i} \frac{1}{A^2_{ij}} \ ,$$

where $N_a = \{v \in V \mid \{v, a\} \in E\}$ denotes the neighbourhood or ego network *of a, A denotes the $(|N_a| + 1) \times (|N_a| + 1)$ adjacency matrix induced by $N_a \cup \{a\}$ with $A_{ij} = 1$ if $\{i, j\} \in E$ and 0 otherwise; A^2_{ij} denotes the ij-th entry of the matrix square, guaranteed positive for all $i, j \in N_a$ since all such nodes are connected through a.*

Chapter 3

Privacy-Preserving Queries over Secret-Shared Graph-Structured Data

In this chapter[1], we propose graph secret sharing as a secure and flexible approach for making different queries such as neighbourhood and next hop. We apply the SPDZ2 multi-party computation platform [25] to graph structured data for achieving a more flexible approach in processing such queries. We build on secure multi-party computation methods and SPDZ as overviewed in Chapter 2, Section 2.1.1.

3.1 Introduction

We investigate the use of the SPDZ multi-party computation platform to facilitate secure cloud storage and processing of graph-structured data such as telecommunications metadata. We report on an implementation of a simple scheme for answering adjacency, nearest-neighbour and second-hop queries. Our solution hides the data, the query and the answer from the cloud servers unless they all collude to recover them. Our aim is to use secret sharing rather

[1]This chapter is based on the following article:
Leyla Roohi and Vanessa Teague. Privacy-preserving queries over secret-shared graph-structured data. In: 2017 IEEE Trustcom/BigDataSE/ICESS, pages 955–960. IEEE, 2017.

than encryption to allow for a trust model without any single point of failure (such as a decryption key). We begin our investigation with a simple adjacency matrix that uses the SPDZ system to share the graph among multiple nodes and then to compute answers to simple queries without any of the cloud servers learning anything about the data, the query or the answer.

3.2 SPDZ vs. SPDZ2

SPDZ is a secure multi-party computation technique in which all parties take part in generating shared secrets in the pre-processing phase and calculating the function. The idea is that each player holds a share of the secret x that is generated by additive secret sharing. Recall from Chapter 2 that: each player P_i keeps share x_i of secret $x = x_1 + x_2 + \ldots x_n$, where $x \in \mathbb{F}_p$ for some large prime p, and x_is are randomly chosen in such a way that if $n-1$ of the players collude, x remains unknown. Each player P_i also holds $\gamma(x_i)$ for authenticating the value of x, where $\gamma(x_i)$ is the additive secret shared of $\gamma(x) = \alpha * x$, that is $\gamma(x) = \gamma(x_1) + \gamma(x_2) + \ldots + \gamma(x_n)$ and α is the global MAC key. Therefore, we say that player P_i holds tuple $(x_i, \gamma(x_i))$ which is denoted as $< x >$ when it is used in computation.

In SPDZ, MAC checking is done in two phases for output value y of any function f, where players hold shares $< y_i >$: first players check that all the values which are publicly opened so far, are correct. Second, players open commitments on y_i and $\gamma(y_i)$ and check whether $\alpha(\sum(y_i) + \delta) = \sum(\gamma(y_i))$, where y_i is the share of the output value y and $\gamma(y_i)$ represents the MAC of y_i. Both phases follow Algorithm 3.1, where it is assumed that all players have a public set of opened values $\{a_1, \ldots, a_t\}$. Algorithm 3.1 provides a sketch of SPDZ MAC checking where the commitments to the intermediate values are not considered.

SPDZ2 is the updated version of SPDZ which is improved in the offline and the online phase by adding the key generation technique, producing squaring tuples and shared bits in offline phase. The main change in SPDZ2 which has affected both online and offline phase is MAC checking protocol. Recall from

Algorithm 3.1 SPDZ MAC Checking

Input: Each player has input α_i and $\gamma(a_j)_i$ for $j = 1, \ldots, t$ and random vector l_j; all players have a public set of opened values $\{a_1, \ldots, a_t\}$.

`//player` P_i `computes:`

1: $a \longleftarrow \sum_{j=1}^{t} l_j.a_j$

`//Player` P_i `computes and broadcasts:`

2: $\gamma_i \longleftarrow \sum_{j=1}^{t} l_j.\gamma(a_j)_i$

`//MAC key` α `is opened`

`//Each player` p_i `checks if:`

3: $\alpha.a == \sum_i \gamma_i$

`//if not the protocol aborts`

Algorithm 3.2 SPDZ2 MAC Checking

Input: Each player has input α_i and $\gamma(a_j)_i$ for $j = 1, \ldots, t$ and random vector l_j; all players have a public set of opened values $\{a_1, \ldots, a_t\}$.

`//player` P_i `computes:`

1: $a \longleftarrow \sum_{j=1}^{t} l_j.a_j$

2: $\gamma_i \longleftarrow \sum_{j=1}^{t} l_j.\gamma(a_j)_i$

`//Player` p_i `broadcasts:`

3: $\sigma_i \longleftarrow \gamma_i - \alpha.a$

`//Players check if:`

4: $\sigma_i + \ldots + \sigma_n == 0$

`//if not the protocol aborts`

Chapter 2 in the SPDZ protocol the MAC checking only can be done at the end of the protocol because the global MAC key α will be revealed after MAC checking. However, SPDZ2 MAC checking can be done at any time during the offline and the online phase. The reason for this manoeuvre is that: the MAC checking on all partially opened values will not reveal the MAC key, so the first direct impact on the protocol is, pushing the process of triples, squaring tuples and shared bits MAC checking to the offline phase. The idea is that every player P_i computes $\sigma_i := \gamma_i - \alpha_i.a$ and broadcasts the commitment of σ_i to the other players. Every player opens the commitments and checks if $\sigma_1 + \sigma_2 + ... + \sigma_n$ is 0. Algorithm 3.2 shows a sketch of SPDZ2 MAC checking protocol where the commitments to the intermediate values are not considered.

3.3 Client-Server Model in SPDZ

SPDZ as multi-party computation protocol, characteristically needs all parties to take part in all steps of the protocol, including: pre-processing, secret sharing and MAC checking. But, in applications which need a client-server model, we like to make the client separate from other parties. The reason for this is that in many applications we do not know the client who wants to participate in the multi-party computation scenario beforehand and it is also unnecessary and inefficient for clients to collaborate in computationally expensive pre-processing phase. Damgard *et al* [23] proposed a protocol which is based on SPDZ multi-party computation and can meet the client-server model.

In this protocol, the client as a data owner generates the public value e and sends it to the servers, where $e = x - k$. x is the value that client wants to share it with the servers and k is the random value generated by the servers. Servers add shares of the random value to the e and generate $< x_i >$ in the same way as SPDZ protocol. The computation is done in the servers, on the client data and the private data which was being kept by the servers. Figure 3.1 shows the client-server input supply protocol which is based on SPDZ.

The output of the function in this scenario should only be revealed to the client. Logically, sending shares of the output $< y_i >$ to the client seems viable but in this case the servers can easily forge the $< y_i >$. The reason for this forging is back to the SPDZ protocol because the global MAC key α has been revealed to the servers at this phase and servers can forge the $< y_i >$. Therefore, servers use two unused random values $< r >$ and $< u >$ from the pre-processing phase to make the algebraic manipulation detection (AMD) code to prevent forging $< y_i >$. In other words, random value $< r >$ can be assumed as a new MAC key and $< r.y >$ as the MAC and servers can change the y_i with probability of $1/p$ when p is large enough to neglect this possibility. Another random value $< u >$ is used as an authentication key for $< r >$ to prevent leakage of information about $< y >$ by iterative guessing of value r.

However, this protocol is not suitable for applications which need to store the data on the servers once and then use it several times because as soon as the MAC key is revealed, unused shares on the servers become useless. As a result,

π_{input} protocol: Input Supply.
Client holds value x that he wants to supply as input to the servers.
1. The servers do the following:
Retrieve an unused random $< k_i >$ from the pre-processed material and use the Output delivery protocol to give k to client.
2. Client broadcasts $e = x - k$ to the servers, and the servers compute $(x - k) + < k_i > = < x_i >$.

Figure 3.1: Input supply protocol

π_{output} :Output Delivery.
Given $< y_i >$ where we want to reveal y to the client and only to the client.
1. The servers send $< y_i >$ to the client as follows: Each server p_i sends its share y_i, related MAC $\gamma(y_i)$ and share of the MAC key α_i privately to the client.
2. Client does the following: compute $y = \sum y_i$, $\alpha = \sum \alpha_i$, $\gamma(y) = \sum \gamma(y_i)$ check that $\gamma(y) = y\alpha$. If not, abort, else output y.

Figure 3.2: Output Delivery protocol

the protocol needs to change the shares on the servers every time the function f is computed on the servers.

We utilise SPDZ2 to overcome this problem. Since this updated protocol does not reveal the MAC key, it is a good solution for storing the shares once and computing the different functions on it, without changing the shares. As the MAC key is not revealed to the servers, they can not forge the y_i and $\gamma(y_i)$, So it suffices that servers send the shares of the output y as y_i, related MAC $\gamma(y_i)$ and shares of the MAC key $\alpha_1, \alpha_2 ..., \alpha_i$ to the client so that the client do the MAC checking. Figure 3.2 shows the client-server output delivery protocol.

3.4 Using SPDZ2 Client-Server Model in Graph Secret Sharing (GSS)

Metadata is generated every day and is laborious for companies in terms of storing the data securely and searching on it properly. The graph is a general structure that is widely used for storing the structured data such as metadata in order to perform searching, and it is also a suitable structure for presenting the connections of the different entities such as people, IDs or telephone numbers. IBM System G [41], Node 4j [79] and GraphDB [81] are examples that use graph data structures for carrying out queries such as topological analyses, graph matching and so on. Therefore, as a solution to the metadata storage and querying issues highlighted in Chapter 1, we present a client-server model which is based on SPDZ2 in order to store and query metadata as graph shared data using an adjacency matrix representation. This model uses additive secret sharing for hiding edges of the graph in order to provide a flexible structure for storing and making queries. We assume that all vertices are encrypted on the client side and just edges are saved as an adjacency matrix on the cloud. In this model, we need servers with different cloud authorities because otherwise, the servers can easily collude in multi-party computation technique. The model covers three privacy properties:

1. Cloud authorities gain no information of stored data on cloud storage, except size of the data stored.

2. Cloud authorities cannot learn anything about the query inputs, though they do learn the query type.

3. Cloud authorities cannot learn the answer of the queries.

In the next section, we show how the GSS protocol can cover these privacy properties. A direct application of the GSS protocol is the metadata requested by authorities and legitimate users. There are many service providers or organisations that keep valuable metadata as a graph-structured data on the cloud storage, such as telecommunication companies which store the information about the connections between people.

In this scenario the data owner avoids to store a honeypot with one entity, so they make a secret-shared store with multiple entities on the cloud storage. Moreover, with leveraging of GSS protocol, the legitimate users such as police or government that need to search on the graph-structured metadata do not reveal the identity of the node that there are searching for to the attackers.

3.4.1 GSS Protocol

In this section, we propose GSS as a client-server protocol based on SPDZ2 secure multi-party computation in order to cover all the privacy properties in Section 3.4. In this protocol, there are two participants: the client and the cloud servers. The client needs to send the data S and q respectively to the cloud for computing the function F in such a way that $f(< S >, < q >) =< y >$ where S is the adjacency matrix, q is the query and y is the answer of the query. In this protocol, the client refers to the data owner and the querier, and both can be one entity or two different ones. Figure 3.3 shows the communications between client and servers. More precisely, the protocol consists of four steps as following:

1. Client uses protocol π_{input}, Figure 3.1, to generate the additive shares s_i along with MAC $\gamma(s_i)$ of adjacency matrix S on the servers.

2. Client uses protocol π_{input}, Figure 3.1, to generate the additive shares q_i along with MAC $\gamma(q)$ of query q on the servers.

3. Servers compute function f on inputs $< s >$ and $< q >$.

4. Servers use π_{output}, Figure 3.2, to send the shares y_i ,α_i, and $\gamma(y_i)$ to the client.

Step 1 of the protocol shows how the client can privately store the adjacency matrix S in the cloud storage as shares of the secret S and its related MAC. This can cover the privacy property 1.

The privacy property 2 is covered in Steps 2 and 3 of the GSS protocol. The cloud authorities will know that an adjacency (neighbour) or second-hop queries were made, but they don't know which people were queried or what the answer

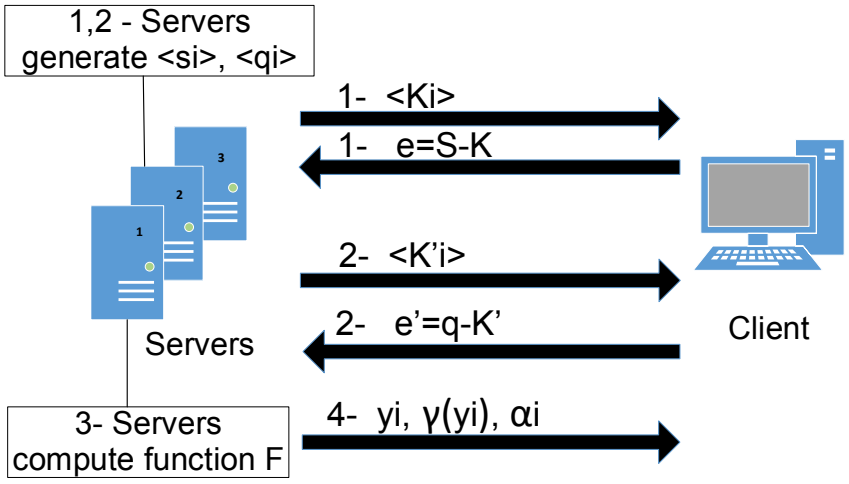


Figure 3.3: GSS protocol.

was. The secret sharing and software in SPDZ2 provide the private querying in GSS protocol. It means that the querier sends the shares of the query inputs to the the different servers and with leverage of the distributed equality test on the row and column indices in SPD2 software, the queries can be answered without revealing the query inputs to the servers. See Section 3.5 for more details.

The privacy property 3 is addressed by the last step of the GSS protocol. Every server sends just a share of the query answer to the querier, so none of the servers learns about the true answer.

Moreover, applying SPDZ2 to the client-server model can give the client the opportunity of storing the data on the cloud securely once and querying it several times while covering all steps of our privacy description.

3.4.2 Neighbourhood and Second-Hop Query

We apply two widely used queries in the graph searching scheme on GSS protocol which are namely neighbourhood and next hop queries. Neighbourhood query

is a straightforward query which asks the servers: is there any connections between two specific vertices? In this model, all of the vertices are saved on the client side with their assigned index, so the related indices of two vertices show the position of the desired elements in the adjacency matrix that is sent by the client to the servers. Therefore, the input of neighbourhood query function f_n is $q = (\alpha, \beta)$ where $(\alpha, \beta) \in N$, 'N' is set of vertices. Consequently, function of the neighbourhood query f_n computes $f_n(< s >, < q >) =< y >$ on the servers. Note that the servers do not learn the query or the answer.

A second-hop query asks whether two vertices are connected with at most one vertex in between. The query function is more complicated compared to the neighbourhood query, because it needs more computation and N multiplications where N is the number of vertices. To generate the query function f_s, suppose we have an adjacency matrix in which the row and column indices are known among a set of servers, but the contents of each cell (which is 0 or 1) is secret-shared among the servers over prime ρ. The second-hop neighbours are the relation found by multiplying the matrix by itself. That is, given an $N * N$ matrix S, we wish to find matrix S^2 where:

$$S_{ij}^2 = \sum_{k=1}^{N} S_{ik} S_{kj} \tag{3.1}$$

So, if α and β are indices which are assigned to vertices "A" and "B" for example, (α, β) pointing to element of the matrix S_i^2 which is the number of connections between "A" and "B", so the servers compute $f_s(< s >, < q >) =< y >$ and send $< y >$ to the client. In this function there is no need to multiply the whole matrix by itself—we could simply multiply the related row and column to α and β respectively and decrease the number of multiplications to $O(N)$. However, in order to hide α and β from the servers, we need secret share those values, iterate over all the rows and columns, and use an accumulator to collect the correct value. These and other implementation details are described in the next section.

Algorithm 3.3 Neighbourhood Query

Input: L, K matrix dimensions, A adjacency matrix, i, j query inputs, G random vector with size L

```
// Checking the connection between the nodes privately
```

1: $Accumulator1 \longleftarrow 0$
2: $s \longleftarrow 0$
3: **for** $t \in range(0, k)$ **do**
4: $\quad Accumulator2 \longleftarrow 0$
5: $\quad$ **for** $index_0 \in range(s, L + s)$ **do**
6: $\quad\quad Accumulator2 \longleftarrow Accumulator2 + (index_0 == j + t * k) * A_{index_0}$
7: $\quad$ **end for**
8: $\quad s \longleftarrow index_0 + 1$
9: $\quad G_t \longleftarrow Accumulator2$
10: **end for**
11: **for** $index_1 \in range(0, k)$ **do**
12: $\quad Accumulator1 \longleftarrow Accumulator1 + (index_1 == i) * G_{index_1}$
13: **end for**
14: **return** $Accumulator1$

3.5 Implementation

We used an accumulator to avoid revealing the intermediate results such as indices, rows and columns of the adjacency matrix. Moreover, we could not use condition commands directly on shared values, because we would need to reveal the result of the condition terms first and then use it in the program. However, the SPDZ software allows distributed evaluation of equality tests on secret-shared values. We iterate over the entire list (or matrix), using the distributed equality test on the row and column indices to accumulate only the values we wish to include. In other words, this evaluation technique is really a sub-protocol in which we're running a secure multiparty computation of a function that tests a pair of vectors for equality of their elements, but leaves the answer only in secret-shared form. So the computation complexity of neighbourhood and second-hop queries are $O(N^2)$ in the privacy-preserving form. The program for neighbourhood query is shown in Algorithm 3.3; second-hop query is shown in Algorithm 3.4.

Algorithm 3.4 Second-hop Query

Input: L, K matrix dimensions, A adjacency matrix, i, j query inputs, G random vector with size L, M zero array with size of L

`//Extracting row` i `and column` j `from matrix` A `without revealing` i `and` j

1: $Accumulator1 \longleftarrow 0$
2: $s \longleftarrow 0$
3: **for** $t \in range(0, k)$ **do**
4: $\quad Accumulator \longleftarrow 0$
5: $\quad P = -1$
6: $\quad$ **for** $index \in range(s, L + s)$ **do**
7: $\quad\quad p \longleftarrow p + 1$
8: $\quad\quad Accumulator2 \longleftarrow 0$
9: $\quad\quad Accumulator2 \longleftarrow Accumulator2 + (t == i) * A_{index}$
10: $\quad\quad Accumulator \longleftarrow Accumulator + (index == j + t * k) * A_{index}$
11: $\quad\quad M_p \longleftarrow M_p + Accumulator_2$
12: $\quad$ **end for**
13: $\quad s \longleftarrow index + 1$
14: $\quad G_k \longleftarrow Accumulator$
15: **end for**
16: $Accumulator3 \longleftarrow 0$
17: **for** $n \in range(0, k)$ **do**
18: $\quad Accumulator3 \longleftarrow Accumulator3 + G_n * M_n$
19: **end for**
20: **return** $Accumulator3$

3.6 Experimental Results

We used SPDZ2 software [80], to examine the neighbourhood and next-hop queries for dataset of 64 vertices. The dataset is an undirected graph that shows the dolphins society connection [52]. We performed neighbourhood query, shown in Algorithm 3.3 and second-hop query, Algorithm 3.4 on the dataset to test the computation time. The run times are given for two, three and four servers and all the results are obtained on 2.5 GHz Intel core i7 machine with 4GB RAM. All the players are running on one machine, so we neglect the time of sending and receiving data in the network and just consider the computation time in the online phase.

Table 3.1: Neighbourhood and Second-hop Query Computation Time for Dataset of 64 Nodes

Number of players	Neighbourhood query Time(s)	Second-hop query Time(s)
2	0.43	0.82
3	0.46	0.94
4	0.50	1.84

Tab. 3.1 shows the timing of neighbourhood and second-hop query for 2, 3 and 4 parties; for neighbourhood query the timing does not show significant change by increasing the number of parties while for second-hop query the timing is significantly changing due to increasing the number of parties.

One implementation detail is that the SPDZ software does not currently allow the client-server model—it assumes that everyone who shares a value is also involved in the multi-party computations. Hence we implemented the data owner as simply one of the multi-party computation participants—this gives accurate timings, though a real implementation would have to alter SPDZ to allow this party to share its data and then not participate in the answering of queries.

To set up the online phase, the program needs to generate 93744 triples and 117180 bits in the offline phase for performing neighbourhood query and 184574 triples and 230640 bits for performing second-hop query of 64 vertices that most of the them are used to build the accumulators.

3.7 Concluding Remarks

In this chapter, we proposed a technique for storing and querying metadata, represented as the adjacency matrix of a graph. We utilised SPDZ2 in the client-server model and conducted neighbourhood and next-hop queries. Using the SPDZ2 protocol instead of SPDZ in the client-server model gave us the advantage of storing once and computing several times on the stored data. This technique seems suitable for applications that need one-off archiving followed by multiple

queries by a trusted client.

Moreover, the adjacency matrix representation of the graph provided a flexible structure to make a wide range of queries such as neighbourhood and second-hop queries. It would also allow even more advanced topological analysis such as finding cliques—we leave this for future work. We also implemented the queries on the SPDZ2 platform to find out the computation time. Although the results are not quite satisfying for real time applications, they are acceptable for applications which do not need real time computations, such as law enforcement queries of metadata.

Chapter 4

Differentially Private Two-Party Egocentric Betweenness Centrality

In Chapter 3 we examined the SPDZ2 secure multi-party protocol for privately answering queries on graph-structured data stored in an untrusted cloud. One drawback of this approach is increased computational complexity when the query demands a high level of comparisons or multiplications. In this chapter[1] we turn to a more sophisticated form of graph query: computing the egocentric betweenness centrality of a node when relevant edge information is spread between two distrusting parties. More precisely, each node belongs to one of two networks, with relevant ego network edges may span both networks. We develop a sequence of differentially private mechanisms to hide each network's internal edge structure from the other. We contribute a new two-stage stratified sampling algorithm which delivers an exponential improvement to computational complexity of naïvely implementing the exponential mechanism of differential privacy. Empirical results on open graph data sets demonstrate practical relative error rates under strong privacy guarantees.

[1]This chapter is based on the following article:
Leyla Roohi, Benjamin IP Rubinstein, and Vanessa Teague. Differentially private two-party egocentric betweenness centrality. In IEEE INFO-COM 2019-IEEE Conference on Computer Communications, pages 2233–2241. IEEE, 2019.

4.1 Introduction

Datasets such as social, communication, and transport networks are graph structured: people are nodes and their interactions edges. Such graph structures are valuable for understanding real-world properties. However, revealing the graph or its statistics can cause privacy disclosure even with anonymisation techniques [4,71,73].

Furthermore, for many corporations, customer data is an asset they are reluctant to share. This motivates interest in joint computation over databases with limited exposure to sensitive information.

Differential privacy (DP) [35] guarantees that a release output distribution does not change by more than a small multiplicative factor under input data perturbation. We consider *edge DP* wherein perturbations correspond to edge flips: the existence of sensitive edges is not revealed by edge-DP release.

In this chapter, we envisage two networks controlled by different corporations, such as telephone or email providers, or two different social networks. The complete list of nodes (*i.e.*, people) is public knowledge, but the individual connections between them are not, so we consider edge DP in order to hide the connection between the nodes in each network. Each service provider knows the connections within its own network, plus the connections between each of its members and the outside (*e.g.*, when they contact someone in a different network), but not the internal connections in other networks.

We are the first to consider differentially private computation of *egocentric betweenness centrality* (EBC) [43].

Informally, EBC measures the importance of a node as a link between different parts of the graph. A node that forms a link between otherwise-isolated parts of the network has high betweenness centrality; a node that is simply an easily-bypassed member of an interconnected network has a low betweenness centrality. This is a property of the whole communication graph: one service provider cannot compute it using its partial view of the graph alone.

Betweenness centrality could be used in targeted advertising or customer retention campaigns, as individuals with high EBC have the capacity to transfer information from one community to another. EBC is equally important in

understanding and combating the spread of misinformation or "fake news": individuals with high EBC can be educated to be more discerning about what they spread through the network, thereby mitigating spread of fake news. The difficulty of assessing misleading political content and obstructing its spread has become one of the most important research questions in online social network analysis, to which even the networks themselves are devoting significant research effort.[2]

We enable a network provider to compute the egocentric betweenness centrality of a node, while requiring only differentially private information about internal connections to be shared between networks. In this chapter, our main contributions are:

1. We introduce a privacy preserving method to compute the egocentric betweenness centrality of nodes in undirected graphs. In this work the network has local connections, while there are also inter-network connections.

2. We propose a two-stage sampling process that delivers a simple approach to implement and exponential savings in time and space over naïve sampling from the exponential mechanism directly.

3. We report on thorough experiments using a Facebook graph dataset on 63,000 nodes. The experiments in Section 4.5 show that the error is approximately 16% of the true EBC for reasonable values of privacy level ϵ. Similar results hold for other networks from Enron and PGP email.

4.2 Two-Party Protocol

Consider a two-party setting of a telecommunications network with two service providers X, Y: every customer is represented as a node a that belongs to one and only one service provider; pairs of customers who *e.g.*, have called one another are represented as edges in a simple undirected graph on the disjoint union of nodes. Edges can either connect nodes within one party (X or Y) in which case are unknown to the other party (Y or X respectively), or edges span both parties

[2] `https://newsroom.fb.com/news/2018/04/new-elections-initiative/`

Table 4.1
Glossary of symbols used in this chapter.

X, Y	The two parties *e.g.*, competing service providers.
V_X, V_Y	The nodes per party.
E_X, E_Y	Edges entirely within each party.
E_{XY}	Edges spanning both parties.
a	The ego node (assumed WLOG to be in X).
N_a	The ego network of a.
X^-	Party X's nodes V_X excluding a.
$R^\star$	The ego network contained in X.
T_{ij}	Counts of 2-paths spanning X, Y.
S_X, S_Y, S_{XY}	Partial EBC sums by endpoints.
R	A private, randomised approximation to $R^\star$.
Q_i	For $i \in \{0, \ldots, \lvert X^- \rvert\}$, a partition of $\mathcal{P}(X^-)$.
ϵ	The differential-privacy budget.
Δ	A global sensitivity bound.

and are known to both. We consider all nodes to be known to both parties, as being addressable within a global addressing system (*e.g.*, a phone book).

Denote by V_X, V_Y the nodes of X, Y respectively, $E_X \subseteq V_X, E_Y \subseteq V_Y$ the edges (two-element sets) within parties X, Y respectively, and $E_{XY} \subseteq V_X \cup V_Y$ the edges spanning X, Y as sets with one element each from V_X, V_Y. The simple undirected graph on the entire network comprises node-set disjoint union $V_X \cup V_Y$ and edge-set disjoint union $E_X \cup E_Y \cup E_{XY}$. Note we will often equivalently represent edge sets as adjacency matrices (or flattened vectors) with elements in $\{0, 1\}$. Table 4.1 shows all of the symbols used in this chapter.

We wish to enable one party (without loss of generality) X to compute the ego betweenneess centrality (EBC) of one of its nodes $a \in V_X$, while maintaining *edge privacy* between parties. Before detailing a protocol for accomplishing this task, we must be precise about a privacy model.

Problem 9 (Private Two-Party EBC)**.** *Consider a simple undirected graph* $(V_X \cup V_Y, E_X \cup E_Y \cup E_{XY})$ *partitioned by parties* X, Y *as above, and an arbitrary node* $a \in V_X$*.*

The problem of private two-party egocentric betweenness centrality *is for the parties* X, Y *to collaboratively approximate* $\mathrm{EBC}(a)$ *under assumptions that:*

A1. Both parties X, Y *know the entire node set* $V_X \cup V_Y$*;*

A2. Each party knows every edge incident to nodes within their own network. That is, X *knows* $E_X \cup E_{XY}$ *while* Y *knows* $E_Y \cup E_{XY}$*; and*

A3. The computed $\mathrm{EBC}(a)$ *needs to be available to* X *but need not be shared with* Y*.*

Any solution must not reveal to X, Y *what is not already known except for* X *discovering* $\mathrm{EBC}(a)$ *(Assumption 3). We seek solutions under an honest-but-curious adversarial model: while* X, Y *will follow any agreed upon protocol prescribing computations to take and messages to send to one-another, without attempting to manipulate the other party; each party is curious about the other's edges and may apply arbitrary auxiliary computation and leverage data sources in attempting to discover the other's edges. Formally, what is revealed by* X *(*Y*) to* Y *(respectively* X*) must preserve* ϵ*-differential privacy with respect to* E_X *(respectively* E_Y*).*

4.3 Warm-Up: A Non-Private Protocol

We first consider how X, Y might cooperate without preserving differential privacy. In particular X cannot itself count 2-paths that are

- Contained entirely within Y; or
- Ending in both X, Y with intermediate node in Y.

Any protocol must involve Y in aggregating over such paths. But while the first case can be aggregated independently by Y, the second case requires X to communicate its endpoint neighbours of a to Y. This significantly complicates the differentially private solution developed in the next section.

Recall that $N_a = \{v \in V \mid \{v, a\} \in E\}$ denotes the ego network of a anywhere in the graph (notably not including a since the graph has no self-loops). Figure 4.1 summarises the following protocol.

Protocol 10. *Proceeding in sequence:*

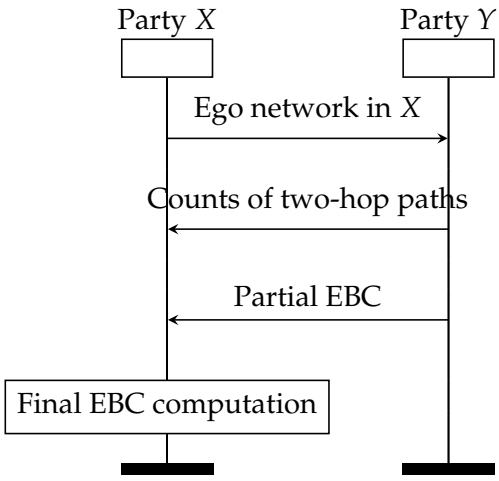


Figure 4.1: EBC two-party computation protocol, comprising one forward and two backward messages.

i. *[Forward message] X sends to Y the set* $R^\star = N_a \cap V_X$ *of neighbours of a contained within X;*

ii. *[Backward message] Y computes and sends to X, for each* $i \in R^\star$ *and for each* $j \in N_a \cap V_Y$ *(where* i, j *are not directly connected), a count* T_{ij} *of 2-paths with endpoints* i, j *and intermediate point in* $N_a \cap V_Y$*;*

iii. *[Backward message] Y computes and sends to X, the EBC partial sum over endpoint nodes* $i, j \in N_a \cap V_Y$ *with intermediate nodes in* $N_a \cup \{a\}$*. That is,* $S_Y = \sum_{i,j \in N_a \cap V_Y : A_{ij}=0, j>i} 1/A^2_{ij}$*;*

iv. *X increments the received* T_{ij} *by the number of 2-paths between* i, j *with intermediate point in* $R^\star \cup \{a\}$*. It then sets* S_{XY} *to the sum of their reciprocals;*

v. *X computes, over distinct and disconnected endpoint nodes* $i, j \in R^\star$ *with intermediate node in* $N_a \cup \{a\}$*, the EBC partial sum. That is:* $S_X = \sum_{i,j \in R^\star : A_{ij}=0, j>i} 1/A^2_{ij}$*; and*

vi. *X completes computation of* $\mathrm{EBC}(a)$ *as* $S_X + S_{XY} + S_Y$*.*

Remark 11. *We now briefly comment on where edge privacy is potentially breached, thereby highlighting challenges faced by any solution to Problem 9. When X sends Y*

its set of neighbours $R^\star$ of a, party Y learns directly of all edges incident to a in X. When Y sends X its 2-path counts T_{ij}, while counts aggregate exact connectivity at worst this level of aggregation could be very small therefore revealing information about connections to a within Y, and the inter-connections between these nodes. A worst case occurs when there are two nodes in Y connected to a: as soon as X receives the vector of counts of 2-paths spanning X, Y, it can learn whether these two nodes are connected.

4.4 A Privacy-Preserving Protocol

We now develop our protocol for private EBC which involves a series of differentially-private mechanisms for overcoming the privacy disclosures identified in Remark 11.

We use the exponential mechanism (*viz.* Lemma 4 in Chapter 2) to release a set R of nodes in X that privately approximates a's ego network $R^\star$ in X. This is Protocol 10.i's 'forward message' (Section 4.4.1).

While our application of the exponential mechanism protects edge privacy for X, there is still potential for privacy disclosure when Y communicates counts to X. To overcome this problem, we leverage the Laplace mechanism to privatise vectors of 2-path counts communicated by Y within Protocol 10.ii's 'backward message' (Section 4.4.2). The components of this message are indexed (in part) by the approximate $R \approx R^\star$ sent in the forward message. A second Laplace mechanism makes private the partial EBC of Protocol 10.iii's 'backward message'.

In this way our privacy-preserving protocol follows the broad-brush sequence of steps outlined in Section 4.3 but is made more involved by the addition of differential privacy.

4.4.1 Forward Message

The goal of the forward message is to communicate a privacy-preserving approximation to $R^\star = N_a \cap V_X$ chosen from power set $\mathcal{P}\left(V_X \backslash \{a\}\right)$. In order to leverage the exponential mechanism (*viz.* Lemma 4) we must specify a quality function of the form $q : \{0,1\}^{|V_X|^2} \times \mathcal{P}\left(V_X \backslash \{a\}\right) \to \mathbb{R}$. That is a mapping from the $V_X \times V_X$ adjacency matrix for E_X and a candidate response $R \subseteq V_X \backslash \{a\}$, to

a score reflecting the approximation quality of $R^\star$ by R. Since the response set is finite (albeit exponential in the graph size), the exponential mechanism then has normalised response probability mass function,

$$\frac{\exp(q(R) \cdot \epsilon/(2 \cdot \Delta q))}{\sum_{S \subseteq X} \exp(q(S) \cdot \epsilon/(2 \cdot \Delta q))} \ , \tag{4.1}$$

with implicit dependency on fixed E_X adjacency matrix.

In designing an appropriate quality function, we typically want q to be maximised uniquely by the desired non-private output $R^\star$. The function should also be a semantically meaningful 'distance' between outputs R and $R^\star$ such that the utility bounds for the exponential mechanism of [67, Lemma 7 and Theorem 8] make meaningful guarantees. The utility guarantee states that with high probability the released random R has q score not too much lower than $\max_{S \subseteq V_X \setminus \{a\}} q(S)$. And so if $R^\star$ meets this global maximum, then we have that the released set has score not much lower than that of $R^\star$. If responses close in q are also 'close' then this guarantees a good approximation to $R^\star$ with high probability.

Remark 12. *A natural choice for quality function is $q(R) = |R \cap R^\star|$ as it is clearly maximised by $R^\star$. However it is not uniquely maximised, indeed for any superset $R^\star \subset S \subseteq V_X \setminus \{a\}$ (including the entire set of nodes) we have that $q(S) = |S \cap R^\star| = |R^\star|$ also. There are many such sets: $2^{|V_X|-1-|R^\star|}$ which is not far from the number of all possible responses $2^{|V_X|-1}$ for modest ego network sizes, in which case the exponential mechanism does not achieve our goal.*

Section 4.4.1.1 develops a sound choice of quality function in the symmetric set difference.

In the rest of this section we will abuse notation and abbreviate $X = V_X$ with the meaning understood from context. We will also denote by $X^- = V_X \setminus \{a\}$.

4.4.1.1 Symmetric Set Difference

We adopt (for minimisation) the symmetric set difference with $R^\star$ given by $(R \setminus R^\star) \cup (R^\star \setminus R)$ as a promising basis for quality function design. Define com-

plements $\overline{B} = X^{-}\backslash B$ relative to X^{-}. We have

$$\begin{aligned} (R\backslash R^{\star}) \cup (R^{\star}\backslash R) = &\quad X^{-}\backslash \left((R \cap R^{\star}) \cup \overline{R \cup R^{\star}}\right) \text{ ; and} \\ |(R\backslash R^{\star}) \cup (R^{\star}\backslash R)| = &\quad |X^{-}| - |R \cap R^{\star}| - |\overline{R \cup R^{\star}}| \ , \end{aligned} \tag{4.2}$$

where the second equality follows from a disjoint union in the first equality's right-hand side.

In minimising the symmetric difference, dismissing the constant $|X^{-}|$ as redundant to optimisation, we can equivalently maximise[3]

$$q(R) = \quad |R \cap R^{\star}| + |\overline{R \cup R^{\star}}| \ . \tag{4.3}$$

This quality function takes values in $\{0, \ldots, |X^{-}|\}$ and is uniquely maximised by $R^{\star}$.

While the machinery of the exponential mechanism only requires sensitivity of this quality function (Section 4.4.1.5) to guarantee differential privacy, a significant challenge is involved in sampling from the mechanism's response distribution as it is defined over an enormous response space: the power set of X^{-}. Thanks to the amplification of q by the exponential, the distribution's mass varies an incredible amount even for graphs of modest size.

4.4.1.2 Equi-Quality Responses

It will be useful to consider the sets of candidate exponential mechanism responses, with equal quality score value i,

$$Q_i = \quad \{R \subseteq X^{-} : q(r) = i\} \ , \tag{4.4}$$

where $i \in \{0, \ldots, |X^{-}|\}$. It can be shown that the Q_i form a partition of $\mathcal{P}(X^{-})$: the sets are pairwise disjoint, and their union is all subsets of X^{-}. It can also be shown that for $i \in \{0, \ldots, |X^{-}|\}$,

$$|Q_i| = \sum_{k=0}^{i} \binom{|R^{\star}|}{k} \binom{|X^{-}| - |R^{\star}|}{i-k} = \binom{|X^{-}|}{i}.$$

[3]While q's dependence on $R^{\star}$ is suppressed, it should be implicitly understood.

A consequence of this identity is an efficient approach to computing the normalising constant for the exponential mechanism response distribution.

Corollary 13. *Consider the normalised exponential mechanism response distribution* (4.1) *for the quality function given in* (4.3)*. The normalising constant is equivalent to*

$$C = \sum_{i=1}^{|X|} |Q_i| \exp\left(\frac{i \cdot \epsilon}{2\Delta}\right) \quad . \tag{4.5}$$

Computing this expression takes time and space $O(|X|)$.

Note that other phases of our protocol, to be specified, also require linear space. By comparison computing C naïvely would take time exponential in X and constant space.

Proof. Consider the denominator of the exponential response distribution (4.1):

$$\begin{aligned} C &= \sum_{R \subseteq X} \exp\left(\frac{q(R) \cdot \epsilon}{2\Delta}\right) = \sum_{i=1}^{|X|} \sum_{R \in Q_i} \exp\left(\frac{q(R) \cdot \epsilon}{2\Delta}\right) \\ &= \sum_{i=1}^{|X|} \sum_{R \in Q_i} \exp\left(\frac{i \cdot \epsilon}{2\Delta}\right) = \sum_{i=1}^{|X|} |Q_i| \exp\left(\frac{i \cdot \epsilon}{2\Delta}\right) \quad , \end{aligned}$$

establishing the result.

□

4.4.1.3 Two-Stage Sampling

We will now outline how to sample from the exponential mechanism response distribution, using a two-stage sampling process that delivers a simple approach to implement, and exponential savings in time and space vs naïve sampling from the exponential mechanism.

Remark 14. *Another standard approach to sampling from challenging distributions is acceptance-rejection sampling or the Metropolis algorithm. However no clear surrogate probability mass presents itself that yields acceptable rates of rejection for practical applicability.*

Stratified Sampling To simplify notation, let us consider the problem at-hand more generally: let $V \in \mathcal{V}$ be a discrete random variable on finite probability space $(p, \mathcal{V})$ *i.e.*, a multinomial with $p : v \in \mathcal{V} \mapsto \Pr(V = v)$. Suppose there exists a partition of $\mathcal{V}$ into the disjoint union of $\mathcal{V}_0, \ldots, \mathcal{V}_k$, such that for all i, and all $v, v' \in \mathcal{V}_i$, $p(v) = p(v')$ *i.e.*, the probability mass is constant within each part. We can exactly sample from $V \sim p$ by (1) drawing a random variable that selects a part in the partition according to the relative part sizes then (2) sampling uniformly from within the chosen part—this is the approach taken by Algorithm 4.1 FORWARDMESSAGE. Denote by p_i the constant probability mass $p(v)$ of any $v \in \mathcal{V}_i$.

Lemma 15. *Define random variable $I \in \{0, \ldots, k\}$ where $\Pr(I = i) \propto |\mathcal{V}_i| \cdot p_i$ for each $i \in \{0, \ldots, k\}$, and $U \sim \text{Unif}(\mathcal{V}_I) \mid I$. Then $\Pr(V = v) = \Pr(U = v)$ for all $v \in \mathcal{V}$. Moreover, the probability mass stated for $\Pr(I = i)$ is already normalised,* i.e., *$\Pr(I = i) = |\mathcal{V}_i| \cdot p_i$.*

Proof. The proof follows by splitting on I, the chain rule of probability, and by definition of the r.v.'s. For any $v \in \mathcal{V}$, denote $i(v) \in \{0, \ldots, k\}$ to be such that $v \in \mathcal{V}_{i(v)}$, then

$$\begin{aligned}
\Pr(U = v) &= \sum_{i=0}^{k} \Pr(U = v, I = i) \\
&= \sum_{i=0}^{k} \Pr(U = v \mid I = i) \cdot \Pr(I = i) \\
&= \Pr(U = v \mid I = i(v)) \cdot \Pr(I = i(v)) \\
&= \left(\frac{1}{|\mathcal{V}_{i(v)}|} \right) \cdot \left(\frac{|\mathcal{V}_{i(v)}| \cdot p_{i(v)}}{\sum_{j=0}^{k} |\mathcal{V}_j| \cdot p_j} \right) \\
&= \frac{p_{i(v)}}{\sum_{j=0}^{k} |\mathcal{V}_j| \cdot p_j} \\
&= p_{i(v)} \\
&= \Pr(V = v) \ .
\end{aligned}$$

The penultimate equality follows from

$$1 = \sum_{v' \in \mathcal{V}} \Pr(V = v') = \sum_{i=0}^{k} \sum_{v' \in \mathcal{V}_i} p_i = \sum_{i=0}^{k} |\mathcal{V}_i| \cdot p_i \ .$$

This also establishes that the probability mass $\Pr(I = i) \propto |\mathcal{V}_i| \cdot p_i$ is already normalised. □

Corollary 16. *The following sampling process, implemented in* FORWARDMESSAGE *Algorithm 4.1, is equivalent to sampling R from the exponential mechanism* (4.1)
(i) Sample $I \in \{0, \ldots, |X^-|\}$ with log-space probability mass

$$\begin{aligned} &\log \Pr(I = i) \\ &= \begin{cases} -|X^-| \log\left(1 + \exp\left(\frac{\epsilon}{2\Delta}\right)\right), & i = 0 \\ \log(|X^-| - i + 1) - \log i + \log \Pr(I = i - 1), & \textit{o.w.} \end{cases} \end{aligned}$$

(ii) Sample $R \sim \mathrm{Unif}\,(Q_I) \mid I$

Proof. As the exponential mechanism (4.1) on $R \in \mathcal{P}(X^-)$, with quality function (4.3), is a multinomial distribution with strata of constant probability given by the Q_i, Lemma 15 establishes that the stratified sampler successfully implements the mechanism. All that remains is to compute the probability of selecting Q_i as the stratum's cardinality times constant probability (normalised by C as given in Corollary 13).

$$\Pr(I = i) \ = \ |Q_i| \cdot \frac{\exp\left(\frac{i \cdot \epsilon}{2\Delta}\right)}{C} \ = \ \frac{\binom{|X^-|}{i} \exp\left(\frac{i \cdot \epsilon}{2\Delta}\right)}{\left(1 + \exp\left(\frac{\epsilon}{2\Delta}\right)\right)^{|X^-|}} \ ,$$

where the final equality follows from the observation that the expression for C is a Binomial expansion. We simplify and convert this expression to log-space as the expression exponentially increases in i:

$$\begin{aligned} &\log \Pr(I = i) \\ &= \log \binom{|X^-|}{i} + \frac{i \cdot \epsilon}{2\Delta} - |X^-| \log\left(1 + \exp\left(\frac{\epsilon}{2\Delta}\right)\right) \end{aligned}$$

Algorithm 4.1 FORWARDMESSAGE Two-Stage Sampler

Input: edge set E_X; ego node $a \in V_X$; $\epsilon, \Delta > 0$

$R^\star \longleftarrow \{v \in V_X \mid \{v, a\} \in E_X\}$
$X^- \longleftarrow V_X \backslash \{a\}$
$I \longleftarrow$ INVERSETRANSFORMSAMPLER$(|X^-|, \epsilon, \Delta)$
$R \longleftarrow$ PICKANDFLIPSAMPLER$(X^-, R^\star, I)$
return R

Algorithm 4.2 INVERSETRANSFORMSAMPLER

Input: cardinality $|X^-|$; $\epsilon, \Delta > 0$

`// Compute log-space PDF of` I
$p_0 \longleftarrow -|X^-| \log\left(1 + \exp\left(\frac{\epsilon}{2\Delta}\right)\right)$
for $i \in [|X^-|]$ **do**
  $p_i \longleftarrow p_{i-1} + \log(|X^-| - i + 1) - \log i$
end for
`// Search in CDF for random quantile`
$\psi \longleftarrow -\mathrm{Exp}(1)$
$c \longleftarrow p_0$
for $I \in [|X^-|]$ **do**
  if $c \geq \psi$ **then**
    return $I - 1$
  end if
  $c \longleftarrow \log(\exp(c) + \exp(p_I))$
end for
return I

$$= \begin{cases} -|X^-| \log\left(1 + \exp\left(\frac{\epsilon}{2\Delta}\right)\right), & i = 0 \\ \log(|X^-| - i + 1) - \log i + \log \Pr(I = i - 1), & i > 0, \end{cases}$$

completing the result. □

4.4.1.4 Linear-Complexity Sampling

While Corollary 16 reduces the problem from sampling from a large support set with highly skewed probability mass, we must address efficient implementation of the two-stage sampling.

Algorithm 4.3 PICKANDFLIPSAMPLER

Input: node set $X^{\text{-}}$; node set $R^\star \subseteq X^{\text{-}}$; $I \in \{0, \ldots, |X^{\text{-}}|\}$

$R \longleftarrow R^\star$
$V_1, \ldots, V_{|X^{\text{-}}|-I} \sim \text{Unif}\,(X^{\text{-}})$ without replacement
for $j \in [|X^{\text{-}}| - I]$ **do**
  if $V_j \in R$ **then**
    $R \longleftarrow R \backslash \{V_j\}$
  else
    $R \longleftarrow R \cup \{V_j\}$
  end if
end for
return R

Inverse Transform Sampling of I The sampling of multinomial I over much smaller support set $\{0, \ldots, |X^{\text{-}}|\}$ can be accomplished efficiently via *inverse transform sampling*. Given access to a random variable Z's (invertible) CDF F_Z, one can sample realisations of Z by first sampling $\psi \sim \text{Unif}\,([0,1])$ then releasing quantile $F_Z^{-1}(\psi)$. For $Z \in \mathbb{Z}$, we can always take $F_Z^{-1}(\psi) = \inf\{z \in \mathbb{Z} \mid F_z(z) \geq \psi\}$. However highly-skewed distributions can suffer from numeric instability in floating-point computation of the CDF. Though not as severe as for R, this remains a problem for I. To combat this we employ a library for arbitrary floating-point precision in our implementation (see Section 4.5) and we represent I's probability mass in log space as reported in Corollary 16.

Proposition 17. *Consider r.v. I defined in Corollary 16. It can be sampled via the* INVERSETRANSFORMSAMPLER *(Algorithm 4.2) in time and space $O(|X^{\text{-}}|)$ given oracle access to exponential r.v.'s.*

Proof. The pseudo-inverse of the CDF follows the general case, while it is easy to show that $-\log Z$, for $Z \sim \text{Unif}\,([0,1])$, is distributed as $\text{Exp}(1)$. The time complexity corresponds to both computing the CDF (which need not be stored in its entirety) and a linear search for its inversion. □

Pick-and-Flip Sampling of R After sampling I, we must sample R uniformly from within chosen stratum Q_I, a constrained and potentially large subset of $\mathcal{P}(X^{\text{-}})$. For sampled R we are to have $q(R) = |R \cap R^\star| + |\overline{R \cup R^\star}| = I$, so the

size of R and $R^\star$'s symmetric set difference is $|X^\text{-}| - I$. Moreover, this describes all candidates for R within Q_I.

Proposition 18. *If r.v. I is sampled with* INVERSETRANSFORMSAMPLER *(Algorithm 4.2), then Algorithm 4.3* PICKANDFLIPSAMPLER *yields a sample of r.v. R defined in Corollary 16 in time and space $O(|X^\text{-}|)$.*

Proof. Every time a new node V_j sampled from $X^\text{-}$, it could be sampled from either $X^\text{-} \setminus R^\star$ or $R^\star$. In both cases, V_j is added to the set difference. This loop invariance continues until the set difference size reaches $|X^\text{-}| - I$ establishing $R \in Q_I$ and uniformly so due to the uniform sampling of the V_j. Sampling the nodes (like the rest of the algorithm) can be achieved in linear time/space with Fisher-Yates shuffling. □

4.4.1.5 Quality Function Global Sensitivity

The remaining ingredient for invoking the exponential mechanism to privately release $R \approx R^\star = N_a \cap V_X$, is bounding q's sensitivity.

Lemma 19. *Consider any fixed $R \subseteq X^\text{-}$ and X-contained ego networks $R^\star$, $R^{\star\prime}$ induced by neighbouring adjacency matrices on E_X and fixed ego node a. Noting explicitly the dependence of the quality function* (4.3) *on non-private ego network, $|q(R, R^\star) - q(R, R^{\star\prime})| \leq 1$.*

Proof. Consider the effect of switching an edge within X on the symmetric difference cardinality between $R, R^\star$ *i.e.*, the quality function. Adding/removing an edge can impact at most one node being neighbours with ego node a; it can therefore only decrease or increase the first or second terms of q by 1, at most. Since these two sets are disjoint, it cannot change both simultaneously. □

Theorem 20. FORWARDMESSAGE *(Algorithm 4.1) takes time and space $O(|X^\text{-}|)$, and when run with $\Delta = 1$, preserves ϵ-differential privacy of the edge set E_X within party X's network.*

Proof. Privacy follows from Lemma 4, Corollary 16 and Lemma 19, complexity from Propositions 17 and 18. □

Algorithm 4.4 BACKWARDMESSAGE

Input: ego node $a \in V_X$; edge sets E_{XY}, E_Y; private node set $R \subseteq V_X \backslash \{a\}$; $\epsilon, \Delta_1, \Delta_2 > 0$

`// Count 2-paths of type` $X - Y - Y$

1: **for** $i \in R$ and $j \in N_a \cap V_Y$ **do**
2: $K \longleftarrow \{k \in N_a \cap V_Y \mid \{i,k\} \in E_{XY}, \{k,j\} \in E_Y\}$
3: $T_{ij} \longleftarrow |K| + \text{Lap}(2\Delta_1/\epsilon)$
4: **end for**

`// Partial EBC sum over 2-paths in` Y

5: $E'_Y \longleftarrow E_{XY} \cup E_Y$
6: $S_Y \longleftarrow 0$.
7: **for** $i, j \in N_a \cap V_Y$ with $j > i$ and $\{i,j\} \notin E_Y$ **do**
8: $K \longleftarrow \{k \in R \cup \{a\} \cup (N_a \cap V_Y) \mid \{i,k\}, \{k,j\} \in E'_Y\}$
9: $S_Y \longleftarrow S_Y + \frac{1}{|K|}$
10: **end for**
11: $S_Y \longleftarrow S_Y + \text{Lap}(2\Delta_2/\epsilon)$
12: **return** $\mathbf{T}, S_Y$

4.4.2 Backward Message

Analogous to the non-private protocol of Section 4.3 (Figure 4.1), Y receives node-set R approximating a's ego network in X, via FORWARDMESSAGE. Subsequently, Y must send back: counts T_{ij} of 2-paths spanning X, Y with intermediate node in Y—indexed by R; and its partial EBC sum S_Y over paths with endpoints in Y and intermediate node in $R \cup \{a\} \cup (N_a \cap V_Y)$. Note this last set is the private appoximation to $N_a \cup \{a\}$. We apply in BACKWARDMESSAGE (Algorithm 4.4) the Laplace mechanism (Lemma 3) to both backward message components to avoid disclosure of edges E_Y in Y.

Note cases in which BACKWARDMESSAGE need not be run by X: If $|N_a \cap V_Y| \leq 1$ there are no paths incident to Y with intermediate point in Y; or contained entirely within Y. We may therefore assume within the algorithm that these cases are not present.

Algorithm 4.5 PRIVATEEBC

Input: node sets V_X and V_Y; T; S_Y; ego node $a \in V_X$; edge sets E_{XY}, E_Y

// Party X runs:

$R \longleftarrow$ FORWARDMESSAGE$(E_X, a, \epsilon, \Delta_0)$

Send R to party Y

// Party Y runs:

$\mathbf{T}, S_Y \longleftarrow$ BACKWARDMESSAGE$(a, E_{XY}, E_Y, R, \epsilon, \Delta_1, \Delta_2)$

Send $\mathbf{T}, S_Y$ to party X

// Party X runs:

$S_{XY} \longleftarrow 0$

for $i \in R^\star$ and $j \in N_a \cap V_Y$ where $\{i,j\} \notin E_{XY}$ **do**

  $K \longleftarrow \{k \in R^\star \cup \{a\} \mid \{i,k\} \in E_X, \{k,j\} \in E_{XY}\}$

  if $i \notin R$ **then**

    $T_{ij} \longleftarrow 0$

  end if

  $T_{ij} \longleftarrow T_{ij} + |K|$

  $S_{XY} \longleftarrow S_{XY} + \frac{1}{T_{ij}}$

end for

$S_X \longleftarrow 0$

for $i, j \in R^\star$ where $j > i$ and $\{i,j\} \notin E_X$ **do**

  $K \longleftarrow \{k \in N_a \cup \{a\} \mid \{i,k\} \in E_X, \{k,j\} \in E_X\}$

  $S_X \longleftarrow S_X + \frac{1}{|K|}$

end for

return $S_X + S_{XY} + S_Y$ to party X

4.4.2.1 Privately Counting Paths

The first part of the backward message compares the T_{ij}—noisy counts of 2-paths with intermediate node in $N_a \cap V_Y$—over i in the given R approximating $R^\star$, and $j \in N_a \cap V_Y$. The sensitivity of these counts relates to adding or removing an edge in E_{XY}.

Lemma 21. *Let query f denote the vector-valued non-private response* $\mathbf{T}$*. The L_1-global sensitivity of f is upper-bounded by $\Delta f = 2|R|$.*

Proof. Suppose that graphs G_1 and G_2 differ in some edge $\{j,k\}$ with j and k both in $N_a \cap V_Y$ (that is, the edge $\{j,k\}$ would belong to E_Y). Our task is to upper bound the corresponding change to counts $\|f(G) - f(G')\|_1$ resulting from run-

ning query f on the two graphs. There can be at most R choices of endpoint node $i \in R$ for forming 2-hop paths (i, k, j) affected by the addition/deletion. Similarly there can be at most R choices of endpoint $i \in R$ for paths (i, j, k) affected by the addition/deletion. For each of these $2R$ paths the addition/deletion can affect $\|f(G) - f(G')\|_1$ by at most 1. This proves the result. □

4.4.2.2 Private Partial EBC

The second part of BACKWARDMESSAGE is a partial EBC sum over 2-paths with end-points in $N_a \cap V_Y$ (and intermediate point in either the same or $R \cup \{a\}$) as in Protocol 10.iii. We again apply the Laplace mechanism to avoid privacy disclosure of edges in Y, which requires bounding sensitivity of the non-private sum as follows.

Lemma 22. *Let query f' denote the partial EBC sum over 2-paths with end-points $i, j \in N_a \cap V_Y$ and intermediate node $k \in (N_a \cap V_Y) \cup R \cup \{a\}$. Then the L_1-global sensitivity of f' is upper-bounded by $\Delta f' = |N_a \cap V_Y| - 1$.*

Proof. There are two ways the addition or removal of an edge can affect S_Y. If the edge is the one between endpoints i and j, then this can change the $1/A_n^2(i, j)$ term by at most 1, (from 0 to 1, in the case that the only other connection between i and j is via a). If the edge is within $N_a \cap V_Y$, then it can affect a $1/A_n^2(i, j)$ term by at most $1/2$: the denominator is incremented/decremented by 1 while the denominator must always be at least 1 as a 2-path must go through a. This can occur for at most $2 \cdot (|N_a \cap V_Y| - 2)$ terms in the sum, because they're paths involving some intermediate node in Y that is neither i nor j. So overall the change in S_Y resulting from the addition or removal of one edge is at most:

$$\Delta f' \;=\; 1 + \frac{2 \cdot (|N_a \cap V_Y| - 2)}{2} = |N_a \cap V_Y| - 1.$$

□

As both applications of the Laplace mechanism run with privacy budget $\epsilon/2$, Lemma 5 implies overall E_Y edge privacy is guaranteed.

Corollary 23. BACKWARDMESSAGE *(Algorithm 4.4) takes time and space*

$$O(\max\{|R|, N\} \cdot N)$$

where $N = |N_a \cap V_Y|$*, and when run with* $\Delta_1 = 2|R|, \Delta_2 = N - 1$*, preserves* ϵ*-differential privacy of edge set* E_Y *within party* Y*'s network.*

4.4.3 PRIVATEEBC: Putting it All Together

After parties X and Y have respectively run FORWARDMESSAGE and BACKWARDMESSAGE, X must complete the computation of the private EBC. As shown in Algorithm 4.5, PRIVATEEBC comprises two phases that closely mirror the two components of BACKWARDMESSAGE: counting 2-paths spanning X, Y, and counting 2-paths with endpoints in X.

Within the first stage we incorporate BACKWARDMESSAGE noisy counts T_{ij} contributed by Y, which count paths having intermediate nodes in Y. Party X simply increments these values with counts of paths having endpoints in X. The sum reciprocals forms S_{XY}. *We make one optimisation to utility at no cost to privacy: counts* T_{ij} *for* $i \in R \backslash R^\star$ *are discarded.*

A straightforward sum of paths with endpoints in X and intermediate points in $N_a \cup \{a\}$ completes S_X. Finally party X completes PRIVATEEBC by summing the partial EBCs.

Remark 24. *We opt to use the same* ϵ *privacy budget for both parties* X *(Theorem 20) and* Y *(Corollary 23) in* PRIVATEEBC *out of symmetry. However the algorithm can operate with separate budgets if desired.*

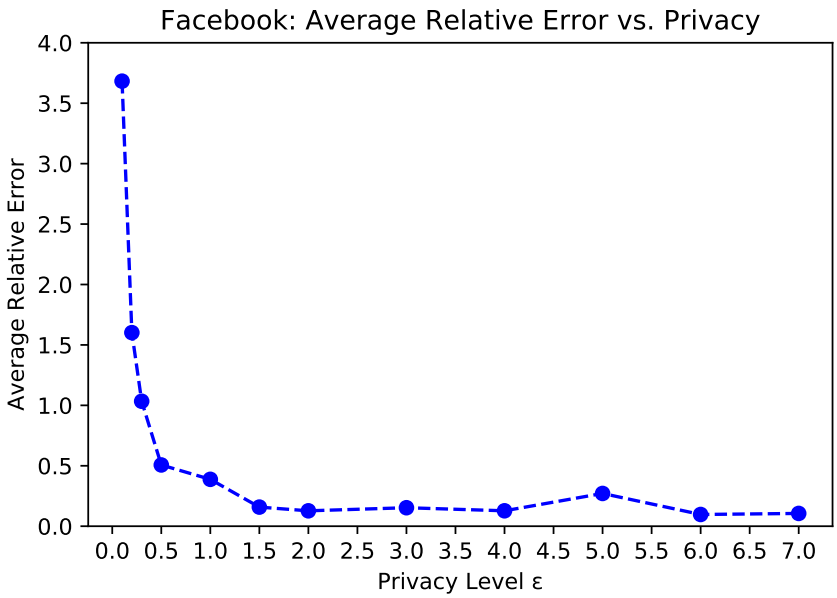


Figure 4.2: Average relative error of the 60 random nodes with $\epsilon = 0.1$ to 7, Facebook dataset.

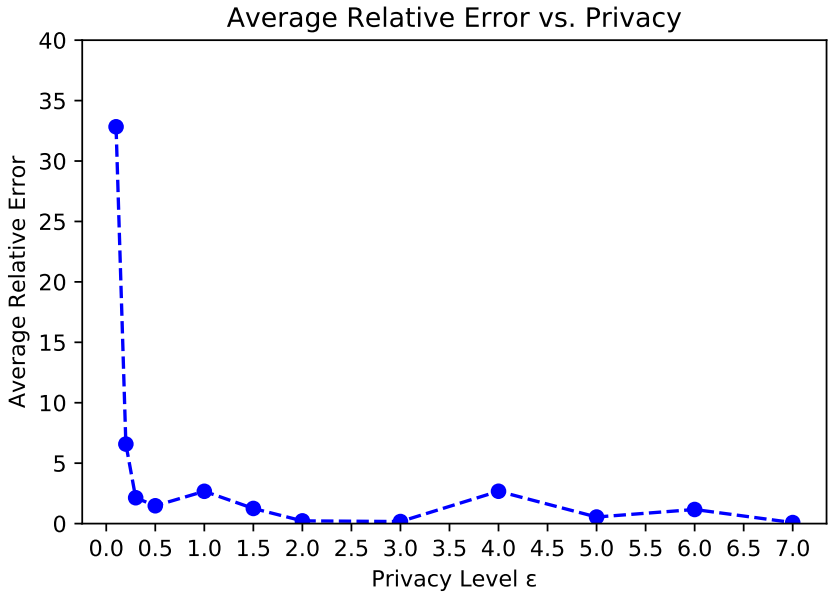


Figure 4.3: Average relative error of 60 random nodes with $\epsilon = 0.1$ to 7, Enron dataset.

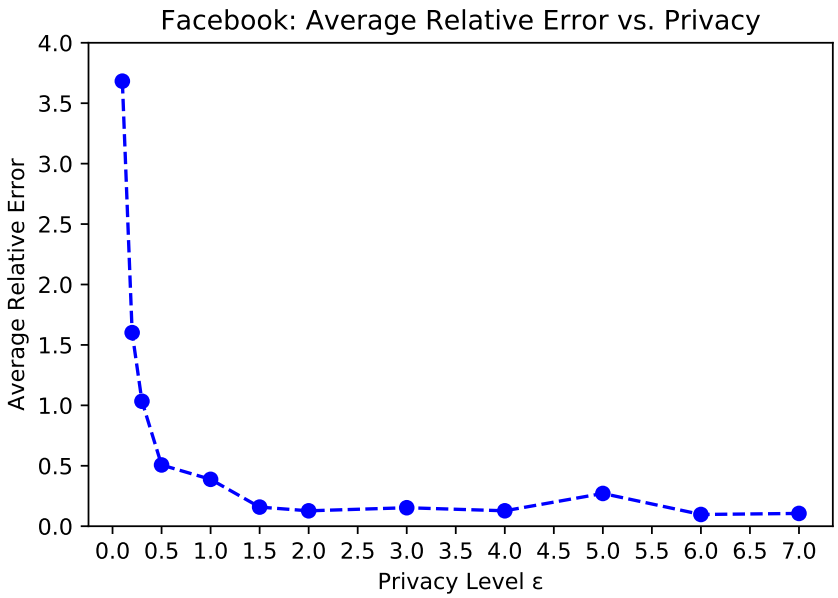


Figure 4.4: Average relative error of 60 random nodes with $\epsilon = 0.1$ to 7, PGP dataset.

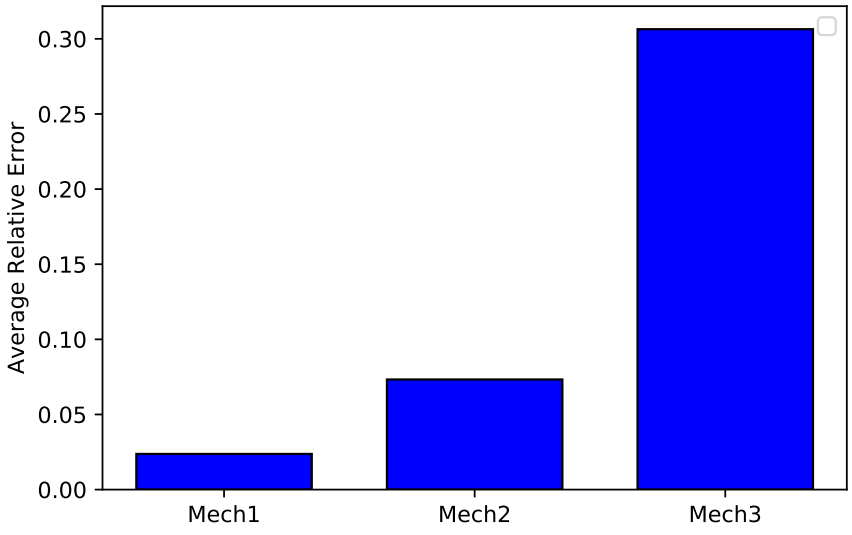


Figure 4.5: Average relative error of 60 random nodes in three different mechanisms, $\epsilon = 1$, Facebook dataset.

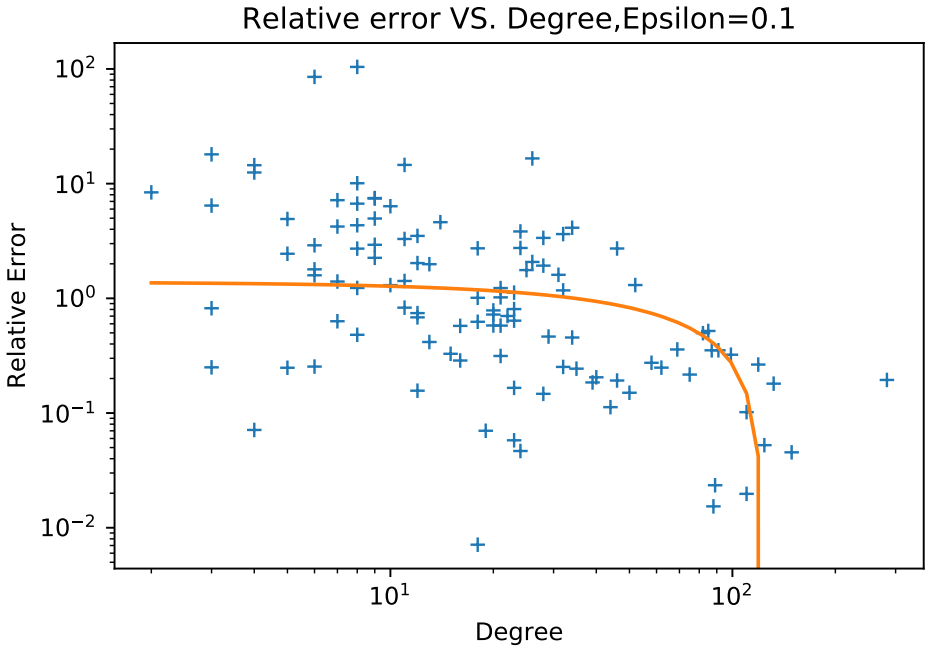


Figure 4.6: Relative error of 100 random nodes with different degrees for $\epsilon = 1$, Facebook dataset.

4.5 Experiments

To empirically validate the effectiveness of PRIVATEEBC we ran experiments on three graph datasets: a Facebook friendship graph [52] with 63,731 vertices and 817,035 edges; the Enron email network [94] with 36,692 nodes and 183,831 edges; and the Pretty Good Privacy (PGP) [52] dataset with 10,680 users as vertices and 24,316 inter-user interactions as edges. We follow a random process to partition the nodes, while the structure of the graph stays intact: nodes are assigned to parties X or Y independently and uniformly at random, while edges are not changed.

The experiments were run on a server with 2×28 core Xeon's (112 threads with hyper threading) and 1.5 TB RAM, using Python 3.7 without parallel computations for fair comparison. We use relative error between true and private EBC—the lower the relative error the higher the utility. We employed the `Mpmath` arbitrary precision library and set the precision to 300 bits. Arbitrary precision is vital for implementing inverse transform sampling as described in Section 4.4.1.4.

4.6 Results

We first examine the relationship between utility and privacy for PRIVATEEBC. For 60 uniformly-at-random ego nodes selected from randomly-partitioned party X, we report average relative error (comparing private and true EBC) for a range of privacy levels ϵ between 0.1 and 7. Figure 4.2—4.4 show the results for the three datasets, where it is apparent that average relative error decreases dramatically when ϵ is increased to 1, and stays very small for larger ϵ. For $\epsilon > 0.5$, average relative error is usually below 50%. And at the strong guarantee of $\epsilon = 1.5$ the average relative error is 16% (Facebook) 47% (Enron) and 25% (PGP).

As we have employed three different privacy-preserving mechanisms in our proposed protocol—one exponential (Mech1) and two Laplace (Mech2, Mech3)—we examine each separately to evaluate how they affect overall relative error. Specifically, we run the PRIVATEEBC protocol with only one of the privacy-preserving mechanisms intact and use the non-private version for remaining

mechanisms, with each of Mech1–Mech3 taking turns being private. In this way we can isolate the incremental cost to utility of each mechanism. Figure 4.5 reports the results on Facebook, which demonstrate that Mech1 FORWARDMESSAGE has the least impact on the relative error while Mech3 BACKWARDMESSAGE second component, has the highest impact. This suggest future work may focus on the third mechanism.

We next report on timing analysis for PRIVATEEBC as function of privacy level. Median computation time of 20 random ego nodes for ϵ from 0.1 to 7 is reported in Figure 4.9 on Facebook data. Here total time is overall decreasing as privacy decreases (increasing ϵ), while a small increase to runtime can be seen at very high levels of privacy (low but increasing ϵ). This dual effect is slightly more pronounced on Enron and PGP 4.7, 4.8, and is likely due to different behaviours in the protocol with increasing ϵ. When the set difference of R and $R\star$ is small, the two-stage sampler generates just small numbers of nodes in faster time. However faster runtime with lower privacy dominates behaviour overall. Moreover any effect of privacy is not strong, with at most a $5\times$ change in runtime which across datasets is practical at under 10 min (median) on the larger datasets.

Figure 4.6 shows how the relative error between true and private EBC varies by ego node degree. We report results on $\epsilon = 1$, which do not show significant dependence: for node degrees up to 10^2, deviation is approximately 7% of the maximum relative error which is low.

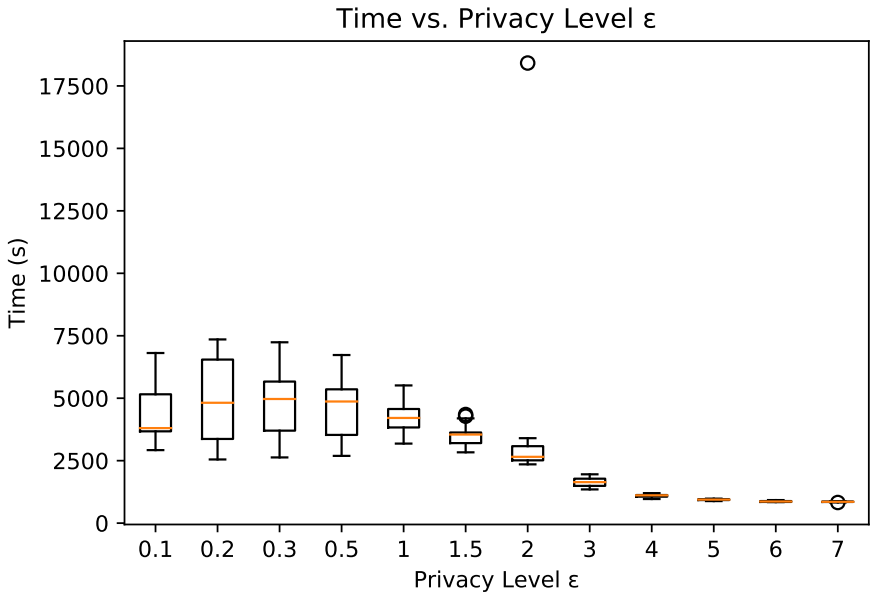


Figure 4.7: Time of computing 20 random nodes with $\epsilon = 0.1$ to 7, Enron dataset.

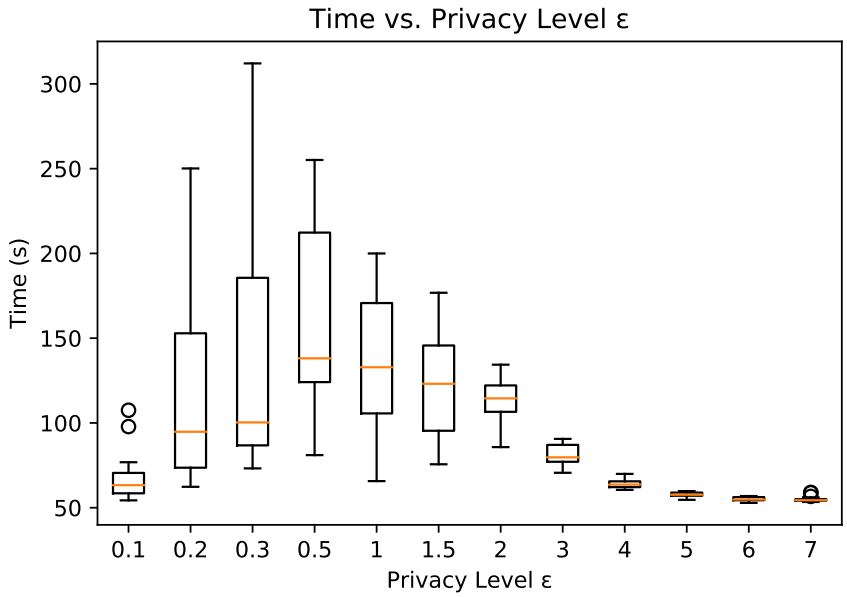


Figure 4.8: Time of computing 20 random nodes with $\epsilon = 0.1$ to 7, PGP dataset.

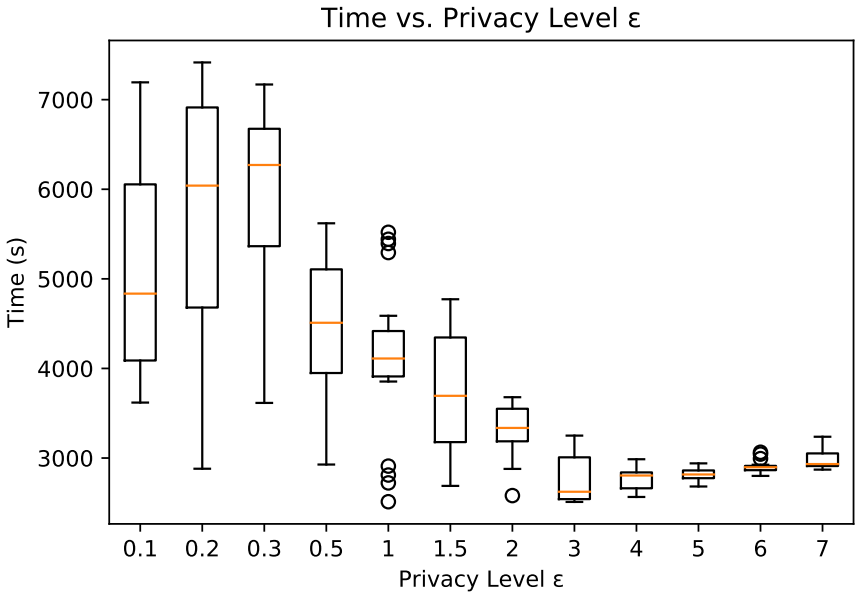


Figure 4.9: Time of computing 20 random nodes with $\epsilon = 0.1$ to 7, Facebook dataset.

4.7 Concluding Remarks

In this chapter, we have developed the PRIVATEEBC algorithm which comprises a protocol of differentially private mechanisms for cooperative 2-party computation of egocentric betweenness centrality. Theoretical and empirical results demonstrate that our approach achieves strong privacy guarantees for both parties which achieving practical levels of utility with efficient time and space complexity. Notably we contribute a novel two-stage sampler that improves upon the exponential mechanism's time and space complexities exponentially. PRIVATEEBC should extend naturally to multiple networks—we expect to add to our empirical investigations of efficiency in that case. It would be interesting to extend differential privacy to the case in which the answer needs to be returned *by* the party whose node is being queried to some untrusted authority.

Chapter 5

Differentially Private Multi-Party Egocentric Betweenness Centrality

In this chapter[1], we show for the first time how multiple mutually-distrusting parties can successfully compute node EBC while revealing only differentially private information about their internal network connections. This protocol presents better computation and communication complexity compared to the two-party scenario in Chapter 4. Moreover, even any third party is able to receive the answer of the query with edge differential privacy guarantees.

5.1 Introduction

This chapter is concerned with measuring node importance in communication networks: *egocentric betweenness centrality* (EBC) [43] measures to what extent a node's neighbours depend on the node for remaining inter-connected with one another. EBC has emerged as a popular centrality measure and is used widely in practice [66].

As discussed in Chapter 4, EBC computation has many possible applications. In conjunction with methods for identifying fake news [59, 69], EBC could be used to limit its propagation by targeting interventions at those individuals who

[1]This chapter is based on the following article:
Leyla Roohi, Benjamin IP Rubinstein, and Vanessa Teague. Assessing Centrality Without Knowing the Connections. A conference paper, 2019, under review

are most critical in spreading information. There are numerous other applications from identifying influential communicators for advertising, to disrupting criminal networks [33].[2]

Chapter 4 considers the case of only two mutually-distrusting networks collaborating to jointly compute EBC from a partitioned network. But of course modern telecommunications involve numerous different telecom providers, even within one country, while many people communicate between countries with completely different networks. Thus multiple networks are absolutely essential for understanding one person's communication.

Here we extend privacy-preserving techniques for EBC computation to arbitrarily many networks. Relevant information is split over multiple participants. We show that by carefully structuring information flow, we achieve highly accurate results and strong privacy protection, while incurring only moderate total communication complexity. We also improve the privacy model by producing a private output that can be safely published, while the prior approach of Chapter 4 only protects one party's data from the other's.

Once again, our setting involves the complete list of nodes (*i.e.*, people) being public knowledge, while individual connections are to be kept private. Each service provider knows the connections within its own network, plus the connections between one of its members and the outside (*e.g.*, from when they contact someone in a different network). Connections internal to other networks are unknown.

Once again, we prove that our protocol preserves *edge differential privacy* [48] in which the existence or non-existence of an edge must be protected. Our main contributions follow.

1. We develop a protocol for multi-party privacy-preserving computation of EBC.

2. We strengthen the adversarial model in comparison to Chapter 4. All participating networks are protected by edge-DP, even when the final

[2]From [33]: "The [criminal] network gravitates around a few central actors who have relatively more direct connections in the network than the rest of the network. It means that there is a distinction between a core and periphery in the network. This implies that peripheral actors depend on a few central actors for their information and resources flowing through the network."

output is published.

3. We deliver a high-probability utility bound on a core component of our protocol: private distributed release of ego networks used also in Chapter 4.

4. We conduct comprehensive empirical validation of our algorithm using open Facebook, Enron and PGP datasets. These demonstrate broad applicability of our algorithm with reasonable 1.07 relative error at strong $\epsilon = 0.1$ DP, with practical runtimes, and insignificant degradation with increasingly many parties.

Near-constant accuracy with increasing numbers of parties is both surprising and significant, as is our innovation over past work Chapter 4 that prevents leakage at final EBC release.

This protocol is substantially more efficient than application of two-party techniques from Chapter 4, which would involve sending information about each edge between each pair of parties, for a total communication of $O(|V|^2|A|^2)$ where V is the set of vertices and A the parties. Instead, by carefully structuring the information flow, we achieve total communication of $O((|A| + |V|)|A||V|)$.

5.2 Another View of the Two-Party Exponential Mechanism

We use the subset release mechanism from Chapter 4 which leverages the exponential mechanism to release subset of nodes in one party that privately approximates the ego network. We repeat the relevant main result of Chapter 4 here for ease of use. Note that we recast the mechanism here as one of releasing private subsets of a public set. This mechanism may be of independent interest.

Lemma 25. *Consider any publicly-known set V and a privacy-sensitive subset $R^\star \subseteq V$ equivalent to a characteristic function $D \in \{0,1\}^{|V|}$. The exponential mechanism run with quality function $q(R^\star, R) = |R \cap R^\star| + |\overline{R \cup R^\star}|$ and $\Delta q = 1$ preserves ϵ-DP. Furthermore Algorithm 4.1 implements this mechanism via two-staged sampling that runs in $O(|V|)$ space and time which is exponentially faster than a naïve application of the exponential mechanism.*

Table 5.1
Glossary of symbols used in this Chapter.

$A = \{\alpha_1, \ldots, \alpha_{\lvert A\rvert}\}$	The parties *e.g.*, competing service providers.
V_α	The nodes of party α.
$E_{\alpha,\alpha}$	Edges entirely within party α.
$E_{\alpha\beta}$	Edges running between parties α and β.
a	The ego node (element of query party α_1).
N_a	The ego network of a.
X^-	A set of nodes X excluding a.
$R^\star_\alpha$	The ego network contained in V_α.
T^{ij}_α	Counts of 2-paths joining i, j, with intermediate nodes in α
S_α	Partial EBC sum contributed by party α.
R_α	A private, randomised approximation to $R^\star_\alpha$.
Q_i	For $i \in \{0, \ldots, \lvert V^-_\alpha\rvert\}$, a partition of $\mathcal{P}(V^-_\alpha)$.
ϵ	The differential privacy budget.
Δ	A global sensitivity bound.

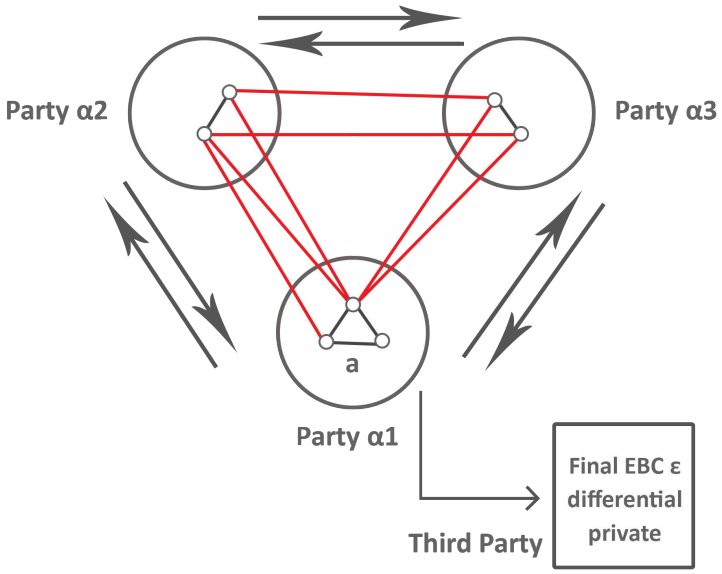


Figure 5.1: Ego network of node 'a' spanning between three parties.

5.3 Problem Statement

We revisit the problem statement of Chapter 4, now with multiple parties—two or more. We have $|A|$ participating parties $A = \{\alpha_1, \ldots, \alpha_{|A|}\}$, each representing a telecommunications service provider. They control a global communication graph (V, E) whose nodes are partitioned into $|A|$ (disjoint) sets one per service provider such that V_α contains the nodes of party α. Every customer is represented as a node that belongs to one and only one service provider; pairs of customers who have had some communication (*e.g.*, a phone call, SMS or email) are edges.

We will often equivalently represent edge sets as adjacency matrices (or flattened vectors) with elements in $\{0, 1\}$. Table 5.1 collects all of the symbols used in this chapter.

We write $E_{\alpha,\alpha}$ for the set of edges in (V, E) between nodes in V_α—these are communications that happened entirely within α. Similarly, $E_{\alpha,\beta}$ are the edges with one node in V_α and the other in V_β—these represent communications between two service providers. Once again, E is the disjoint union of all such edge sets. $E = \bigcup_{\alpha,\beta \in A} E_{\alpha,\beta}$.

We assume that all nodes are known to all parties, but that each party learns only about the edges that are incident to a node in its network, including edges that link two nodes within its network.

We wish to enable all parties to learn the EBC of any chosen node a, while maintaining *edge privacy* between all parties. Without loss of generality we assume $a \in V_{\alpha_1}$.

Before detailing a protocol for accomplishing this task, we must be precise about a privacy model. Figure 5.1 shows the nodes and edges of an ego network of node 'a' for three parties which are spanning between three parties.

Problem 26 (Private Multi-Party EBC)**.** *Consider a simple undirected graph* $(\bigcup_{\alpha \in A} V_\alpha, \bigcup_{\alpha,\beta \in A} E_{\alpha,\beta})$ *partitioned by parties* $A = \{\alpha_1, \ldots, \alpha_{|A|}\}$ *as above, and an arbitrary node* $a \in V_{\alpha_1}$*. The problem of* private multi-party egocentric betweenness centrality *is for the parties* α *to collaboratively approximate* $\mathrm{EBC}(a)$ *under assumptions that:*

A1. All parties $\alpha \in A$ *know the entire node set* $\bigcup_{\alpha \in A} V_\alpha$*;*

A2. Each party knows every edge incident to nodes within their own network. That is, each party α knows $\bigcup_{\beta \in A} E_{\alpha,\beta}$;

A3. The computed approximate $\mathrm{EBC}(a)$ *needs to be available to all of the parties.*

The intermediate computation must protect ϵ-differential edge privacy of each party from the others. We seek solutions under a fully adversarial privacy model: irrespective of whether other parties $\beta \neq \alpha$ follow the protocol, the releases by party α protect its edge differential privacy. (Of course a cheating participant can always release information about edges it already knows, which may join another network.)

Furthermore, the output *must protect ϵ-differential privacy of the edges. In Chapter 4, the final EBC could be revealed just to the party who made the query, however, in this chapter, the final EBC is ϵ-differentially- private and could be sent safely to anyone.*

5.4 Non-Private Multi-Party Protocol

We first consider how multiple parties α (potentially more than two) might cooperate without preserving differential privacy with special attention to efficiency (so as to improve on a application of the two-party protocol of Chapter 4).

A key observation is that no α can itself count 2-paths that are

- contained entirely within other parties; or
- have one end node in α, but the other two nodes in other parties.

The party β that contains a node j can always count the number of 2-step paths through j, but it doesn't know which of the nodes adjacent to j are in the ego-network of a (except in some special cases, such as if a or the adjacent nodes are in β). So in order for β to count the number of 2-paths in a's ego network that pass through j, we require each other network α to communicate its endpoint neighbours of a to β. This significantly complicates the differentially private solution developed in the next section.

Recall that $N_a = \{v \in V \mid \{v, a\} \in E\}$ denotes the ego network of a anywhere in the graph (notably not including a since the graph has no self-loops). Figure 5.2 summarises the following protocol.

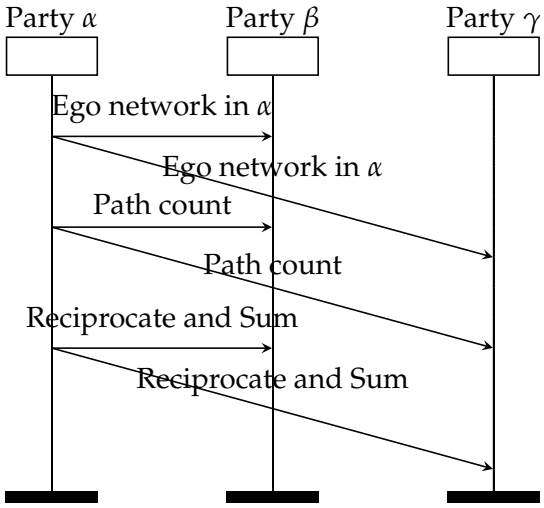


Figure 5.2: EBC multi-party computation protocol for party α, comprising three messages per party. Visualised for three parties.

Protocol 27. *Every party proceeding in sequence:*

i. [EgoNetwork] α broadcasts to every party the set $R^\star_\alpha$ of neighbours of a contained within α;

ii. [PathCount] For all nodes $i, j \in \bigcup_\beta R^\star_\beta$ such that $i < j$, α computes $T^{i,j}_\alpha$, the number of 2-paths from i to j where the intermediate point $k \in R^\star_\alpha$ (irrespective of whether i, j are directly connected).

Party α sends $T^{i,j}_\alpha$ to the β such that $i \in V_\beta$.

iii. [ReciprocateAndSum] For every $i \in R^\star_\alpha$ and all $j > i$, α computes the total number of 2-paths between i, j provided these nodes are disconnected: it sums $T^{i,j}_\beta$ for all $\beta \in A$. It then sets S_α to be the reciprocal of this sum and broadcasts this value to all parties;

iv. α completes the computation of $\mathrm{EBC}(a)$ *as* $\sum_{\alpha \in A} S_\alpha$.

5.4.1 Privacy Disclosure

Now consider how the privacy of edges can be compromised in the first three steps of Protocol 27.

i. When $R_\alpha^\star$ is broadcast in Step i, other parties learn directly of all edges incident to a in α.

ii. When T_α^{ij} is sent to $\hat{\beta}$ in Step ii, it reveals information about edges outside $\hat{\beta}$. A worst case occurs for node i when there is only one node $k \in \alpha$ connected to it. Then T_α^{ij} reveals the existence of edge j, k for all $j > i$.

iii. When α broadcasts S_α, it reveals the connection status of edges within α. In the worst case when there is just two nodes i and j in α, an edge between them can change S_α from a non zero value to zero.

Communication complexity Step i of Protocol 27 requires each party to send to each other party $|V|$ bits of length 1 that shows the node is present or not, hence a total of $O(|A|^2|V|)$. Step ii sends, for each node i, up to $|V|$ messages $T^{i,j}$ from each party to the owner of node i. The pathcounts T^{ij} are at most $|V|$, so the total size is $O(|A||V|^2)$. Finally, Step iii requires every participant to send each other one message: $O(|A|^2)$. Hence the total communication complexity is $O((|A| + |V|)|A||V|)$.

5.5 Multi-Party Private EBC

In this section, we describe three algorithms—SUBSETRELEASE, PRIVATEPATH-COUNT, PRIVATERECIPROCATEANDSUM—in order to implement PRIVATEEBC, a differentially private version of the protocol described in Section 5.4.

5.5.1 Private Egonetwork Broadcast

Each party α runs SUBSETRELEASE, our name for Algorithm 4.1 in this chapter with its share of the egonetwork $R_\alpha^\star$ in V_α. Then each party broadcasts the resulting R_α—the approximation of $R_\alpha^\star$— to all other parties.

SUBSETRELEASE uses the exponential mechanism to privately optimise a particular quality function that encourages a large intersection between $R_\alpha^\star$ and release R_α, along with a minimal symmetric set difference. As each party runs this mechanism relative to its own node set, it operates its own quality function

defined relative to $R^\star_\alpha$ (see the proposition below for the formal definition). We observe a convenient property of the quality functions run by each party: they sum up to the overall quality function if the ego party were to run SUBSETRELEASE in totality. This permits us to prove that this simple distributed protocol for private ego network approximation exactly implements a centralised approximation, as made formal by the following result. *There is no loss to privacy or accuracy due to decentralisation.*

Proposition 28. *Consider parties $\alpha \in A$ running* SUBSETRELEASE *with identical budgets $\epsilon_1 > 0$ and quality functions $q_\alpha(R) = |R \cap R^\star \cap V_\alpha| + |\overline{R \cup R^\star} \cap V_\alpha|$, on their disjoint shares $R^\star_\alpha = R^\star \cap V_\alpha$ to produce disjoint private responses $R_\alpha \subseteq V_\alpha$. Then $R = \cup_{\alpha \in A} R_\alpha$ is distributed as* SUBSETRELEASE *run with ϵ, quality function $q(\cdot)$, on the combined $R^\star$ in X. Consequently the individual R_α and the combined R, each preserve ϵ_1-DP simultaneously.*

Proof. Observe that since the V_α form a disjoint partition of X, that for any $R_\alpha \subseteq V_\alpha$ we have that

$$\sum_{\alpha \in A} q_\alpha(R_\alpha) = \quad q\left(\bigcup_{\alpha \in A} R_\alpha\right) \quad . \tag{5.1}$$

Combining this with the independence of the concurrent executions of SUBSETRELEASE, we have that the joint density over their releases corresponds to

$$\begin{aligned} p(r) &= \prod_{\alpha \in A} p(r_\alpha) \\ &\propto \prod_{\alpha \in A} \exp\left(q_\alpha(r_\alpha) \cdot \epsilon_1\right) \\ &= \exp\left(\epsilon_1 \sum_{\alpha \in A} q_\alpha(r_\alpha)\right) \\ &= \exp\left(q(r) \cdot \epsilon_1\right) \quad . \end{aligned} \tag{5.2}$$

Due to uniqueness of probability density normalisation, this proves the result. □

Algorithm 5.1 PRIVATEPATHCOUNT

Input: ego node $a \in V_{\alpha_1}$; execution party $\alpha \in A$; true node set $R^\star_\alpha$; for each $\beta \in A$, edge set $E_{\alpha,\beta}$ and private node set R_β; $\epsilon_2, \Delta_2 > 0$

Ensure: A vector of noisy counts, indexed by endpoints $\{i, j\}$ with $i < j$, of the total number of nodes k in $R^\star_\alpha$ that are connected to both i and j.

if $\alpha = \alpha_1$ **then**
  $R^\star_\alpha \longleftarrow R^\star_\alpha \cup \{a\}$
end if
$R_A \longleftarrow \bigcup_{\beta \in A} R_\beta$
for $i \in R_A$ **do**
  for $j \in R_A$ with $i < j$ **do**
    $K \longleftarrow \left\{k \in R^\star_\alpha \mid \{i,k\}, \{k,j\} \in \bigcup_{\beta \in A} E_{\alpha,\beta}\right\}$
    $T^{ij}_\alpha \longleftarrow |K| + \mathrm{Lap}(2\Delta_2/\epsilon_2)$
  end for
end for
return $\mathbf{T_{ff}}$

5.5.2 Private Path Count

After every party α runs SUBSETRELEASE and broadcasts R_α then each party runs Algorithm 5.1, using the R_α's received. In this phase every party counts all the two paths where the intermediate node is in R_α. For each node pair (i, j) with $i < j$, α sends the 2-path count $T^{i,j}_\alpha$ to the party that contains node i—so far, this mirrors the non-private version of the protocol described in Section 5.4.

In order to privatise this vector of counts, Laplace noise is added to the two-path counts according to the sensitivity in the following lemma; thereby preserving ϵ_2-DP in the stage's release.

Lemma 29. *Let query f denote the vector-valued non-private response $\mathbf{T}_\alpha$ of party α in Algorithm 5.1. The L_1-global sensitivity of f is upper-bounded by $\Delta f = 2|R_A|$.*

Proof. Algorithm PRIVATEPATHCOUNT is executed by all parties; the output of these computations is broadcast to other parties. As we need to preserve the privacy of each party's edges individually, we consider one (arbitrary) party $\alpha \in A$. The output of α is a vector of the counts of all 2-paths connecting $i, j \in R_A = \bigcup_{\beta \in A} R_\beta$ with intermediate node $k \in R^\star_\alpha$. Adding or removing an edge from $E_\alpha = \bigcup_{\beta \in A} E_{\alpha,\beta}$, can worst-case change $2|R_A|$ elements of α's counts

Algorithm 5.2 PRIVATERECIPROCATEANDSUM

Input: ego node $a \in V_{\alpha_1}$; execution party $\alpha \in A$; for each $\alpha \leq \beta \in A$, edge set $E_{\alpha,\beta}$ and private node set R_β; for each $\beta \in A$, noisy counts $\mathbf{T}_\beta$; $\epsilon_3, \Delta_3 > 0$

$R \longleftarrow \bigcup_{\beta>\alpha} R_\beta \cup R^\star_\alpha$
$E_\alpha \longleftarrow \bigcup_{\beta\geq\alpha} E_{\alpha,\beta}$
$S_\alpha \longleftarrow 0$
for $i \in R^\star_\alpha$ **do**
  for $j \in R$ with $i < j$ **do**
    if $\{i, j\} \notin E_\alpha$ **then**
      $T \longleftarrow \sum_{\gamma\in A} T^{ij}_\gamma$
      $S_\alpha \longleftarrow S_\alpha + (\lfloor \max\{0, T\} \rfloor + 1)^{-1}$
    end if
  end for
end for
$S_\alpha \longleftarrow S_\alpha + \mathrm{Lap}(2\Delta_3/\epsilon_3)$
return $\mathbf{S_{ff}}$

by one each. To see this, consider a very highly-connected node k and the deletion of $\{i, k\}$ for any i: this reduces the counts for paths joining i, j for all $j \in R_A$. Likewise if i were also within party α and were highly connected then the edge removal would also reduce counts for paths joining k, j for all $j \in R_A$. This proves that the sensitivity for part α running PRIVATEPATHCOUNT is $2|R_A|$ irrespective of party α. □

5.5.3 Private Reciprocate and Sum

Every party α receives noisy counts from PRIVATEPATHCOUNT and for any pairs of i and j in party α where $i \in R^\star_\alpha$ and $j \in \bigcup_{\beta>\alpha} R_\beta \cup R^\star_\alpha$, α increments the received $T^{i,j}$ by the number of 2-paths that are not directly connected. Each party then reciprocates the summation of the counts. In this algorithm, each party my replace noisy R_α with true $R^\star_\alpha$. This optimises utility at no cost to privacy: counts T^{ij} for $i, j \in R_\alpha \backslash R^\star_\alpha$ are discarded. This is safe to do, since the Laplace mechanism already accounts for changes in $R^\star_\alpha$. The Laplace noise is utilised to privatise the reciprocated sum S_α to ϵ_3-DP, calibrated by sensitivity bounded next.

Lemma 30. *Let query f' denote the reciprocate and sum over 2-paths with intermediate*

Algorithm 5.3 PRIVATEEBC in Multi-Party

Input: (Public) ego node $a \in V_{\alpha_1}$; ordered set of parties A; node sets V_α for $\alpha \in A$; parameter vectors $\epsilon, \Delta \succ 0$.
Input: (Private) for each $\alpha, \beta \in A$, edges $E_{\alpha,\beta}$, nodes $R^\star_\alpha$;

1: **for** $\alpha \in A$ in parallel **do**
2: Party α does:
3: **if** $\alpha = \alpha_1$ **then**
4: $V = V_\alpha \backslash \{a\}$
5: **else**
6: $V = V_\alpha$
7: **end if**
8: $R_\alpha \leftarrow$ SUBSETRELEASE$(V, R^\star_\alpha, \epsilon_1)$
9: Broadcast R_α
10: $\mathbf{T_{ff}} \leftarrow$ PRIVATEPATHCOUNT$(\alpha, E_{\alpha\beta}, R_\beta, \epsilon_2, \Delta_2)$
11: **for all** $i, j \in V$ with $i < j$ **do**
12: Send $T^{i,j}_\alpha$ to Party β s.t. $i \in V_\beta$
13: **end for**
14: $S_\alpha \leftarrow$ RECIPROCATEANDSUM$(a, \{E_{\alpha,\beta}, R_\beta | \beta > \alpha\}, \epsilon_3)$. {Party α reciprocates and sums only paths with $\beta \geq \alpha$.}
15: Broadcast S_α
16: pEBC$(a) \leftarrow \sum_{\alpha \in A} S_\alpha$
17: Return pEBC(a)
18: **end for**

point in $\bigcup_\beta R_\beta$ *while the* i *and* j *are not connected and* $i \in R^\star_\alpha$ *and* $j \in \bigcup_{\beta<\alpha} R_\beta \cup R^\star_\alpha$. *Then the* L_1*-global sensitivity of* f' *is upper-bounded by* $\Delta f' = (\lfloor \max\{0, T\} \rfloor + 1)^{-1} \leq 1$ *irrespective of party.*

Proof. Once again we apply post-processing to previously sanitized outputs R_β, T_β. Consider any party $\alpha \in A$ and the effect of removing/adding an edge incident to a node of α. At worst this will result in the condition $\{i, j\} \notin E_\alpha$ of line 6 evaluating differently—for at most one pair $\{i, j\}$. That is, the non-private sum of reciprocals S_α can be affected by the addition/removal of a single term $(\lfloor \max\{0, T\} \rfloor + 1)^{-1}$. Such a term is upper bounded by the reciprocal of lower bound on $\lfloor \max\{0, T\} \rfloor + 1 \geq 1$. That is, the sensitivity, irrespective of executing party α, is 1. □

Communication complexity The communication complexity of the private version is the same as that of the non-private one: $O((|A| + |V|)|A||V|)$.

5.5.4 PRIVATEEBC: Putting it All Together

After the parties have each run the protocol phases together, namely SUBSETRELEASE, PRIVATEPATHCOUNT and PRIVATERECIPROCATEANDSUM, the parties must finally complete the computation of the private EBC. Algorithm 5.3 depicts PRIVATEEBC orchestrating the high-level protocol thus far, and then adding the received S_α to compute final EBC.

Theorem 31. PRIVATEEBC *preserves* $(\epsilon_1 + \epsilon_2 + \epsilon_3)$*-DP for each party.*

Remark 32. *While we have used uniform privacy budgets across parties, our analysis extends to custom party budgets.* ϵ *privacy budget for all parties* α *in* PRIVATEEBC *out of symmetry. However the algorithm can operate with separate budgets if desired.*

5.6 Utility Bound

In this section we develop a utility analysis of privacy-preserving betweenness centrality. Our analysis focuses on a utility bound on EBC resulting from the subset release mechanism first introduced in Chapter 4 run to privatise the ego network.

We abuse notation with $q(R) = q(R^\star, R)$ referring to the quality function of the SUBSETRELEASE mechanism of Lemma 25 with dependence on the private $R^\star$ made implicit; likewise for the quality functions run by each party in the decentralised setting. The technical challenge is in leveraging the following well-known utility bound on the exponential mechanism, which only establishes high-probability near-optimal quality.

Lemma 33 (Lemma 7 of [67])**.** *Consider a centralised party running* SUBSETRELEASE *with budget* $\epsilon > 0$*, quality function* $q(\cdot)$ *on* $R^\star$*. For* $t > 0$ *let:*

$$S_t = \{R : q(R) > q(R^\star) - t\}$$

and let μ be the uniform probability mass function on $\mathcal{P}(X^{\text{-}})$. Then:

$$\Pr(\overline{S}_{2t}) \leq \exp(-\epsilon t)\mu(\overline{S}_{2t})/\mu(S_t)$$

.

Corollary 34. *Consider parties $\alpha \in A$ each running* SUBSETRELEASE *concurrently with budgets $\epsilon > 0$ and quality functions $q_\alpha(R) = |R \cap R^\star \cap V_\alpha| + |\overline{R \cup R^\star} \cap V_\alpha|$ on their disjoint shares $R^\star_\alpha = R^\star \cap V_\alpha$ to produce disjoint responses $R_\alpha \subseteq V_\alpha$. Then the consequent high-probability quality bound of Lemma 33 holds for random combined response $R = \bigcup R_\alpha$.*

Proof. The claim follows directly from Lemma 33 combined with Proposition 28. □

Our first step is relating EBC error on a released R to the quality $q(R)$. We organise differences in EBC by reciprocal 2-path count terms, enumerating shared and unshared such terms between private and non-private EBCs. As these terms and their differences are bounded by one, the task reduces to measuring differences in egonetwork cardinalities. This is also the goal of our quality score function.

Lemma 35. *For any $R \subseteq X^{\text{-}}$ we may bound the additive EBC error resulting from using R instead of non-private $R^\star$ according to the quality function applied to the random release:*

$$|EBC - EBC_1| \leq \frac{1}{2}(|X| - q(R) + |R^\star|)^2$$

Proof. Let T^{ij}, T_1^{ij} denote the number of 2-paths connecting i, j within node-sets $R^\star$ and R respectively. Our goal is to upper bound the quantity:

$$\begin{aligned}
&\left| \sum_{i,j \in R^\star} \frac{1\left[\{i,j\} \in E\right]}{T^{ij}} - \sum_{i,j \in R} \frac{1\left[\{i,j\} \in E\right]}{T_1^{ij}} \right| \\
\leq &\sum_{i,j \in R^\star \cap R} \left| \frac{1\left[\{i,j\} \in E\right]}{T^{ij}} - \frac{1\left[\{i,j\} \in E\right]}{T_1^{ij}} \right| \\
&+ \sum_{i \in R \setminus R^\star, j \in R} \left| \frac{1\left[\{i,j\} \in E\right]}{T_1^{ij}} \right|
\end{aligned}$$

$$+ \sum_{i \in R^\star \setminus R, j \in R^\star} \left| \frac{1\left[\{i,j\} \in E\right]}{T^{ij}} \right| \ ,$$

following from the triangle inequality and collection of terms with shared end-point nodes i, j. By cases: only when both end points are elements of the intersection $R \cap R^\star$ is there a pair of matching EBC terms. Otherwise at least one (or both) end-point nodes sit outside R or $R^\star$ respectively—in which case there is only one EBC term for the corresponding node set.

We can see that both types of summand are bounded above by unity. For in the second case, for any $i, j \in R^\star$,

$$\left| \frac{1\left[\{i,j\} \in E\right]}{T^{ij}} \right| \ \leq \ 1 \ ,$$

since $T^{ij} \geq 1$ since there is always a path through egonode a. The same holds for the case of $i, j \in R$ by definition. And in the first case of $i, j \in R \cap R^\star$, we have that

$$\begin{aligned} & \left| \frac{1\left[\{i,j\} \in E\right]}{T^{ij}} - \frac{1\left[\{i,j\} \in E\right]}{T_1^{ij}} \right| \\ \leq \ & 1 - \frac{1}{\max(\{|R^\star|, |R|\}) - 2} \\ < \ & 1 \ . \end{aligned}$$

Therefore it follows that

$$\begin{aligned} & \left| \sum_{i,j \in R^\star} \frac{1\left[\{i,j\} \in E\right]}{T^{ij}} - \sum_{i,j \in R} \frac{1\left[\{i,j\} \in E\right]}{T_1^{ij}} \right| \\ \leq \ & |\{i,j : i,j \in R \cap R^\star\}| \\ & + \ |\{i,j : i \in R \setminus R^\star, j \in R\}| \\ & + \ |\{i,j : i \in R^\star \setminus R, j \in R^\star\}| \\ = \ & \binom{|R \cup R^\star|}{2} - |R^\star \setminus R||R \setminus R^\star| \\ \leq \ & |R \cup R^\star|^2/2 \\ \leq \ & \left(|R \setminus R^\star| + |R^\star \setminus R| + |R^\star|\right)^2/2 \end{aligned}$$

$$= \quad (|X| - q(R) + |R^\star|)^2 / 2 \ ,$$

where the first equality follows by enumerating the i, j pairs being counted as all those within R or $R^\star$ provided that both nodes do not reside in separate set differences $R \backslash R^\star$ and $R^\star \backslash R$. This completes the result.

We may now use this lemma with our lifted exponential utility bound Proposition 28 to directly bound EBC error. Note that the previous lemma is agnostic to the number of parties—the released R might be produced in its entirety by the ego node's party through a single call to SUBSETRELEASE, or it could be the disjoint union of multiple calls to SUBSETRELEASE by each party in cooperation. Likewise our lifted bound on high-probability quality also holds as if R is produced centrally. As such the proof of the following result may proceed as if this is the case. □

Theorem 36. *Consider privacy budget $\epsilon > 0$, true ego network $R^\star$ and $t > |R^\star|^2/2$. And suppose that each party $\alpha \in A$ runs* SUBSETRELEASE *with budget ϵ, quality function $q_\alpha(R) = |R \cap R^\star \cap V_\alpha| + |\overline{R \cup R^\star} \cap V_\alpha|$, on their disjoint share $R^\star_\alpha = R^\star \cap V_\alpha$ to produce disjoint private response $R_\alpha \subseteq V_\alpha$. Then EBC_1 produced from $R = \cup_{\alpha \in A} R_\alpha$ incurs error relative to non-private EBC run on non-private $R^\star$, upper bounded as*

$$|EBC - EBC_1| \leq t$$

with probability at least

$$1 - \exp\left(-\epsilon(\sqrt{2t} - |R^\star|)/2\right) 2^{|X^-|} \ .$$

Proof. We begin by defining two events of interest on $R \in \mathcal{P}(V_\alpha \backslash \{a\})$ for any $s > 0$ to be chosen later: S_s that our (centralised by Proposition 28) exponential mechanism achieves near-optimality, and T_s that the resulting EBC is near optimal. Rewriting the event S_s defined in Lemma 33, using $\tilde{q}(R) = |X| - q(R)$ and $\tilde{q}(R^\star) = 0$, we have that

$$S_s = \{R : \tilde{q}(R) \leq s\} \ ,$$

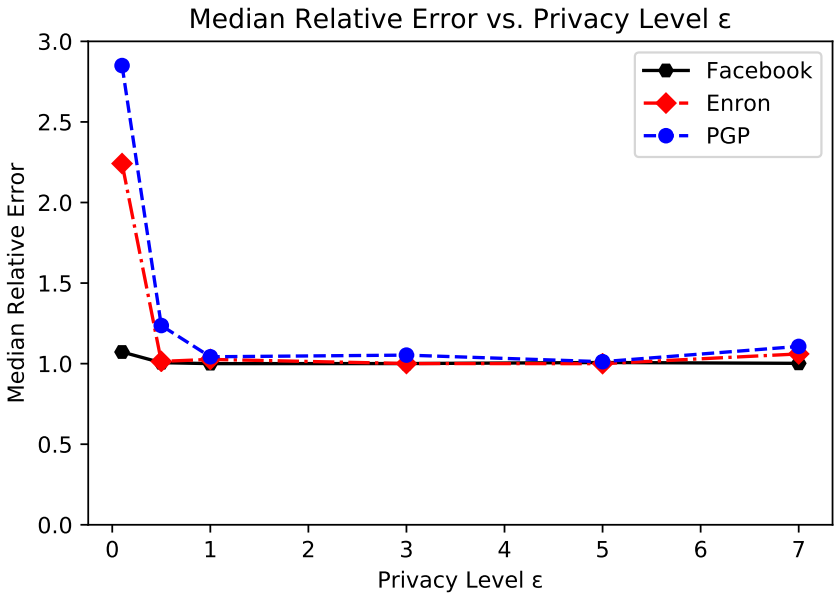


Figure 5.3: Median relative error of the 60 random nodes with ϵ =0.1 to 7, Facebook, Enron and PGP dataset, for three parties.

which holds w.p. at least $1 - \exp(-\epsilon s/2)2^{-|X^-|}$. Next define

$$T_s = \left\{ R : |EBC - EBC_1| \leq \frac{1}{2}(s + |R^\star|)^2 \right\} \quad .$$

By Lemma 35 we have that $|EBC - EBC_1| \leq \frac{1}{2}(\tilde{q}(R) + |R^\star|)^2$. And since $(a + b)^2/2$ is increasing in $a \in \mathbb{R}$ for $b \geq 0$, event S_s implies then that $|EBC - EBC_1| \leq \frac{1}{2}(s + |R^\star|)^2$ and so $S_s \subseteq T_s$ and $\Pr(T_s) \geq \Pr(S_s)$. Provided that $t > |R^\star|^2/2$, taking $s = \sqrt{2t} - |R^\star| > 0$ this completes the result. □

Remark 37. *We now examine whether the bound of Theorem 36 can make meaningful predictions* (i.e., *is non-vacuous). Consider a very sparse graph on nodes* $|X| > 10$, *and consider a true ego network* $R^\star$ *with cardinality represented as a fraction* $\alpha \in [0, 1]$ *of* $|X|$ i.e., $|R^\star| = \alpha|X|$. *Let us now invoke the theorem with error bound* t *taken to be a multiplier* $\gamma > 1$ *of EBC—yielding a relative error bound of* $|EBC - EBC_1|/EBC \leq \gamma$. *For very sparse graphs, EBC can arbitrarily approach* $\binom{|R^\star|}{2} \leq |R^\star|^2/2$ *for* γ *bounded away from zero and large* $|X|$. *Thus setting* $t = \gamma|R^\star|^2/2$ *corresponds to relative error* γ

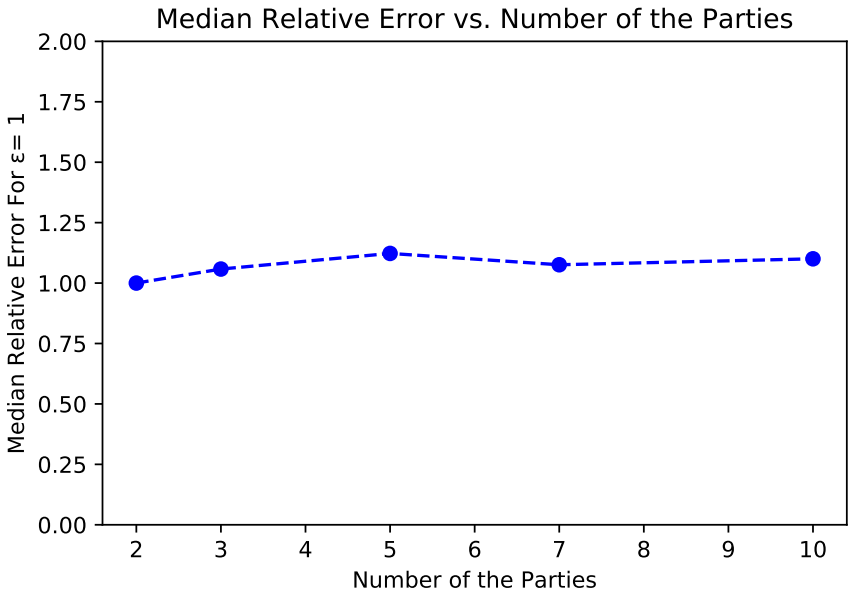


Figure 5.4: Median Relative error of 120 nodes with $\epsilon = 1$, for different number of parties for PGP.

which is meaningful for γ not much larger than 1 (corresponding to private EBC within the same order of magnitude as non-private EBC). Next we wish to set the confidence $1 - \delta$ to be very close to unity e.g., *$\delta = 2^{-10}$ corresponds to confidence exceeding 99.9%. With these choices, we set the confidence in Theorem 36 to $1 - \delta$ and solve for the privacy budget required for* SUBSETRELEASE*:*

$$\epsilon \geq \frac{3 \log_e 2}{(\gamma - 1)\alpha} \ .$$

For example ***a modest $\epsilon = 2.1$ budget is sufficient to guarantee reasonable relative error 3 w.h.p 0.999*** *for a large ego network spanning half an (otherwise sparse) graph. Similar levels of relative error (for end-to-end private EBC) at similar privacy budgets are observed in experiments on real, non-sparse networks below.*

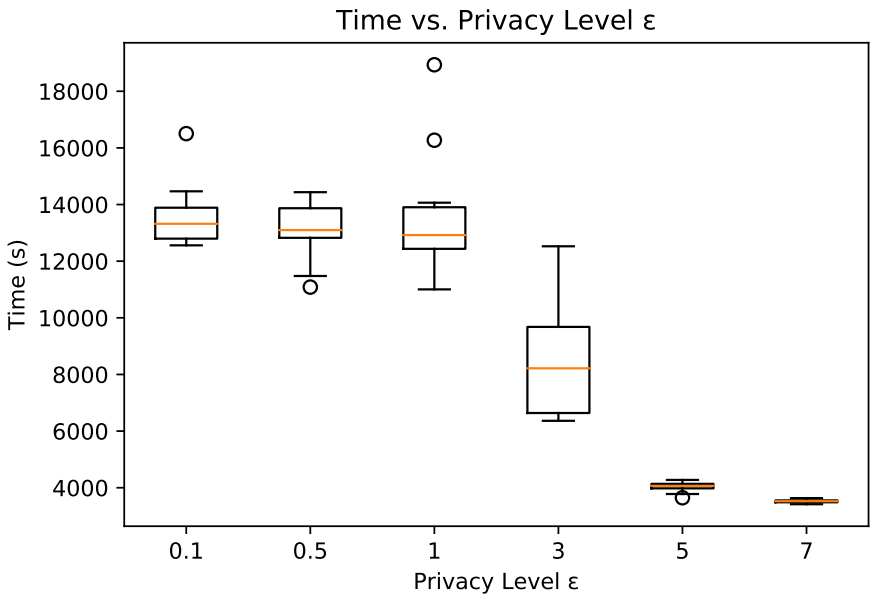


Figure 5.5: Time of computing 60 random nodes with ϵ =0.1 to 7, Facebook data , for three parties.

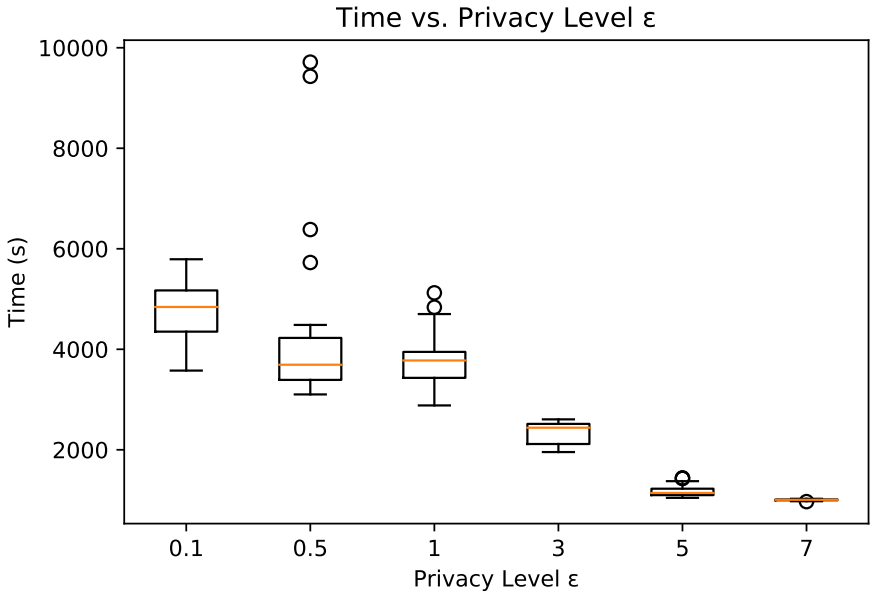


Figure 5.6: Time of computing 60 random nodes with ϵ =0.1 to 7, Enron dataset , for three parties.

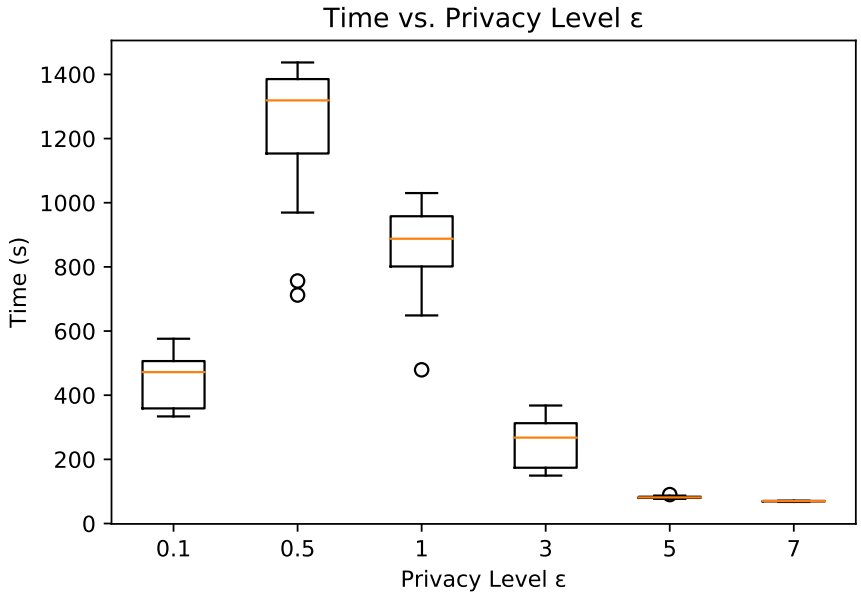


Figure 5.7: Time of computing 60 random nodes with ϵ =0.1 to 7, PGP dataset, for three parties.

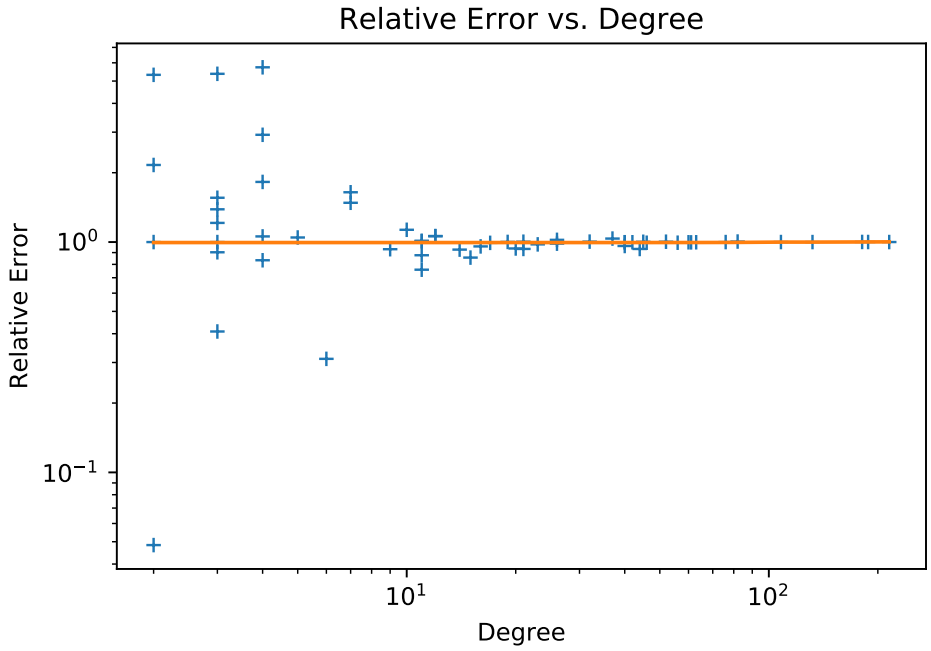


Figure 5.8: Relative error of 60 nodes with different degrees for ϵ =1, Facebook dataset.

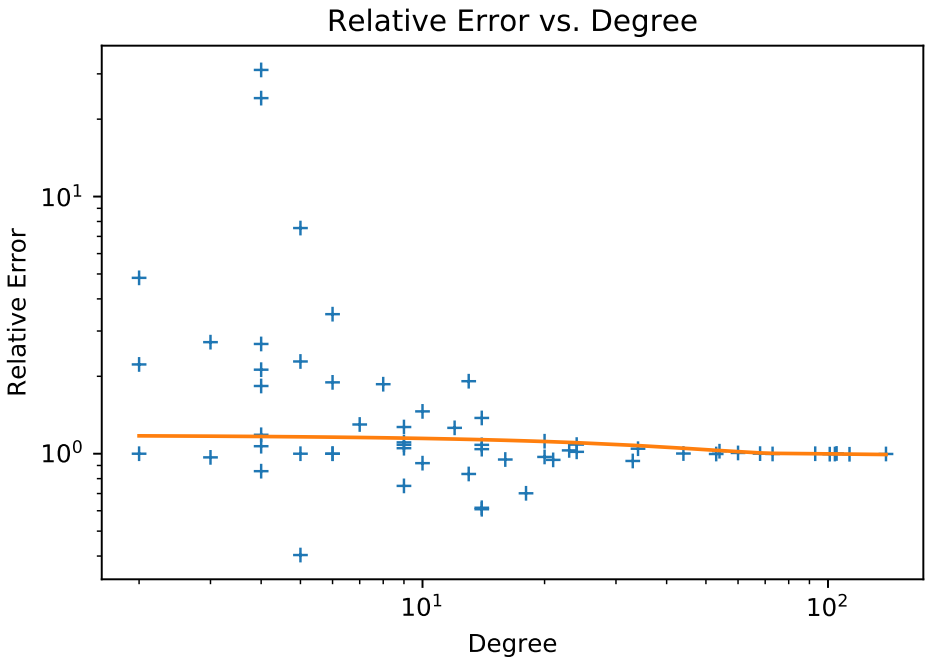


Figure 5.9: Relative error of 60 nodes with different degrees for ϵ =1, degree, Enron dataset.

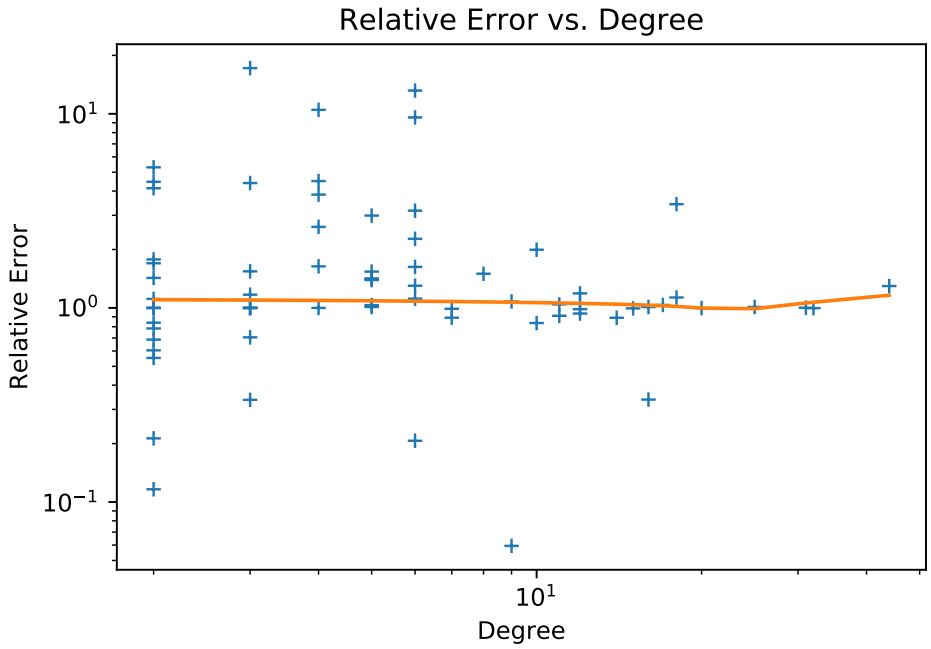


Figure 5.10: Relative error of 60 nodes with different degrees for ϵ =1, PGP dataset.

5.7 Experimental Setup

In order to validate the utility and privacy of PRIVATEEBC, we experimented with three different graphs on Facebook friendships, the Enron email network and Pretty Good Privacy (PGP). These are in common with Chapter 4. Table 5.2 shows the number of nodes and edges of these datasets, for completeness, where users are nodes and inter-user interactions are edges. We employ uniform random sampling in order to partition the graphs into disjoint multi parties while keeping the structure of the graph intact. In addition to evaluations on three parties across datasets, we also validated utility across 2, 3, 5, 7 and 10 parties on the PGP dataset. The experiments were run on a server with 2×28 core Xeon's (112 threads with hyper threading) and 1.5 TB RAM, using Python 3.7 without parallel computations for fair comparison. We employed the `Mpmath` arbitrary precision library for implementing inverse transform sampling (Algorithm 4.2 in the previous chapter) and set the precision to 300 bits. We use relative error between true EBC and private EBC—the lower the relative error the higher the utility. Any errors around 1 or 2 are considered practical as they signify EBCs within the same order of magnitude. We ran the experiment 60 times for each chosen value ϵ by choosing the target ego nodes randomly and robustly aggregating the relative error by median. Throughout we unifset $\epsilon_1 = \epsilon_2 = \epsilon_3 = \epsilon/3$.

Table 5.2: Dataset Characteristics

Name of Dataset	Nodes	Edges
Facebook friendship graph [52]	63,731	817,035
Enron email network [94]	36,692	183,831
Pretty Good Privacy (PGP) [52]	10,680	24,316

5.8 Results

First, we demonstrate how PRIVATEEBC utility varies with increasing privacy budget from 0.1 to 7, for three parties across each of three different graph datasets. The median relative error between real and private EBC represents utility. Figure 5.3 displays the results for Facebook, Enron and PGP datasets, where median relative error decreases significantly when ϵ is increased to a strong level of 1, and remains small for larger ϵ. For strong privacy grantee of $\epsilon = 0.5$, median relative error is usually ≈ 1 for all three datasets. These results demonstrate that PRIVATEEBC *achieves practical utility across a range of graph sizes and privacy levels.*

Next we report utility at $\epsilon = 1$ for the number parties ranging over 2, 3, 5, 7 and 10. Every point in Figure 5.4 shows the median relative error between private and real EBC across 120 randomly chosen nodes in the PGP dataset. *Significantly this represents insignificant degradation to accuracy or privacy when growing the number of parties.*

Remark 38. *While more parties means more calls to* RECIPROCATEANDSUM *and* PRIVATEPATHCOUNT *such that the scale of the second and third mechanisms' Laplace noise increases moderately, the major source of error—*SUBSETRELEASE *due to its impact on all downstream analyses—is not affected by the number of parties as proved in Proposition 28.*

We report on timing analysis for PRIVATEEBC as a function of privacy. Median computation time of 60 random ego nodes for ϵ from 0.1 to 7 is reported in Figure 5.5, 5.6 and 5.7, on Facebook, Enron and PGP datasets. Here total time overall decreases as privacy decreases (increasing ϵ), while a small increase to runtime can be seen at very high levels of privacy (low but increasing ϵ) for Enron it is likely due to different behaviours in the protocol with increasing ϵ. Figure 5.7 exhibits an interesting increase then decrease. This occurs due to random sampling of the nodes, although we expect less timing when the epsilon increases. Sometimes the number of random nodes with high number of two paths are so high that this affects the timing. When the set difference of R and $R\star$ is small, the two-stage sampler generates small numbers of nodes in faster time. However faster runtime with lower privacy dominates behaviour overall.

Figures 5.8, 5.9, 5.10 show how the median relative error is changing by ego node degree. We report results on $\epsilon = 1$, which do not show significant dependence: In Facebook the median relative error is almost constant for different node degrees and in Enron and PGP for node degrees up to 10^2, deviation are approximately 1% and 0.5% of the maximum relative error respectively which is quite low.

5.9 Concluding Remarks

This chapter develops a new protocol for multi party computation of egocentric betweeness centrality (EBC) under edge differential privacy, per party. We significantly improve on past work by extending to multiple parties, achieving very low communication costs, theoretical utility analysis, the facility to release private EBC to all parties, and practical experimental results. For future work we hope to allocate differential privacy budgets per stage, by optimising utility bounds.We also intend to develop a network model that reflects a person's use of multiple media, so the node set is not partitioned, but the privacy of edges remains important.

Chapter 6

Conclusion

In this thesis, we have developed multi-party computation techniques in order to address two challenges. The first challenge being storing and querying graph structured data privately on cloud storage by leveraging secret sharing and cryptographic methods. And secondly, joint computation between parties on graph databases with limited exposure of private data with differential privacy guarantees. We have also extended the second idea from two-party to multi-party settings and provide for every party to obtain the answer of the query with differential privacy guarantee, and similarly for external third party. In this concluding chapter, we summarise the contributions made throughout this thesis and look forward to future research directions.

6.1 Summary of Contributions

In this section, the summary of goals, contributions and challenges of each core chapter are presented.

6.1.1 Privacy-Preserving Queries over Secret-Shared Graph-Structured Data

In Chapter 3, we propose a client-server method based on SPDZ2 multi-party computation technique in order to securely store graph-structured data on cloud

storage and privately query them. In this scenario the cloud authority is the main adversary and stored data and queries are kept private from the cloud authority. This method leverages additive secret sharing and somewhat homomorphic encryption in order to achieve this goal.

This method also provides privacy against any attacker that aims to break in to the system, including the cloud authority itself. The attacker who wants to intrude in the system should break several servers in order to obtain the information. More specifically $n-1$ servers in this technique where the number of servers is n. Hence, the method avoids having a single point of failure by inheriting distributed trust from the SPDZ2 system.

Specifically, this method provides protection against an active adversary model where the cloud authorities can deviate from the protocol. The SPDZ2 protocol provides an integrity check for intermediate results during the running protocol and after revealing the results to the querier. It can guarantee the protection against the active adversary model for either stored data and queries.

Since we are interested in preserving the privacy of the edges, which are the connections between individuals in the network, the data is stored and queried is the adjacency matrix of the nodes. We examine the technique with a graph of 62 nodes for neighbourhood and next-hop queries. The results show, although the technique has a strong security model, the running time for large graph datasets is high due to computation complexity of this method. On the other hand, although the results are not quite satisfying for real time applications, they are acceptable for applications which do not need real time computations.

6.1.2 Differentially Private Two-Party Egocentric Betweenness Centrality

In Chapter 4, we propose differential privacy as a framework for private two-party computation of graph properties. A network is controlled by two service providers, such as two email or telecommunications providers which are interested in jointly computing statistics of their graph databases, and they are reluctant to share their valuable data with other parties. However, they need to share some information about graph databases with the other parties. Here,

we guarantee that the communication of data between two parties are differentially private while one party computes the measure of node importance or ego network betweenneess centrality (EBC) for one of the nodes within that party.

The privacy of edges or connections between nodes should be maintained as is the goal throughout this thesis. So, we apply two basic tools of differential privacy in order to privatise the edges in communications between two parties. We use Laplace and exponential mechanisms to maintain the privacy of the edges. The sensitivity of the query function is a challenge as it can affect the amount of noise added to the system and thus, implicitly changing the utility of the system. Therefore, we design the protocol in such a way that the sensitivity could be bounded in order to have a practical error rate.

The other challenge occurs because of using the exponential mechanism. This mechanism naively samples from the space of possible outcomes, biasing toward the outcomes performing well on a quality function that we have defined. We propose a two stage sampler in order to have exponential saving in time and space. We use inverse sampling method along with this technique to provide an easy to implement. We ran the experiments on a graph of 63000 nodes and the results show efficient and accurate private computation of two-party EBC.

6.1.3 Differentially Private Multi-Party Egocentric Betweenness Centrality

In Chapter 5, we address a problem of multi-party computation. In real world scenarios, it is more probable that more than two parties or organisations are interested in jointly computing statistics of their graph databases while maintaining the privacy of the edges. We again propose differential privacy as a framework for multi-party computation of graph database. In this framework, not only the communication between parties are maintained as differential private as the Chapter 4, but also the answer of the query may be revealed to the all participants and even external with a differential privacy guarantee.

The main challenges here are sensitivity of the query function, and the computation and communication complexity of the protocol. Therefore, in order to present better computation and communication complexity compared to the

two-party scenario while bounding the sensitivity function of non-private EBC analysis, we re-designed the protocol.

The experiments on 2, 3, 5, 7 and 10 parties show that number of parties do not affect the utility of the protocol and we can still have practical error rate while maintaining the differential privacy of the edges for computing EBC of a node.

6.2 Discussion

Essentially, the scalability of the SMC protocols depend on function to be computed and adversary model. In Chapter 3, we are only able to scale to graphs of 64 nodes by leveraging the SPDZ2 protocol and achieving computations like neighbourhood and next-hop with a strong adversary model. Here, the setting would appear extremely challenging owing to the vast number of triples for multiplication functions and bits for comparison required and these can affect the scalability of the protocol. However, a more relaxed adversary model can reduce the complexity and improve the scalability of the protocol.

Chapters 4 and 5 consider a completely different framework. While DP is much more scalable (*e.g.*, we easily to get to graphs of size 63,731) DP is lossy. However we are able to bound a major component of the utility with high probability in Chapter 5, and demonstrate excellent empirical results. While Chapter 4 does not make full use of the different threat model of DP (in being able to release query responses to untrusted third parties), we close the loop in Chapter 5 and safely permit general release.

6.3 Future Research

In this section, we discuss some of the potential research directions that are motivated by this thesis.

Changing Privacy Model

In Chapter 3, one of the possible directions is the change of privacy model from active to passive adversary. In a passive model the communication complexity is less than an active model and it provides the opportunity that the SMC protocol can run for larger sizes of graphs and also can answer more complicated queries, such as Boolean queries more efficiently. However, making the authentication technique explicit is another possible direction that might lead to a stronger security model, but slower protocol running time.

Applying a New Graph Structure

In Chapter 3, one of the issues to be resolved for the future, is the structure of the graph—a more succinct data structure can be devised for storing larger graphs. Including time, location and other useful information in the graph is another possible direction in order improve the scheme's functionality. Applying a new graph structure such as weighted graphs is also one of the possible directions in Chapters 4 and 5. Weighted edges pose additional challenges, as we would likely want to maintain privacy of the weights in the graph.

Privacy Budget Optimisation

In Chapters 4 and 5, we allocate the same privacy budget to every party through all the stages of the protocol. One possible direction as future work could be allocating differential privacy budgets per stage and per party by optimising utility bounds.

EBC Multi-Party Computation on Multiple Media

One of the immediate applications of the differential privacy framework for computing EBC, would be in a network model that reflects a person's use of multiple social media platforms, such as Facebook, Instagram, LinkedIn and Tweeter. In this case the node set is not partitioned like our scenario, but the privacy of edges is important. Here, a user might have a similar identity in

two or more platforms and also have its own connections per network. So that a party could compute the EBC of a node, the party needs to know about the connections of the node in other media and it can reveal information about these edges.

Answering Different Queries

In Chapters 4 and 5, we propose differential private two-party and multi-party frameworks for computing EBC. One of the possible directions of this research is using the building blocks of these frameworks in order to answer different queries on graph databases, such as network statistics based on shortest paths between nodes.

Answering Multiple EBC Queries

In Chapters 4 and 5, we propose differential private two-party and multi-party frameworks for computing EBC. One of the possible directions of this research is answering multiple EBC queries. We can repeat the algorithm, with privacy budget increasing linearly as our mechanism preserve pure DP. One may speculate on privatives achieve the multiple nodes goal more efficiently. The new work of Gopi*et al.* [46] is potentially very relevant for combining neighbour networks. An easy trick is to not resend path counts within the same neighbourhood component – in terms of privacy budget used by the Laplace mechanism. In terms of running Lemma 21 twice, we would have a total sensitivity of $2|R_1| + 2|R_2|$ if separately releasing two EBCs for the private count stage (R1 and R2 might be the neighbouring networks of the two nodes in this expression). But if there is overlap between the networks, potentially we don't have to resend that overlap. And so we might reduce this effective sensitivity and therefore privacy budget used.

Considering Node Differential Privacy

Another open problem is maintaining the node differential privacy of the graph data bases along with edge privacy in the two and multi-party settings.

Bounding the Sensitivity

In Chapters 4 and 5, we design the protocols in such a way that the sensitivity of the functions are bounded. However, for some query functions it is not possible to bound the functions just by design and other techniques such as, local sensitivity is applied to bound the sensitivity [78]. The open problem is then: proposing another method to bound the sensitivity of graph structured-data to perform differentially private computations with practical error rates.

Local Differential Privacy

It is an interesting direction to investigate how the local model [38,56] differs from central differential privacy. In other words how does the privacy-utility tradeoff change when moving to local differential privacy from central differential privacy in the two-party and multi-party frameworks?

Lower Bound for Utility

An interesting direction for all privacy mechanisms especially differentially private approaches is to consider computing lower bounds on the utility. By finding the minimum error, one can calibrate the efficiency of a mechanism and benchmark utility analysis.

More Accurate Utility Bound

In Chapter 5, a more accurate utility bound can be achieved by considering Laplace mechanisms in multi-party protocol. In Section 5.6 of Chapter 5, we just consider the exponential mechanism in order to bound the utility while we use two Laplace mechanisms in the protocol that need to be considered for providing more accurate utility bounds.

Chapter 7

Bibliography

[1] Charu C Aggarwal and S Yu Philip. A general survey of privacy-preserving data mining models and algorithms. In *Privacy-Preserving Data Mining*, pages 11–52. Springer, 2008.

[2] Abdelrahaman Aly, Edouard Cuvelier, Sophie Mawet, Olivier Pereira, and Mathieu Van Vyve. Securely solving simple combinatorial graph problems. In *International Conference on Financial Cryptography and Data Security*, pages 239–257. Springer, 2013.

[3] Gilad Asharov and Yehuda Lindell. A full proof of the BGW protocol for perfectly secure multiparty computation. *Journal of Cryptology*, 30(1):58–151, 2017.

[4] Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th International Conference on World Wide Web*, pages 181–190. ACM, 2007.

[5] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *Annual International Cryptology Conference*, pages 420–432. Springer, 1991.

[6] Assaf Ben-David, Noam Nisan, and Benny Pinkas. Fairplaymp: a system for secure multi-party computation. In *Proceedings of the 15th ACM*

Conference on Computer and Communications Security, pages 257–266. ACM, 2008.

[7] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 1–10. ACM, 1988.

[8] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 169–188. Springer, 2011.

[9] Smriti Bhagat, Graham Cormode, Balachander Krishnamurthy, and Divesh Srivastava. Class-based graph anonymization for social network data. *Proceedings of the VLDB Endowment*, 2(1):766–777, 2009.

[10] Raghav Bhaskar, Srivatsan Laxman, Adam Smith, and Abhradeep Thakurta. Discovering frequent patterns in sensitive data. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 503–512. ACM, 2010.

[11] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 87–96. ACM, 2013.

[12] Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. In *European Symposium on Research in Computer Security*, pages 192–206. Springer, 2008.

[13] Justin Brickell and Vitaly Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 236–252. Springer, 2005.

[14] Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas Dimitropoulos. SEPIA: Privacy-preserving aggregation of multi-domain network events and statistics. *Network*, 1(101101), 2010.

[15] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 494–503. ACM, 2002.

[16] Ruichuan Chen, Alexey Reznichenko, Paul Francis, and Johanes Gehrke. Towards statistical queries over distributed private user data. In *Presented as part of the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*, pages 169–182, 2012.

[17] Shixi Chen and Shuigeng Zhou. Recursive mechanism: towards node differential privacy and unrestricted joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 653–664. ACM, 2013.

[18] Chris Clifton, Murat Kantarcioglu, Jaideep Vaidya, Xiaodong Lin, and Michael Y Zhu. Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):28–34, 2002.

[19] Graham Cormode, Divesh Srivastava, Ninghui Li, and Tiancheng Li. Minimizing minimality and maximizing utility: analyzing method-based attacks on anonymized data. *Proceedings of the VLDB Endowment*, 3(1-2):1045–1056, 2010.

[20] Ronald Cramer, Ivan Bjerre Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation*. Cambridge University Press, 2015.

[21] Ronald Cramer, Matthew Franklin, Berry Schoenmakers, and Moti Yung. Multi-authority secret-ballot elections with linear work. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 72–83. Springer, 1996.

[22] Chris Culnane, Benjamin I. P. Rubinstein, and Vanessa Teague. Health data in an open world. *CoRR*, abs/1712.05627, 2017.

[23] Ivan Damgård, Kasper Damgård, Kurt Nielsen, Peter Sebastian Nordholt, and Tomas Toft. Confidential benchmarking based on multiparty computation. In *International Conference on Financial Cryptography and Data Security*, pages 169–187. Springer, 2016.

[24] Ivan Damgård, Martin Geisler, Mikkel Krøigaard, and Jesper Buus Nielsen. Asynchronous multiparty computation: Theory and implementation. In *International Workshop on Public Key Cryptography*, pages 160–179. Springer, 2009.

[25] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P Smart. Practical covertly secure mpc for dishonest majority–or: breaking the SPDZ limits. In *European Symposium on Research in Computer Security*, pages 1–18. Springer, 2013.

[26] Ivan Damgård and Jesper Buus Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In *Annual International Cryptology Conference*, pages 247–264. Springer, 2003.

[27] Ivan Damgård and Claudio Orlandi. Multiparty computation for dishonest majority: From passive to active security at low cost. In *Annual Cryptology Conference*, pages 558–576. Springer, 2010.

[28] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology–CRYPTO 2012*, pages 643–662. Springer, 2012.

[29] Wei-Yen Day, Ninghui Li, and Min Lyu. Publishing graph degree distribution with node differential privacy. In *Proceedings of the 2016 International Conference on Management of Data*, pages 123–138. ACM, 2016.

[30] Xuan Ding, Lan Zhang, Zhiguo Wan, and Ming Gu. A brief survey on de-anonymization attacks in online social networks. In *2010 International*

Conference on Computational Aspects of Social Networks, pages 611–615. IEEE, 2010.

[31] Hoang Giang Do and Wee Keong Ng. Privacy-preserving triangle counting in distributed graphs. In *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, pages 917–924. IEEE, 2016.

[32] Yitao Duan, Jingtao Wang, Matthew Kam, and John Canny. Privacy preserving link analysis on dynamic weighted graph. *Computational & Mathematical Organization Theory*, 11(2):141–159, 2005.

[33] Paul AC Duijn and Peter PHM Klerks. Social network analysis applied to criminal networks: Recent developments in dutch law enforcement. In *Networks and Network Analysis for Defence and Security*, pages 121–159. Springer, 2014.

[34] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT'06, Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503. Springer, 2006.

[35] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pages 265–284. Springer, 2006.

[36] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[37] Martin Everett and Stephen P Borgatti. Ego network betweenness. *Social Networks*, 27(1):31–38, 2005.

[38] Alexandre Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 211–222, 2003.

[39] Pierluigi Failla. Heuristic search in encrypted graphs. In *2010 Fourth International Conference on Emerging Security Information, Systems and Technologies*, pages 82–87. IEEE, 2010.

[40] Linton C Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215–239, 1978.

[41] IBM System G. Graph databases. `https://www.ibm.com/blogs/cloud-archive/2016/07/graph-databases-natural-way-to-represent-data/`, last accessed 2019-10-09.

[42] Srivatsava Ranjit Ganta, Shiva Prasad Kasiviswanathan, and Adam Smith. Composition attacks and auxiliary information in data privacy. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 265–273. ACM, 2008.

[43] K-I Goh, Eulsik Oh, Byungnam Kahng, and Doochul Kim. Betweenness centrality correlation in social networks. *Physical Review E*, 67(1):017101, 2003.

[44] Oded Goldreich. Secure multi-party computation (working draft). *http://www. wisdom. weizmann. ac. il/˜ oded/pp. html*, 1998.

[45] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 218–229. ACM, 1987.

[46] Sivakanth Gopi, Pankaj Gulhane, Jana Kulkarni, Judy Hanwen Shen, Milad Shokouhi, and Sergey Yekhanin. Differentially private set union. *arXiv*, February 2020.

[47] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[48] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *2009 Ninth IEEE International Conference on Data Mining*, pages 169–178. IEEE, 2009.

[49] Michael Hay, Gerome Miklau, David Jensen, Philipp Weis, and Siddharth Srivastava. Anonymizing social networks. *Computer Science Department Faculty Publication Series*, page 180, 2007.

[50] William H Hsu, Andrew L King, Martin SR Paradesi, Tejaswi Pydimarri, and Tim Weninger. Collaborative and structural recommendation of friends using weblog-based social network analysis. In *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, volume 6, pages 55–60, 2006.

[51] Mathias Humbert, Kévin Huguenin, Joachim Hugonot, Erman Ayday, and Jean-Pierre Hubaux. De-anonymizing genomic databases using phenotypic traits. *Proceedings on Privacy Enhancing Technologies*, 2015(2):99–114, 2015.

[52] Institute of web science and technologies at the university of Koblenz-Landau. The Koblenz network collection. `http://konect.uni-koblenz.de`, last accessed 2019-10-09.

[53] Geetha Jagannathan and Rebecca N Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 593–599. ACM, 2005.

[54] Xin Jin, Nan Zhang, and Gautam Das. Algorithm-safe privacy-preserving data publishing. In *Proceedings of the 13th International Conference on Extending Database Technology*, pages 633–644. ACM, 2010.

[55] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. Private analysis of graph structure. *ACM Transactions on Database Systems (TODS)*, 39(3):22, 2014.

[56] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

[57] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Analyzing graphs with node differential privacy. In *Theory of Cryptography Conference*, pages 457–476. Springer, 2013.

[58] Daniel Kifer and Bing-Rong Lin. Towards an axiomatization of statistical privacy and utility. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 147–158. ACM, 2010.

[59] Sejeong Kwon, Meeyoung Cha, Kyomin Jung, Wei Chen, and Yajun Wang. Prominent features of rumor propagation in online social media. In *2013 IEEE 13th International Conference on Data Mining*, pages 1103–1108. IEEE, 2013.

[60] Kristen LeFevre, David J DeWitt, and Raghu Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 49–60. ACM, 2005.

[61] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115. IEEE, 2007.

[62] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. Closeness: A new privacy measure for data publishing. *IEEE Transactions on Knowledge and Data Engineering*, 22(7):943–956, 2009.

[63] Kun Liu and Evimaria Terzi. Towards identity anonymization on graphs. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 93–106. ACM, 2008.

[64] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-

anonymity. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 24–24. IEEE, 2006.

[65] Dahlia Malkhi, Noam Nisan, Benny Pinkas, Yaron Sella, et al. Fairplay-secure two-party computation system. In *USENIX Security Symposium*, volume 4. San Diego, CA, USA, 2004.

[66] Peter V Marsden. Egocentric and sociocentric measures of network centrality. *Social Networks*, 24(4):407–422, 2002.

[67] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science*, pages 94–103. IEEE, 2007.

[68] Frank D McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, pages 19–30. ACM, 2009.

[69] Federico Monti, Fabrizio Frasca, Davide Eynard, Damon Mannion, and Michael M Bronstein. Fake news detection on social media using geometric deep learning. *arXiv preprint arXiv:1902.06673*, 2019.

[70] Yvonne Mülle, Chris Clifton, and Klemens Böhm. Privacy-integrated graph clustering through differential privacy. In *18th International Conference on Extending Database Technology/ 18th International Conference on Database Theory Workshops, March 23-27, 2015, March 23-27, 2015*, pages 247–254, 2015.

[71] Arvind Narayanan, Elaine Shi, and Benjamin IP Rubinstein. Link prediction by de-anonymization: How we won the kaggle social network challenge. In *The 2011 International Joint Conference on Neural Networks*, pages 1825–1834. IEEE, 2011.

[72] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large datasets (how to break anonymity of the Netflix prize dataset). *University of Texas at Austin*, 2008.

[73] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *2009 30th IEEE Symposium on Security and Privacy*, pages 173–187, 2009.

[74] Mehmet Ercan Nergiz, Maurizio Atzori, and Chris Clifton. Hiding the presence of individuals from shared databases. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of data*, pages 665–676. ACM, 2007.

[75] Mehmet Ercan Nergiz and Chris Clifton. δ-presence without complete world knowledge. *IEEE Transactions on Knowledge and Data Engineering*, 22(6):868–883, 2009.

[76] Hiep H Nguyen, Abdessamad Imine, and Michaël Rusinowitch. Detecting communities under differential privacy. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, pages 83–93. ACM, 2016.

[77] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In *Annual Cryptology Conference*, pages 681–700. Springer, 2012.

[78] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the Thirty-Ninth Annual ACM symposium on Theory of Computing*, pages 75–84. ACM, 2007.

[79] Node4j. Graph databases. `https://neo4j.com/`, last accessed 2019-10-09.

[80] University of Bristol. SPDZ software. `https://www.cs.bris.ac.uk/research/cryptographysecurity/SPDZ/`, last accessed 2019-10-09.

[81] Ontotext. Graph databases. `https://ontotext.com/`, last accessed 2019-10-09.

[82] Indra Rajasingh, Bharati Rajan, and Florence Isido. Betweeness-centrality of grid networks. In *2009 International Conference on Computer Technology and Development*, volume 1, pages 407–410. IEEE, 2009.

[83] Sofya Raskhodnikova and Adam Smith. Efficient Lipschitz extensions for high-dimensional graph statistics and node private degree distributions. *arXiv preprint arXiv:1504.07912*, 2015.

[84] Vibhor Rastogi, Michael Hay, Gerome Miklau, and Dan Suciu. Relationship privacy: output perturbation for queries with joins. In *Proceedings of the Twenty-Eighth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 107–116. ACM, 2009.

[85] David Rebollo-Monedero, Jordi Forne, and Josep Domingo-Ferrer. From t-closeness-like privacy to postrandomization via information theory. *IEEE Transactions on Knowledge and Data Engineering*, 22(11):1623–1636, 2009.

[86] Benjamin IP Rubinstein, Peter L Bartlett, Ling Huang, and Nina Taft. Learning in a large function space: Privacy-preserving mechanisms for SVM learning. *arXiv preprint arXiv:0911.5708*, 2009.

[87] Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. Efficient privacy-preserving face recognition. In *International Conference on Information Security and Cryptology*, pages 229–244. Springer, 2009.

[88] Pierangela Samarati and Latanya Sweeney. Generalizing data to provide anonymity when disclosing information. In *Proceedings of the Seventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, volume 98, page 188. Citeseer, 1998.

[89] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[90] Kumar Sharad and George Danezis. An automated social graph de-anonymization technique. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 47–58. ACM, 2014.

[91] Entong Shen and Ting Yu. Mining frequent graph patterns with differential privacy. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 545–553. ACM, 2013.

[92] HAN Shuguo and Wee Keong Ng. Multi-party privacy-preserving decision trees for arbitrarily partitioned data. *International Journal of Intelligent Control and Systems*, 12(4):351–358, 2007.

[93] Nigel P Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *Designs, Codes and Cryptography*, 71(1):57–81, 2014.

[94] Stanford University. Stanford large network dataset collection. `https://snap.stanford.edu/data/index.html`, last accessed 2019-10-09.

[95] Latanya Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):571–588, 2002.

[96] Latanya Sweeney. k-anonymity: A model for protecting privacy. *Int. J. Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[97] Jaideep Vaidya and Chris Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 639–644. ACM, 2002.

[98] Yue Wang, Xintao Wu, Jun Zhu, and Yang Xiang. On learning cluster coefficient of private networks. *Social network analysis and mining*, 3(4):925–938, 2013.

[99] Raymond Chi-Wing Wong, Ada Wai-Chee Fu, Ke Wang, Philip S Yu, and Jian Pei. Can the utility of anonymized data be used for privacy breaches? *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(3):16, 2011.

[100] Xiaokui Xiao and Yufei Tao. Personalized privacy preservation. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pages 229–240. ACM, 2006.

[101] Xiaokui Xiao, Yufei Tao, and Nick Koudas. Transparent anonymization: Thwarting adversaries who know the algorithm. *ACM Transactions on Database Systems (TODS)*, 35(2):8, 2010.

[102] Xiaokui Xiao, Ke Yi, and Yufei Tao. The hardness and approximation algorithms for l-diversity. In *Proceedings of the 13th International Conference on Extending Database Technology*, pages 135–146. ACM, 2010.

[103] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 1982.

[104] Xiaowei Ying and Xintao Wu. Randomizing social networks: a spectrum preserving approach. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, pages 739–750. SIAM, 2008.

[105] Jeongah Yoon, Anselm Blumer, and Kyongbum Lee. An algorithm for modularity analysis of directed and weighted biological networks based on edge-betweenness centrality. *Bioinformatics*, 22(24):3106–3108, 2006.

[106] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. Private release of graph statistics using ladder functions. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 731–745. ACM, 2015.

[107] Bin Zhou and Jian Pei. Preserving privacy in social networks against neighborhood attacks. In *The 24th International Conference on Data Engineering*, volume 8, pages 506–515. Citeseer, 2008.

[108] Bin Zhou, Jian Pei, and WoShun Luk. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *ACM SIGKDD Explorations Newsletter*, 10(2):12–22, 2008.

[109] Tianqing Zhu, Mengmeng Yang, Ping Xiong, Yang Xiang, and Wanlei Zhou. An iteration-based differentially private social network data release. *Computer Systems Science and Engineering*, 2018.