Florida International University

FIU Digital Commons

FIU Electronic Theses and Dissertations | University Graduate School

11-15-2019

Trajectory Privacy Preservation and Lightweight Blockchain Techniques for Mobility-Centric IoT

Abdur Bin Shahid
School of Computing annd Information Sciences, Florida International University, ashah044@fiu.edu

Follow this and additional works at: https://digitalcommons.fiu.edu/etd

Part of the Databases and Information Systems Commons, Data Storage Systems Commons, and the Information Security Commons

Recommended Citation

Shahid, Abdur Bin, "Trajectory Privacy Preservation and Lightweight Blockchain Techniques for Mobility-Centric IoT" (2019). *FIU Electronic Theses and Dissertations*. 4308.
https://digitalcommons.fiu.edu/etd/4308

This work is brought to you for free and open access by the University Graduate School at FIU Digital Commons. It has been accepted for inclusion in FIU Electronic Theses and Dissertations by an authorized administrator of FIU Digital Commons. For more information, please contact dcc@fiu.edu.

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

TRAJECTORY PRIVACY PRESERVATION AND LIGHTWEIGHT BLOCKCHAIN TECHNIQUES FOR MOBILITY-CENTRIC IOT

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Abdur Rahman Bin Shahid

2019

To: Dean John L. Volakis
College of Engineering and Computing

This dissertation, written by Abdur Rahman Bin Shahid, and entitled Trajectory Privacy Preservation and Lightweight Blockchain Techniques for Mobility-Centric IoT, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Sundaraja Sitharama Iyengar

Deng Pan

Kang K. Yen

Leonardo Bobadilla

Laurent Njilla

Niki Pissinou, Major Professor

Date of Defense: November 15, 2019

The dissertation of Abdur Rahman Bin Shahid is approved.

Dean John L. Volakis
College of Engineering and Computing

Andrés G. Gil
Vice President for Research and Economic Development and Dean of the University Graduate School

Florida International University, 2019

© Copyright 2019 by Abdur Rahman Bin Shahid
All rights reserved.

DEDICATION

To my grandmother, parents, siblings, wife, and son.

Without your love, patience, and encouragement this dissertation would not have been possible.

ACKNOWLEDGMENTS

First, I would like to express my sincere gratitude and appreciation to my advisor Dr. Niki Pissinou, for believing in me and for the continuous support of my Ph.D. study. Her guidance, encouragement, patience, and immense knowledge helped me in every step of my research and completion of this thesis. I am deeply thankful to her for helping me to become the scientist I am today.

I would like to acknowledge the contribution of the committee members, Dr. S.S. Iyengar, Dr. kang Yen, Dr. Deng Pan, Dr. Leonardo Bobadilla, and Dr. Laurent Njilla, for their support, valuable suggestions, and insightful comments on the research of this dissertation. I am specially thankful to Dr. Njilla for supporting our research on lightweight blockchain through Air Force Office of Scientific Research (AFOSR) grants.

I would like to express my sincere thanks to Dr. Kia Makki for his help during my Ph.D. studies. We have coauthored multiple papers and his comments and suggestions on my work were invaluable. I am also grateful to Col. Jerry Miller for his advice in editing my papers. Thanks to Ms. Olga Carbonell and Ms. Ariana Taglioretti for helping me with different administrative tasks.

I am thankful to all my fellow latemates and friends, including Dr. Georges A. Kamhoua, Dr. Xinyu Jin, Dr.Mingming Guo, Samia Tasnim, Hussein Zangoti, Concepción Sańchez Alemań, Sheila Alemany, Azizul Hakim, and Sanjib Biswas. I was fortunate to work with some brilliant undergraduate students in NSR REU program, including Edwin Aguilar, Eric Perez, Corey Staie, Rain Kawn, Keyan Zolfaghar, Aditya Shetty, and Liz Jeukeng. Special thanks to Edwin and Eric for their contribution in the development of lightweight blockchain.

Thanks to Florida International University, School of Computing and Information Sciences, University Graduate School, U.S. Air Force Office of Scientific Re-

search, National Science Foundation's REU and RET sites, and Army Research Office for supporting my research through grants and fellowship.

Last but not least, I would like to thank my parents, siblings, wife, son, and parents-in-law for supporting me throughout this long journey of Ph.D. Without their sacrifice, continued support, and encouragement, I doubt that this dissertation would have been possible.

ABSTRACT OF THE DISSERTATION

TRAJECTORY PRIVACY PRESERVATION AND LIGHTWEIGHT BLOCKCHAIN TECHNIQUES FOR MOBILITY-CENTRIC IOT

by

Abdur Rahman Bin Shahid

Florida International University, 2019

Miami, Florida

Professor Niki Pissinou, Major Professor

Various research efforts have been undertaken to solve the problem of trajectory privacy preservation in the Internet of Things (IoT) of resource-constrained mobile devices. Most attempts at resolving the problem have focused on the centralized model of IoT, which either impose high delay or fail against a privacy-invading attack with long-term trajectory observation. These proposed solutions also fail to guarantee location privacy for trajectories with both geo-tagged and non-geo-tagged data, since they are designed for geo-tagged trajectories only. While a few blockchain-based techniques have been suggested for preserving trajectory privacy in decentralized model of IoT, they require large storage capacity on resource-constrained devices and can only provide conditional privacy when a set of authorities governs the blockchain. This dissertation addresses these challenges to develop efficient trajectory privacy-preservation and lightweight blockchain techniques for mobility-centric IoT.

We develop a pruning-based technique by quantifying the relationship between trajectory privacy and delay for real-time geo-tagged queries. This technique yields higher trajectory privacy with a reduced delay than contemporary techniques while preventing a long-term observation attack. We extend our study with the consideration of the presence of non-geo-tagged data in a trajectory. We design an attack

model to show the spatiotemporal correlation between the geo-tagged and non-geo-tagged data which undermines the privacy guarantee of existing techniques. In response, we propose a methodology that considers the spatial distribution of the data in trajectory privacy-preservation and improves existing solutions, in privacy and usability.

With respect to blockchain, we design and implement one of the first blockchain storage management techniques utilizing the mobility of the devices. This technique reduces the required storage space of a blockchain and makes it lightweight for resource-constrained mobile devices. To address the trajectory privacy challenges in an authority-based blockchain under the short-range communication constraints of the devices, we introduce a silence-based one of the first technique to establish a balance between trajectory privacy and blockchain utility.

The designed trajectory privacy- preservation techniques we established are light-weight and do not require an intermediary to guarantee trajectory privacy, thereby providing practical and efficient solution for different mobility-centric IoT, such as mobile crowdsensing and Internet of Vehicles.

TABLE OF CONTENTS

CHAPTER PAGE

1. INTRODUCTION 1
1.1 Background 1
1.2 Motivation 3
1.3 Research Problems 8
1.4 Research Objectives 9
1.5 Research Contributions 14
1.6 Dissertation Outline 20

2. RELATED WORK 21
2.1 Location Privacy Preservation in Location-Based Service (LBS) 22
2.1.1 Location Inference Models 22
2.1.2 Location Privacy-Preserving Approaches 24
2.2 Blockchain for Internet of Things (IoT) 28
2.3 Location Privacy Preservation in Blockchain 31

3. PRIVACY PRESERVING MECHANISM FOR CONTINUOUS LOCATION SHARING IN CENTRALIZED INTERNET OF THINGS (IOT) SYSTEMS 34
3.1 Introduction 35
3.2 Motivation and Problem Statement 37
3.3 Background 39
3.3.1 Fundamental Concepts 39
3.3.2 Privacy Evaluation 41
3.3.3 Problem Formulation 41
3.3.4 System Model 42
3.4 Proposed Delay-Aware Privacy Preserving Approach 43
3.5 Scheme Analysis 49
3.6 Experimentation and Analysis 50
3.6.1 Privacy Level 52
3.6.2 Impact of Pruning 54
3.6.3 Communication Cost 55
3.6.4 Storage Cost 56
3.7 Discussion and Summary 56

4. LOCATION INFERENCE ATTACKS ON GEO-TAGGED AND NON GEO-TAGGED DATA AND THEIR COUNTERMEASURES 57
4.1 Introduction 57
4.2 Motivation and Problem Statement 59
4.3 Background 62
4.3.1 Fundamental Concepts 63
4.3.2 System Model 63

4.3.3 Adversary Model . . . 64
4.4 Proposed Privacy Preserving Geo-Tagged and Non Geo-Tagged Data Sharing Approach . . . 65
4.4.1 Dummy Generation . . . 67
4.4.2 Privacy Analysis . . . 69
4.5 Experimental Evaluation . . . 69
4.5.1 Privacy Analysis . . . 71
4.5.2 computation cost . . . 73
4.6 Discussion and Summary . . . 73

5. SPATIOTEMPORAL BLOCKCHAIN MANAGEMENT FOR RESOURCE-CONSTRAINED IOT DEVICES TO ACHIEVE DECENTRALIZED PRIVACY . . . 75
5.1 Introduction . . . 76
5.2 Motivation and Problem Statement . . . 78
5.3 Background . . . 78
5.3.1 Size of a blockchain . . . 79
5.3.2 System Model and Important Assumptions . . . 80
5.4 Proposed Sensor-Chain Framework . . . 81
5.4.1 Naive Approach: Conventional Blockchain . . . 81
5.4.2 Our First Approach: Improved Temporal Blockchain . . . 81
5.4.3 Our Second Approach: Spatial Blockchain . . . 83
5.4.4 Our Best Approach: Sensor-Chain . . . 85
5.4.5 Analysis . . . 88
5.5 Proof-of-Concept Evaluation . . . 89
5.6 Implementation Detail of Sensor-Chain . . . 92
5.6.1 Key Components . . . 97
5.6.2 Running the Demonstration . . . 112
5.7 Discussion and Summary . . . 116

6. QUANTIFYING TRAJECTORY PRIVACY IN BLOCKCHAIN-BASED INTERNET OF THINGS (IOT) . . . 119
6.1 Introduction . . . 119
6.2 Motivation and Problem Statement . . . 121
6.3 Background . . . 122
6.3.1 Blockchain System Model . . . 124
6.3.2 Attack Model . . . 124
6.3.3 Problem Formulation and Design Goals . . . 127
6.4 The BlockPriv Approach . . . 127
6.5 Scheme Analysis . . . 132
6.5.1 Privacy Analysis . . . 133
6.5.2 Utility Analysis . . . 134
6.5.3 Security Analysis . . . 134

6.6 Experimental Evaluation . . . 137
6.6.1 Experimental Settings . . . 138
6.6.2 Experiment Results . . . 139
6.7 Conclusion . . . 144

7. LIMITATIONS, FUTURE WORK, AND CONCLUSION . . . 146
7.1 Limitations . . . 146
7.1.1 Delay-Aware Long-Term Privacy Preservation for Frequently Visited Locations . . . 146
7.1.2 Location Inference Attacks on Geo-Tagged and Non geo-Tagged data and Their Countermeasures . . . 147
7.1.3 Spatiotemporal Blockchain Management for Resource-Constrained IoT devices to Achieve Decentralized Privacy . . . 148
7.1.4 Quantifying Location Privacy in Permissioned Blockchain-Based Internet of Things (IoT) . . . 149
7.2 Future Work . . . 150
7.2.1 Secure and Privacy Preserving Inter- Blockchain Communication Scheme for IoT . . . 150
7.2.2 Developing Privacy-Preserving and Secure Scheme for Merging Multiple mobility-centric IoT Blockchains . . . 153
7.2.3 Distributed Spatiotemporal Federated Learning using Blockchain and Smart Contract . . . 154
7.3 Conclusion . . . 155

BIBLIOGRAPHY . . . 157

VITA . . . 174

LIST OF TABLES

TABLE PAGE

3.1 Table of Symbols 39

3.2 Dataset Statistics 51

4.1 Table of Symbols 62

4.2 Average Computation Cost(in second) for Different k 73

5.1 Table of Symbols 79

5.2 Parameters used in the Experiment 90

5.3 Table of Commands 114

6.1 Table of Symbols 123

6.2 Dataset Statistics 137

6.3 Simulation Setup Parameters 138

6.4 Pearson's Correlation Values 139

LIST OF FIGURES

FIGURE PAGE

1.1 Real cases on trajectory privacy issues. 2

1.2 Architectures for Trajectory Privacy Preserving Mechanism (TTPM): (a) Trusted Third Party (TTP)-based architecture and (b) User-centric architecture. 10

3.1 Location inference attacks considered In this chapter: (a) Map, probability distribution, and personal context linking attacks on single query, (b) region intersection attack on multiple short-term queries, (c) Real-Time Traffic Information (RTTI) based time-reachability, or maximum movement-boundary based attack, and (d) long-term obfuscated location tracking attack. 37

3.2 Proposed system model 42

3.3 CR pruning example with $\delta t = 1\ minute$: Each entry refers to the *delay*(in min.) to reach from $\mathrm{W_A}(m)$ to $\mathrm{W}_i(n)$. To reach from $\mathrm{W}_A(2)$ to any location in W_i minimum $(\mathrm{T_B}-\mathrm{T_A})+2.3$ minutes are required. On the other side, to reach $W_i(2)$ from any location in $\mathrm{W_A}$ minimum $(\mathrm{T_B}-\mathrm{T_A})+2.6$ minutes are required. Thus, we exclude both locations from W_i and $\mathrm{W_A}$, respectively, to get pruned CRs. 48

3.4 Locations in (a) NYC and (b) Tokyo datasets. 51

3.5 Relationship between privacy area requirement $\mathcal{A}$ and k; and their impact on privacy level $\mathrm{E_{CR}}$. 52

3.6 Comparison in terms of privacy level, $\mathrm{E_{CR}}$ among different approaches. KLAP settings: $k = 9, l = 5, \alpha = 10$ 53

3.7 Impact of CR Pruning on delay management for (a) NYC and (b) Tokyo. 54

3.8 $(\Delta/\mathrm{E_{CR}})$ based comparison among different methods for (a) NYC & (b) Tokyo. Original-Q refers to the original query without any privacy-preservation. For comparison purpose we use $E_{\mathrm{CR}} = 1$ for Original-Q. 55

4.1 Location inference motivation:(a) a user generates a set of locations using an existing location-privacy preserving mechanism (TPPM) on a mobile device (e.g., smartphone, smartwatch, and so on)to conceal her original location in a geo-tagged photo and (b) submits them to an LBSN. (c) With access to the LBSN, an adversary's goal is to find a user's original location from the location set without processing the content of the photo. 59

4.2 (a) 0/1-probability based inference: User hides his location of geo-tagged photo with 3 different dummy locations using a check-in probability-based(P_c) existing TPPM. Two of those locations have zero photo-sharing probability(P_ρ)(x marked), thus excluded by an adversary. (b) Example on real dataset: adversary's success rate $\sigma(\%)$ on Random[NLZ+14] and Baseline[NLZ+15] algorithms(using $k = 20$) with 0/1-probability based inference, performed using a real dataset. (c) and (d) Inference based on spatiotemporal correlation: Although dummy locations set and photo are submitted in different time, as $(T_1 - T_0)$ is small, an adversary can guess with high confidence that both events were generated from the same location. Thus, can exclude some dummies in a similar way. 61

4.3 Proposed system model: (a) Photo-check's system model on user's device with 4 possible actions to preserve location privacy of an event, (b) visualization of example privacy-preserved events in LBSN. . . . 64

4.4 Photo-check flowchart. Here loc, ph, and Δt refer to location, photo, and $(T_{prev} - T_{cur})$ respectively. 66

4.5 Impact of 0/1-probability based inference on DLS and baseline algorithms at locations with different check-in probability ($k = 20$). Here, (a) and (b) are the locations with the highest and lowest check-in probability, respectively. 70

4.6 Distribution of degree of privacy $\mathcal{E}$ for all the locations with $k = 20$. . . 72

4.7 Degree of privacy (ϵ) comparison: (a) for different k's and (b) for 100 users with $k = 20$. 72

5.1 Proposed blockchain-based IoT architecture: A region is transformed into a Voronoi diagram where C_i is the i-th Voronoi cell and the graph inside of it is a local network G_i. '•'s and'−'s represent nodes and edges between the nodes, respectively. B_i refers to the local blockchain in cell C_i. 83

5.2 Illustrated Sensor-Chain:-T_{i+1}: A mobile node moves from cell C_2 to C_1. First, it deletes the copy of local blockchain B_2 from its memory and then downloads B_1 from its peers in G_1^{i+1}. T_{i+2}: local blockchain B_3 does not exist anymore as C_3 is empty. T_{i+3}: as temporal constraint is met, (a) aggregator node from each local network is selected. The selected nodes compute aggregation over their respective local blockchains and generate aggregated blocks. (b) using the aggregated blocks as the genesis, the local blockchains are regenerated. 85

5.3 Evaluation results: (a) Sensor-Chain, (b) conventional, (c) improved-temporal, and (d) spatial blockchains (experiment Settings: number of cells = 50, number of sensors = 1000). 90

5.4 Comparison between Sensor-Chain and spatial approaches in terms of number of (a) cells and (b) sensors. 91

5.5 Key components of Sensor-Chain platform. 92

5.6 Class diagram of Sensor-Chain. 93

5.7 Architecture diagram of Sensor-Chain. 94

5.8 Sequence diagram of Sensor-Chain. 95

5.9 Use case diagram of Sensor-Chain. 96

5.10 Network initialization in the demonstration. 115

5.11 Genesis block in a newly created blockchain. 115

5.12 Transaction broadcasting in the demonstration. 116

5.13 Transactions recorded in a blockchain. 117

5.14 Genesis block after aggregation. 118

6.1 System model of permissioned-blockchain where BC and BA refer to blockchain and blockchain authority, respectively. The BA also acts as certificate authority and trace manager. The mobile IoT nodes are connected with each other in a P2P network using a short-range communication technology. They make transactions with each other and send information on the transactions (e.g., rating about other mobile nodes at a specific location and time) to the nearest blockchain node. Here, each grid refers to a specific location. 123

6.2 Illustrated BlockPriv: The curve refers to a mobile node (MU)'s actual path between l_0, l_1, and l_2 locations at times t_0, t_1, and t_2, respectively. The location l_1 is privacy sensitive for the MU. Thus, it remained silent at location l_1. It will make a blockchain transaction at l_2 at time t_2 only when the number of locations reachable from both l_0 and l_2 in $t_2 - t_0$ time, meets the privacy requirement for l_1. . 130

6.3 Locations in (a) New York City (NYC) and (b) Tokyo (TKY) datasets. Green markers symbolize the locations. The red colors represent the high density regions. 137

6.4 Average loss of utilities versus privacy level in sim-BlockPriv and diff-BlockPriv. 140

6.5 Distribution of loss of utilities in sim-BlockPriv, regarding different privacy levels. 141

6.6 Average loss of utility versus number of sensitive location types (α) in sim-BlockPriv and diff-BlockPriv. 142

6.7 sim-BlockPriv: comparison between the distribution of loss of utility for different numbers of sensitive location types (α) for r =(a) 500 meter, and (b) 2000 meter. 143

6.8 sim-BlockPriv: correlation values (Corr. value) of loss of utility versus privacy level (U-P) and loss of utility versus number of sensitive location types (U-S) for 100 users in NYC and TKY datasets. 145

CHAPTER 1

INTRODUCTION

1.1 Background

Over the past few years, with the advancement of sensing technologies, wireless communication, and embedded systems, we have observed a dramatic increase in the research effort to turn physical objects into smart devices and enable them to talk to each other through the internet; the network formed by these different smart devices is termed as the Internet of Things (IoT) [AIM10]. From predictive maintenance in industries, improving environment monitoring systems, reducing road fatalities by enabling the vehicles to communicate with each other, minimizing errors in decision making in military systems to better monitoring of health, IoT has a bevy of applications in nearly every industry. A variation of IoT is the mobility-centric IoT, which unifies the sensing ability with mobility of the devices. The core component of such mobility-centric IoT systems is the location information of the IoT devices: users share their location information through their devices with a system to get a variety of location-based services (LBSs). Formally, location-based services are "services accessible with mobile devices through the mobile network and utilizing the ability to make use of the location of the terminals"[VMG+01]. The location information of the users, for example, has made location-based recommendation systems a standard part of our daily life. Real-time navigation systems, intelligent vehicle systems, crowdsensing, indoor navigation for blind people, and health monitoring are just a very few examples of location-based services which are benefited from the mobility aspect of IoT. From an economic perspective, mobility-centric IoT has a tremendous impact on the overall GDP. It was predicted that the revenue from location-based service would be as high as $39.43 billion in 2020 [IoT15]. Under-

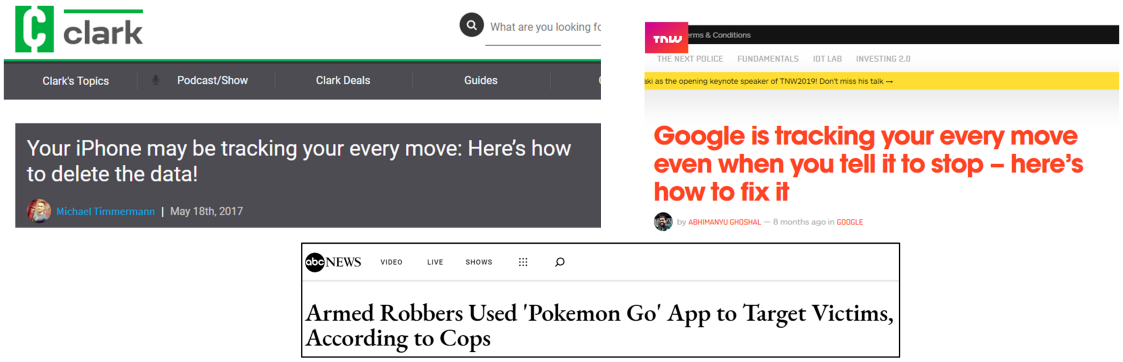


Figure 1.1: Real cases on trajectory privacy issues.

standing the enormous potential of such systems, efforts have been made to connect an increasingly larger number of device to IoT system; according to statistics from Statista, the number of smartphone devices would hit 2.87 billion by the year of 2020[Sta19].

Despite having enormous technological and economic benefits, embracing mobility-centric IoT for location-based services (LBSs) is challenging. One problem is regarding the privacy of the location information of LBS users. IoT systems usually rely on centralized architecture (i.e., cloud server), with a main server responsible for storing, processing, and analyzing all the data sent by the various IoT devices. By gathering all the information from a user and by combining them with other information from a large number of sources, it is possible to reveal sensitive, private information. This type of information is highly valuable, and a malicious system authority may choose to profit by selling the information to third parties, but even in cases where the central authority can be trusted, the sensitive data cannot be assumed secured as centralized architectures are highly susceptible to a single point of failure or external cyber-attacks. Security loopholes in the design of the system or lack of understanding about a variety of threats can make the whole system vulnerable. By exploiting these loopholes, malicious entities can compromise the system

[MF17] or harvest users' sensitive data [Gra18]. Another risk of revealing sensitive spatiotemporal location information is that malicious entities (e.g., robbers) can utilize that information to target potential victims (Figure 1.1). For example, an interview of ex-buglers revealed that burglars use location-based social networks to discover empty houses to burgle. In summary, an IoT system never be wholly trusted and it is necessary to ensure the privacy of an IoT user's data in such a system by design.

1.2 Motivation

This dissertation is motivated by the trajectory privacy issues in IoT systems which are built to provide different kinds of location-based services (LBSs). The research of this dissertation focuses on three different types of LBSs: location query, heterogeneous data sharing platform (e.g. location-based social networks), and aggregation-oriented mobile crowdsensing applications. We first consider an LBS where users send their location information and request to a service provider from their IoT devices (e.g., smartwatch, smartphones, and the on-board unit of vehicles) to get location-centric information(e.g., nearby urgent care, and restaurants having at least 4-star reviews). We can formulate this request or query as follows.

$$\mathcal{Q} = \{\mathcal{U}, l_t, \mathcal{I}\} \tag{1.1}$$

Here $\mathcal{U}$ is the identifier of a user, l_t is the user's location in terms of latitude and longitude at time t, and $\mathcal{I}$ is the requested information. The service provider intelligently generates the best relevant results for $\mathcal{U}$ based on the previously shared location data and other relevant contextual data. A service provider usually stores all the contents from all of its users and generates a predictive model to find out the best results for a particular request; the problem here is that, asides from providing

"honest" results, the models can also be (mis)used to extract sensitive information regarding the user's health status, religious and political views, home and work addresses, their frequently visited locations, and so on.

In addition to sending a request to the service provider, LBS users can also share the location information through check-ins and different types of geo-tagged content. One such example is the location-based social network (LBSN), where users share their location-based experience with other users through location-tagged media content, such as photos, videos, and texts. Formally, "Social Network Sites that include location information into shared contents are called Location Based Social Networks (LBSN)."[RH13]. Both check-ins and geo-tagged contents can be considered as similar to an LBS query as both of them explicitly contain location information of a user and carry the similar location privacy threats. The high temporal correlation between geo-tagged and non-geo-tagged data can also be used to infer a user's location information.

Given the severity of the privacy issues that arise from the shared location information, we have recently observed a surge of research on location privacy [1].The majority of this research has paid particular attention to solving the problem for independent location's privacy preservation, but this focus is shifting towards devising mechanisms to protect the privacy of spatiotemporally correlated locations. In this dissertation, we focus on the problems with (1) ensuring the privacy of frequently visited sensitive locations under spatiotemporal correlation against long-term observation-based attacks, and (2) preserving privacy against an attack based on a combination of geo-tagged and non geo-tagged contents in spatiotemporal domain. We define the notion of "long-term privacy" as follows. Let, $\mathcal{T}_1, \ldots \mathcal{T}_n$ are n number of different trajectories of a user. Each of these trajectories are made up with a set

[1]We use "Trajectory Privacy" and "Location Privacy" interchangeably.

of sensitive (both frequent and infrequent) locations such that $t^i_{last} < t^{i+1}_{last}$. Here, t^i_{last} and t^{i+1}_{first} are the times of last and first locations in $\mathcal{T}_i$ and $\mathcal{T}_{i+1}$, respectively. Then, the long-term privacy preservation refers to preserving privacy of $\mathcal{T}_{i+1}$ in such a way that it will not leak the privacy of any of the sensitive locations in $\mathcal{T}_0, \ldots \mathcal{T}_i$. We observe that when it comes to the issue of preserving privacy against long-term observation, existing approaches either leak privacy to maintain quality of services (QoS) or impose high reduction in QoS, in terms of query drop or delay, to ensure the location privacy [GDSB16, LLL+17]. As such, finding the right balance between the long-term privacy preservation and real-time quality of services is the core problem that motivates the first aspect of this dissertation.

The second problem is mainly associated with LBS scenarios where users share both geo-tagged and non-geo-tagged content. We confine our study to LBSN with two kinds of content: checkins and photos. The location privacy issues with checkins and photos have been studied independently for many years: devising machine learning techniques for identifying a user's location from both indoor and outdoor photos is an active area of computer vision [LCTZ13, TPFF+15, WKP16]. The opposite is also true: preserving visual location privacy against such identifying techniques has also attracted significant attention [MDFFF17, OFS17]. Nonetheless, the synergy between the privacy-invading techniques for check-ins and photos is not addressed well in the literature. In particular, we study the location privacy issues in a scenario, where a trajectory contains spatiotemporally correlated check-ins, geo-tagged, and non-geo-tagged photos, and a malicious entity is unwilling to analyze the contents of the photos of location inference. Formation of such a successful attack model and its defense mechanism is another aspect of this dissertation.

One way of concealing the user is by forming a distributed peer-to-peer (P2P) network without relying on a centralized entity. This idea has been studied at

some extent, and several approaches have been proposed for location privacy protection [STP+14, NLZ+15, GPI16]. However, these approaches utilized the P2P network in an off-the-shelf manner. In reality, tapping into distributed P2P network is not straightforward as it comes up with a bevy of security, trust, and data management issues. We argue that we first need to identify which distributed IoT system can provide security, trust, auditability, and immutability by design. Blockchain technology has emerged as a distributed P2P way of recording digital interactions in a secure, transparent, and auditable way without relying on centralized authority[Nak08, Fra14]. Blockchain is essentially a public ledger of a sequence of blocks that continually grows as newly created blocks are added to record the up-to-date transactions. The meaning of transactions varies with the context of its application. For instance, with cryptocurrencies, it is mainly the amount of currency transacted by two entities in the network. In environment monitoring crowdsensing applications, this could be the value of shared sensor reading. As distributed P2P network is used, each node in the network holds a copy of the blockchain which provides built-in integrity of information, and security of immutability by design, making it very useful for P2P trustless networks composed of a massive number of devices. By design, blockchain provides anonymity, as the nodes can join the network with private and public keys. With these unique set of benefits, blockchain has already gained significant attention from the IoT community to achieve privacy through decentralization. From a broader perspective, integrating blockchain to IoT systems is being considered. Today's centralized architecture is incapable of handling the fast-paced growth of IoT; the frequent change in the mobility-centric IoT network due to node mobility, node failure, damage, energy depletion, or channel fading only further exacerbate the problems of a centralized model. However, the integration of blockchain with IoT is also not straightforward. Since the chain con-

tinues to grow, IoT devices require more and more resources to manage it on their local spaces. Similarly, scalability with constrained computing power and battery also poses a challenge. To be precise, with the integration of blockchain, each node needs to perform a large number of tasks at different stages of the blockchain with their constrained computing power and battery life, and as the network grows this problem becomes increasingly more challenging to address. Hence, we emphasize that, before attempting to preserve privacy with a distributed P2P network using blockchain, we must address its scalability problem for resource-constrained IoT devices, which is another focus of this dissertation.

The majority of the works on blockchain-IoT integration is motivated by bitcoin, the first successful application of blockchain. It is a public blockchain, where the nodes join and leave the network with random public keys and there exists no authority for tracking the nodes. Blockchain has since evolved from public to a permissioned version. The idea of a permissioned blockchain stemmed from the evidence of misuse of freedom in public blockchains for illegal activities. For instance, almost half of the bitcoin transactions are estimated to be related to illegal drug sales, ransomware, and other malicious activities[FKP19]. Organizations are more interested in permissioned blockchain, which would assist them with the power of blockchain as well as having better control over the entities that they interact with. In a such blockchain, there exists an authority which controls who can join the network and do what kind of operations. Hence, unlike public blockchains, trajectory privacy preservation in a permissioned-blockchain is a more robust problem as the authority retains the capability of tracking the nodes. The current privacy-preserving solutions, proposed for the context of permissioned blockchain, can guarantee only conditional privacy[LLC+18]. The problem of preserving trajectory privacy in a permissioned blockchain is the final motivation of this dissertation.

1.3 Research Problems

The overall goal of this dissertation is to devise efficient trajectory privacy preservation mechanisms for mobility-centric IoT systems. To achieve this goal, we aim to understand the different attack models that can be utilized to compromise trajectory privacy, the limitations of existing works against those attacks, and the approaches to improve the existing works to mitigate the identified privacy leakages.

We first study the problem of personalizing trajectory privacy preservation with reduced quality of services (QoS) loss against a long-term observation-based attack. This problem involves modeling of a long-term observation attack and quantification of its privacy leakage, and nullifying the attack with reduced loss of QoS. We extend our study with the problem of understanding how a combination of spatiotemporally correlated geo-tagged and non-geo-tagged contents can affect the existing trajectory privacy-preserving mechanisms (TPPMs) and how can we improve the existing TPPMs to neutralize the effect. This problem involves the design of an inference attack model from the two kinds of contents: quantification of the effect of that inference model on existing TPPMs, and devising appropriate TPPM to mitigate the effect.

The two above problems mainly focus on preserving trajectory privacy in a centralized IoT system. Another drastic way of preserving privacy is the formation of a P2P network of the IoT nodes such that the necessity of an intervening centralized authority can be demolished. Recently, blockchain is being studied as a promising way to decentralize IoT systems. In this dissertation, we explore the problem of making blockchain lightweight for P2P networks of IoT nodes without relying on powerful or expensive edge devices.

We then study the trajectory privacy preservation problem in the context of

permissioned-blockchain, where a malicious authority is capable of tracking the IoT nodes and the P2P interaction between the nodes constitute proofs regarding their location information. The problem includes modeling of a different attacks that can be exploited by the authority and measuring their impact on location privacy of the IoT nodes.

1.4 Research Objectives

To answer the above mentioned research questions, this dissertation aims to achieve the following four objectives: 1. measure the effect of long-term observation attack on trajectory privacy preservation and determine the factors that can mitigate such an attack in a personalized manner while maintaining the loss of quality of services (QoS) within a constraint, 2. assess the impact of a combination of spatiotemporally correlated geo-tagged and non-geo-tagged contents on trajectory privacy without analyzing the contents of the non-geo-tagged contents and formulate and evaluate a mechanism to mitigate the impact, 3. evaluate the relationship of spatiotemporal mobility of the IoT nodes and spatial sharding with the size of blockchain to design a lightweight blockchain framework, and 4. identify and measure the trajectory privacy risks in a permissioned-blockchain and evaluate the influence of a privacy-preserving mechanism on QoS while alleviating those risks.

1. Long-Term Privacy Preservation for Frequently Visited Locations

In a location-based service(LBS), a user queries a service provider by providing his/her precise location information (Equation 1.1) to get a variety of services. As the shared location information may reveal a user's sensitive information, trajectory privacy preservation mechanisms (TPPM) come into play to preserve privacy. A

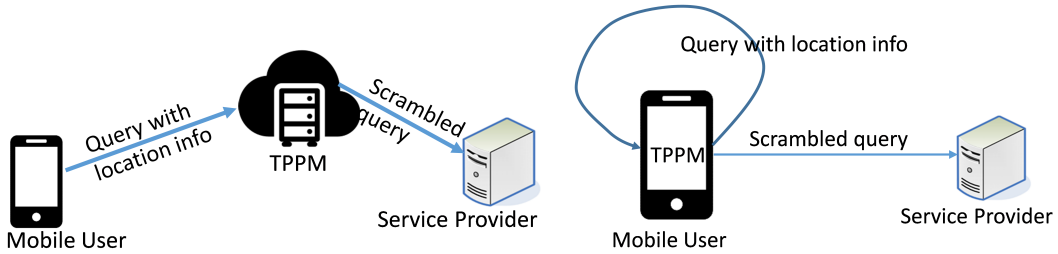


Figure 1.2: Architectures for Trajectory Privacy Preserving Mechanism (TTPM): (a) Trusted Third Party (TTP)-based architecture and (b) User-centric architecture.

TPPM works between the user and the service provider. It takes the query $\mathcal{Q}$ and scrambles it in such a way that it would not leak privacy (at some degree) of the user against a set of testable attacks. There are two types of TPPM architectures: Trusted Third Party (TTP)-based and user-centric. In a TTP-based architecture, the TPPM could be a cloud server or any other external entity. On the user-centric architecture, the TPPM resides on the user's device and does not depend on a TTP in the runtime (Figure 1.2 depicts these two architectures).

At this time, a variety of TPPMs have been proposed, which can be classified into the broad categories of pseudonym, cryptography, and obfuscation. The pseudonym and cryptographic techniques have any of the following limitations: it relies on a TTP-based architecture, its implementation is challenging for resource-constrained IoT devices, and it requires changes in the already up and running LBS system to facilitate privacy requirements. On the other hand, obfuscation techniques can be built upon a user-centric architecture in a lightweight way, and they do not require changes in an existing architecture of an LBS. These advantages make obfuscation techniques highly practical for trajectory privacy preservation in LBS for resource-constrained IoT devices. As such, this dissertation is focused on obfuscation techniques, wherein a user's original location is replaced by a set of locations which look similar to a user's real location based on some predefined privacy measures. Thus,

using an obfuscation-based mechanism, the original query $\mathcal{Q}$ becomes,

$$\mathcal{Q}' = \{ID, < l_1, \ldots l_m >, \mathcal{I}\} \tag{1.2}$$

Here, $< l_1, \ldots l_m >$ is the set of the locations which represents a user's original location. An objective of this dissertation is to design an obfuscation mechanism to achieve the following properties: 1. personalization, 2. privacy preservation of sensitive locations against long-term observation attacks, and 3. a quantifiable relationship between long-term privacy and quality of services (QoSs).

2. Location Inference Attack Based on Spatiotemporally Correlated Geo-tagged and Non Geo-tagged Contents and Its Defense Mechanism

While our first objective is to study privacy issues with homogeneous data-centric interaction between the users and LBS, i.e., check-ins or queries, we extend the work with heterogeneous data-centric interaction. As mentioned earlier in motivation, a heterogeneous data-centric interaction refers to a trajectory, shared by a user with the LBS, which contains both of geo-tagged and non-geo-tagged contents. We confine our study with two kinds of contents: query and photo. In our study, a checkin in an LBS is similar to query as we do not consider the privacy issues with the requested information ($\mathcal{I}$ in Equation 1.1). While queries are always geo-tagged, a photo can either be geo-tagged or non-geo-tagged. In the case of non-geo-tagged photos, the inference attack models are usually constructed based on a machine learning model which analyzes the contents of photos to localize a user. However, we study the problem from a different perspective. Let us consider a trajectory $\mathcal{T}$ which contains queries, geo-tagged and non-geo-tagged photos as follows.

$$\mathcal{T} =< (\mathcal{Q}, t_1), (\mathcal{P}_{geo}, t_2), (\mathcal{P}_{non-geo}, t_3), (\mathcal{Q}, t_4), (\mathcal{P}_{non-geo}, t_5), \ldots > \tag{1.3}$$

Here, the terms $\mathcal{Q}, \mathcal{P}_{non-geo}, \mathcal{P}_{geo}$ refer to a query, non-geo tagged photo, and geo-tagged photo, respectively.

Our objective is to investigate the case of inferring location information without analyzing the content of a photo using a machine learning model where there is a high temporal correlation between two shared contents. The objective also includes the construction of an efficient obfuscation approach to negate the inference in a lightweight and practical way.

3. Spatiotemporal Blockchain Management for Resource-Constrained IoT devices to Achieve Decentralized Privacy

Replacing a centralized LBS architecture with a decentralized peer-to-peer (P2P) network of the IoT devices is a significant step towards achieving privacy preservation in IoT. Questions of efficiency and practicality aside, blockchain is being considered to design a fully decentralized IoT system with (some degree of) built-in privacy. Inspired by the massive popularity of bitcoin, the first successful application of blockchain, a large number of works in the intersection of blockchain and IoT focused on how to implement "bitcoin"-like blockchain in the IoT. To preserve trajectory privacy, the IoT nodes change their private-public key pairs frequently. However, their assumption of having enough computation and storage capacity on the IoT devices to manage a blockchain is mostly impractical. To make blockchain computationally lightweight, some works focused on simplification of the mining process so that power consumption is reduced. To deal with the constrained resource issue, the solution of managing blockchain using fixed infrastructures based on high-end computing devices (i.e., gateways) has been explored significantly. However, for mobility-centric IoT, where the network is spread through a vast region and the topology changes frequently, having a fixed infrastructure is uneconomic.

In a nutshell, all these works either considered that the IoT devices are capable enough to store the blockchain and to perform blockchain operations, or they employ high-end computing devices to manage the blockchain. While privacy can be preserved, these works are impractical for mobility-centric IoT. What sets this work apart is that our objective is to make blockchain lightweight so that the IoT devices are able to store it without relying on external edge computing devices, with a focus on the mobile crowdsensing applications. We study and measure the relationship of spatiotemporal mobility of the users and spatial division of a blockchain into shards with the storage capacity of the IoT devices to hold blockchain on their space.

4. Quantifying Location Privacy in Permissioned Blockchain-based decentralized IoT

We have now come to the point of studying trajectory privacy issues in a blockchain. In a public blockchain, this is "easy": there is no centralized authority of the blockchain and the certificate authority (CA) is independent of the blockchain system, and as a result the devices can change their public and private key pairs frequently to achieve untraceability in the network, but this high level of privacy comes at a price. In a public blockchain, it is difficult to track or find out malicious entities, which creates a major challenge to law enforcement and government agencies. While several tools have been developed for tracing transactions or entities, organizations are more interested in having control over the identities of the entities in the blockchain, leading to the development of permissioned blockchain. In such a blockchain, there exists at least one authority who controls who can join and perform blockchain operations in the network, and that authority provides the IoT users public and private key pairs in exchange for their real identity, which helps them maintain their privacy from their peers in the network. However, it can only

provide conditional privacy as the authority itself can trace the devices. Preserving trajectory privacy against such the blockchain authority is quite challenging. We study the trajectory privacy in a permissioned-blockchain where the IoT devices form a P2P network using short-range communication technology (e.g., Bluetooth). With such a short-range communication, a transaction between two devices generates proof-of-location (PoL) about each other in the spatiotemporal domain which certifies the presence of the devices at a precise location at a particular time in the network. In such a context, we show that there is an essential trade-off between trajectory privacy and utilization of the system. The final objective of this dissertation is to model the trajectory privacy leakages in the considered case of permissioned-blockchain for IoT, devise a lightweight mechanism to mitigate the leakages, and quantify the trade-off between privacy and QoS.

1.5 Research Contributions

To achieve the goals as mentioned above, we first design a delay aware long-term trajectory privacy-preserving obfuscation technique for frequently visited locations under spatiotemporal constraint for location-based services, then we devise a location inference attack from a combination of spatiotemporally correlated checkins, geo-tagged and non-geo-tagged photos and an obfuscation mechanism as its countermeasure, and we introduce a lightweight, scalable blockchain framework based on region division and mobility of the users for crowdsensing applications. Finally, we propose a silence-based privacy-preserving obfuscation mechanism to achieve trajectory privacy in a permissioned blockchain-based IoT. The critical points of our contributions are as follows:

1. Delay Aware Long Term Trajectory Privacy Preservation for Frequently Visited Locations under Spatiotemporal Constraint for Location-Based Services [SPI$^+$18, SPI$^+$19]

We start our research by studying the different location inference attacks based on probability distribution of historical location data, travel time information between locations using knowledge of a map, and short and long-term observation of privacy-preserving queries.

We propose a trajectory privacy-preserving obfuscation approach, coined as "KLAP", to achieve personalization in the process of long-term privacy preservation against the map and historical knowledge, and short and long-term observation-based attacks. KLAP models a user's preference for different locations based on the historical data and can personalize privacy-preservation by utilizing such a model for sporadic, frequent, and continuous LBS use cases. Specifically, KLAP generates a secure Concealing Region (CR) to obfuscate the user's original location and directs that CR to the service provider. It selects a set of locations, similar to the original location in terms of preference and certain spatiotemporal conditions, such that a malicious service provider cannot distinguish the original location from the set. The CR is computed as the convex hull of the set of locations, and the vertices of the CR replaces the user's location in an original query (Equation 1.2). While it can protect privacy against a variety of spatiotemporal correlation-based attacks, the key contribution of our work lies in the introduction of a CR pruning technique that makes it possible to improve the delay between successive CR submissions with a slight compromise of privacy for infrequent locations while maintaining long-term privacy of frequently visited sensitive locations. We evaluate the proposed approach with two real-world datasets, and the experimental results show that it can achieve better efficiency and efficacy compared to existing state-of-the-art methods for the

different cases considered in this work. Part of this work is published in [SPI+18], and the extended version of the work is under submission in [SPI+19].

2. Location Inference Attack From A Trajectory of Spatiotemporally Correlated Geo-Tagged and Non Geo-Tagged Contents and Its Countermeasure [SPIM18]

Inferring location information from a variety of data has long been studied independently: inferring location from geo-tagged contents and inferring location from non-geo-tagged contents. Without loss of generality, we limit our discussion to queries (which are always geo-tagged), and photos (geo-tagged and non-geo-tagged). Unlike many other works, in this dissertation, we intend to create the link between geo-tagged queries and photos and non-geo-tagged photos for location inference attack without relying on machine learning model for the case when the contents have a spatiotemporal correlation. Our work is based on the following hypothesis: the probability of sharing different contents at different locations based on historical data and the high temporal correlation between a set of geo-tagged contents (generated using an existing obfuscation approach) and non-geo-tagged contents allows to infer a user's location at a finer level without analyzing the non-geo-tagged photos. Let us consider a trajectory $\mathcal{T}$ similar to Equation 1.3. To preserve the privacy of the trajectory, let the user uses an obfuscation approach $\mathcal{O}$ which generates $k-1$ number of fake locations and tags them with the real locations. The obfuscated version of $\mathcal{T}$ is as follows,

$$\mathcal{T}' =< (\mathcal{O}(\mathcal{Q}), t_1), (\mathcal{O}(\mathcal{P}_{geo}), t_2), (\mathcal{P}_{non-geo}, t_3), (\mathcal{O}(\mathcal{Q}), t_4), (\mathcal{P}_{non-geo}, t_5), \ldots > \quad (1.4)$$

We show that, if there is a high temporal correlation among the shared of content, a traditional obfuscation approach can leak privacy of a user. We formulate and

implement such an inference attack model on dummy based obfuscation approach using two different factual datasets to validate the hypothesis. To nullify such inference, we also propose a randomized obfuscation approach which generates dummy locations by considering the both of query and photo sharing probabilities for different locations. Our contribution also includes a technique to visually represent the privacy-preserving queries and photos in the case of location-based social network (LBSN). The work is published in [SPIM18].

3. Spatiotemporal Blockchain Management for Resource-Constrained IoT devices to Achieve Decentralized Privacy [SPSK19, SPN$^+$19a]

Making blockchain lightweight for resource-constrained IoT devices has recently garnered some attention, but the influence of the mobility of the devices in designing blockchain is not explored yet. In this research, we focus on specific mobility-related scenarios where a mobile node is not really required to have a "global" view of a blockchain. Let us consider an environment monitoring mobile crowdsensing application where aggregated data (e.g., temperature, humidity, and air quality) from a small region at a particular time is more critical than an individual's data. The mobile nodes at a location may contact each other in a P2P way to collect each other's environmental sensor's value for some time, then one node is selected to send the aggregated information in a particular form (e.g. max, mean, average, and median.). Since the nodes are mobile, the trust value computed for some nodes may not be significant at a different location and time for an individual node, and the environmental data varies from one region to another; thus, instead of having a single network, region-based multiple smaller networks, as well as blockchains, are more feasible. Furthermore, it is not practical to deploy powerful, expensive edge devices over a large region to carry out the blockchain operations. This research aims to

address the blockchain management problem by designing a lightweight blockchain framework, coined as "Sensor-Chain", for mobility-centric IoT without relying on a fixed infrastructure of edge devices. We show that breaking down a traditional global blockchain into smaller "local" blockchains in the spatial domain and limiting their size through a temporal constraint will allow us to design scalable blockchain for mobile IoT systems. Furthermore, the Sensor-Chain allows the mobile devices to control their storage space on the run. The highlights of our contribution are as follows: 1. The Sensor-Chain blockchain framework consumes little storage space on the IoT sensor devices and is scalable with the increase in network size. We compare the performance of sensor-chain with three (3) other schemes and the results on the relationship of spatial and temporal constraints with the size of blockchain justify its advantage for aggregation-based mobile crowdsensing applications. 2. The proposed framework does not involve any fixed positioned powerful edge devices, which makes it more flexible with a variety of mobility-based IoT applications. 3. Sensor-Chain is independent of any particular ledger platform. Thus, it can be implemented with any platform (e.g. Ethereum, hyperledger, and so on) for IoT. The work is accepted for publication in [SPSK19]. A demonstration on the development of the framework is accepted in [SPN+19a].

4. Quantifying Location Privacy in A Permissioned Blockchain [SPN+19b]

Unlike many other works on preserving trajectory privacy for mobility-centric IoT using a public blockchain, we study the problem in the context of private or permissioned blockchain for mobility-centric IoT. We consider a permissioned blockchain which is governed by at least one authority: the authority provides the user with private and public key pairs to achieve privacy from the peers in the network. However, as the authority knows the real identity of each user, it is possible to map the users

in the spatiotemporal domain, which leads to potential trajectory privacy invasion. We study this problem in the context of two properties: there exists a spatiotemporal correlation between consecutive blockchain transactions, and there presents a (either explicit or implicit) Proof-of-Location (PoL) protocol in the system such that a node cannot fake its location in the network. In this work, a transaction is considered as the atomic blockchain operation in the network. To understand the time reachability based correlation, let the variable O_u^t represent the actual location l_u of a mobile IoT user u at time t. Given the user's locations (l_{i-1}, l_{i+1}) at times (t_{i-1}, t_{i+1}), the user's probability to be at a location l_i at a discrete time t_i is $Pr_u^i(l_i)$. This probability, $Pr_u^i(l_i)$, can be computed using the time reachability correlation as follows:

$$Pr_u^i(l_i) = \mathbb{P}\mathrm{r}(O_u^i = l_i | O_u^{i-1} = l_{i-1}, O_u^{i+1} = l_{i+1}) \tag{1.5}$$

Note that $Pr_u^i(l_i) = 1$ if l_i is reachable to and from l_{i-1} and l_{i+1} in time $(t_{i+1} - t_{i-1})$. Otherwise, $Pr_u^i(l_i) = 0$. A PoL is a digital certificate which confirms the presence of a user at a certain time and location. The Spatiotemporal correlation between two blockchain operations is defined based on the time reachability relationship between the locations exposed by the two transitions. We first show that existing obfuscation approaches, designed to protect trajectory privacy in centralized IoT systems, cannot be implemented in a plug and play way in permissioned-blockchain under the presence of a PoL. This leads us to the realization that it is not possible to achieve privacy while maintaining the utility of the system. In other words, there is an important trade-off between location privacy and utilization of the system under a PoL. We develop our solution based on the hypothesis that to protect the privacy of a sensitive location, a mobile user must keep silent in the network. However, remaining silent infinitely results in location privacy of 100% but a system utilization of 0%. On the other hand, if the user uses the system at a nearby insensitive location

for a short time, then using the time reachability based spatiotemporal correlation, defined by Equation1.5, a malicious authority can reduce the number of potential locations as a mobile user's real location. Thus, we formulate our problem as a two-objective optimization problem: one objective is to remain silent to minimize the number of locations with $Pr_u^i(l_i) = 1$, and the other is to maximize the utility of the system. To solve the problem, we propose BlockPriv, a silence based obfuscation approach, which quantifies the relationship between privacy and utility to protect sensitive locations' privacy dynamically. We analyzed different security, privacy, and utility aspects of BlockPriv, both theoretically and experimentally, with its implementation. The work is accepted for publication in [SPN+19b].

1.6 Dissertation Outline

The outline of the rest of the dissertation is as follows: The review of the related works is presented in chapter 2. The detail of our proposed delay-aware obfuscation mechanism to preserve trajectory privacy against long-term observation-based attacks is detailed in chapter 3. Chapter 4 presents our designed inference attack model from a combination of spatiotemporally correlated geo-tagged and non-geo-tagged contents and its countermeasure. Chapter 5 discusses the proposed lightweight blockchain framework for mobile IoT devices. Chapter 6 covers our work on trajectory privacy preservation in permissioned-blockchain. Finally, chapter 7 concludes the dissertation with a discussion on the limitations of the proposed works and different directions for our future works.

CHAPTER 2

RELATED WORK

In this chapter, we delineate an overview of the contemporary works related to this dissertation. We start with a presentation on the related works of trajectory privacy preservation in location-based services (LBS) from the perspective of homogeneous interaction between a user and LBS. The discussion includes different location inference models proposed in contemporary works and different classifications of the existing trajectory privacy preservation mechanisms (TPPM) according to their characteristics. We then review the TPPMs on preserving trajectory privacy against inferences based on heterogeneous data. The review covers inferences for checkins and (geo-tagged and non geo-tagged) photos, two of the most common means of interaction in LBS, more precisely in location-based social network (LBSN). Another innovative and bold way of improving location privacy is the decentralization of the system through blockchain. This chapter covers a review of the research on blockchain-based IoT systems, their achievements and limitations. The review is largely focused on the management of blockchain in a lightweight manner for resource-constrained devices; as such it portrays a classification of the blockchain based on their reliance of the type of architecture. Finally, the chapter reviews the research on preserving location privacy in blockchain, with a concentration on permissioned-version of it.

2.1 Location Privacy Preservation in Location-Based Service (LBS)

We first discuss the related works on location inference models which underline the different threats associated with location information sharing. Then, we go trough the different approaches that have proposed so far to preserve the location privacy against adversarial inferences.

2.1.1 Location Inference Models

Identification and Understanding of the different threats that can damage the privacy of location information is an active research area for quite a while. These threats include figuring out home and work places, sensitive information, and de-anonymization of database. Majority of these works focus on the explicit location information of the users (e.g. checkin, gps data). Krumm et al.[Kru07] analyzed two-week GPS tracks from 172 known individuals to infer users' home address. Cheng et al. [CCLS11] analyzed 22 million checkins of 220,000 users from different location-based social network, including Foursquare, UberTwitter, Gowalla, and Gravitiy. The authors studied spatial and temporal aspects of the checkins, mobility patterns, different factors (e.g. social status, geographic and economic constraints) motivating the mobility, and frequently visited locations. Recently, an interesting inference model based on visual technique is proposed by Liccardi et al. [LARC16]. Their results show that it is possible to infer workplace and home addresses at high accuracy without using a complex algorithm. A machine-learning method for determining the motivation behind check-ins has been developed by huguenin et al.[HBM+18]. They analyzed a large dataset from Foursquare and results show

that their proposed model can achieve accuracy as high as 63%. Drakonakis et al.[DIIP19] improved the location inference models further by designing a heuristics based on social and behavioral norms of users in twitter checkins. Their approach achieved 92.5% and 55.6% accuracy in identifying user's home and workplace addresses. Olteanu et al.[OHS+17] showed that even if individual's location data is protected, the co-location data with other users can be used to leak user's privacy.

The study of location privacy leakage from non-geo-tagged data has gained wide attention recently. To solve the problem of identifying locations from a non-geo-tagged photo, the models are usually trained a machine learning model with a large dataset of tagged photos[LBTC+15a, LBTC+15b, WKP16]. Using such a model, given a photo, it is possible to detect the place where the photo is taken. For instance, Google supports search by photo [goo], where a photo, without any geo-tagged is given as an input, the search engine gives the name of the place where the photo was taken. Similarly, location privacy leakage through linguistic analysis of textual data (e.g. hashtags in tweets in twitter) has also gained significant attention very recently [Zha19, RKH+19]. As an example, Rusert et al. [RKH+19] developed a spatiotemporal Naive Bayes classifier to find out location-related hashtags in Twitter.

While all these inference models underline the location privacy issues in different domains, a major problem is that, they are studied independently. The impact of joining all these models to develop a more comprehensive inference model and its impact is not studied yet.

2.1.2 Location Privacy-Preserving Approaches

Privacy-preserving approaches for geo-tagged data or simply location information, can be categorized into two major classes: 1) trusted-third party(TTP) based centralized approaches, and 2) user-centric or decentralized approaches. In centralized approaches, a TTP collects all the users' data to provide privacy. In contrast, user-centric approaches do not require middleware. Instead, it operates and stores the required data on the user's devices. In this dissertation, we focus on the following categorization of the existing techniques: 1) pseudonym, 2) k-anonymity, and 3) obfuscation.

Pseudonym is one of the earliest privacy-preserving approaches which replaces the user's real id with another one from a pool of ids[GPI15, YMH15, WYGG18, WY19]. This pool of ids is managed either by a TTP [BSM16] or by the collaboration of the users of the system. In a TTP based approach, a user (e.g., smart vehicle) registers itself with the system by providing its real identification and gets a set of pseudonyms in return [PSFK14]. Another variant is the mix-zone where certain spatial regions are designated for pseudonym change (e.g., gas stations) [LLL$^+$11, BSM16]. For instance, in [BSM16], the vehicles are required to follow a certain driving pattern and change their pseudonym in the mix-zone. These approaches reduce the flexibility of the privacy-preservation approach as they are constrained by the underlying infrastructure (e.g., structure of the road network). Also, an attacker can perform statistical linkage attack by exploiting the knowledge on the infrastructure to undermine the pseudonym changing approaches. In distributed approach for pseudonym changing, the users collaborate with each other for sharing their pseudonyms [GMS$^+$13, PL15, YKH$^+$16, ZLS$^+$16, PLMW17, GMG18]. Such an approach causes high computational and communication overhead in the devices. Besides, the success of collaboration depends on the non-adversarial behavior of

the collaborating users. Another limitation of pseudonym schemes is their limited applicability. Many systems require the users the reveal their real identity to the system. In such a case only conditional privacy is attainable.

Another popular approach is the *k-anonymity* in which user's real location is made indistinguishable with other $k-1$ users' real locations [GL08]. Several variations of k-anonymity have been proposed in literature, including l-diversity[MKGV07] and personalized k-anonymity[GL08]. However, similar to pseudonym approach, the success of these approaches either depends on an TTP or honest collaboration of the users. The TTP, also known as anonymizer, can be an untrusted entity who can undermine the privacy of its users. The TTP can also collude with other malicious entities to achieve financial gain. In case of collaborative k-anonymity, some users can be unconcerned about their privacy and can conduct location injection attacks by faking their location to the service provider. To address these problems, several solutions have been proposed in literature, including providing incentive [YFX13, ZTZ16, WLYD17] and cluster selection based on mobility pattern similarity[ZLZ$^+$18].

A major improvement of vanilla k-anonymity is the *dummy* approach in which $k-1$ number of dummy locations are directed with the real one to a service provider without relaying on a TTP. However, majority of the existing methods generate the dummies at random [LJY08]. *V-circle, V-grid*[NZLL14], *DLS, enhanced-DLS*[NLZ$^+$14] methods improve this limitation by considering probability of submitting queries from locations. These approaches further improved by *MaxMinDistDS* [CS16] and *k-DLCA* [LHA$^+$16] with the introduction of l-diversity over the set of selected dummies. However, most of the existing methods consider only single queries; thus, cannot protect privacy for frequent and continuous queries. Recently, Liu et al. [LLL$^+$17] propose a dummy generation algorithm based on spatiotemporal cor-

relation of the locations. Specifically, they consider time reachability, direction similarity, and in and out degree. However, this approach is computationally expensive and it cannot protect privacy for frequently visited places.

Another alternative is the *Spatial Obfuscation* which replaces user's real location with a larger concealing region, CR [ACdVS11, LJY08, SJZ+17, GDSB16, AHHH16]. Earlier methods concentrate only on minimum privacy area requirement for a CR, [LJY08, ACdVS11] without considering any information related to users and therefore, are highly vulnerable to attacks based on such information. A promising approach is the PROB framework [DBS10] which translates the locations into features and allows users to assign sensitivity levels for each feature. It enlarges the user's region until sensitivity is reduced to a certain threshold. This framework evaluates the privacy level in terms of region area and thus some locations can be sorted out using information of map or historical data. Another interesting method is the n-CD framework [LSTL13], in which the user's circular region of interest (ROI) is divided into n equal sectors and each sector is covered with a concealing disk (CD). It measures the privacy only using the area of the intersecting regions of all CDs. Although a variation of n-CD method is proposed [LHA+16], none of them take into account user's individual information to evaluate the achievable privacy from that region. Recently, Ghinita et al. [GDSB16] extended the PROBE framework for continuous queries by considering user's maximum velocity in free space to generate CRs. In the case of successive queries, if it is not possible to submit a CR in current time, they propose either to delay CR submission or submit a CR without containing the user's current location but closer to the previous CR. However, both methods reduce the quality of services(QoS) with high delay or failure to satisfy user's query. A promising obfuscation method is proposed by Ağir et al.[AHHH16], which uses Markov chain to generate the obfuscated region. However, with the reduction

in the location precision, obfuscation mechanisms lower the quality of services as well. In recent years, differential privacy[Dwo11] has gained ground in trajectory privacy-preserving obfuscation approach design. Andreś et al.[ABCP13] proposed Geo-Indistinguishability where statistical noise from Laplace distribution is added to the actual location to obfuscate it and ensures that an noisy location has the same probability to be generated from geographically close any two locations. Most important difference between differential private mechanism and other approaches is that it is an property of the mechanism not of the output. Differential privacy for location privacy is further improved by taking into consideration the temporal correlation between locations [XX15, CYXX17], and geographic and semantic features of real location traces [BS16]. Yu et al.[YLP17] proposed an personalized differential privacy preserving mechanism by combining both geo-indistinguishability and adversary's expected location inference error.

If we look into the location privacy issues with non-geo-tagged data, there have been proposed some solutions to deal with such privacy issues. Li et al.[LSL$^+$19] proposed HideMe plugin to intelligently hide a person's photo before it is uploaded to the social network. In the process, the authors considered temporal, spatial, interpersonal and attributes of photo sharing and combine it with face matching algorithm. The hiding or Perturbing portion of the image [HLB$^+$15] is in fact one of the most studied solution in privacy-preserving photo sharing. Moving forward, in case of privacy leakage from textual data, some solutions have also been proposed to patch the privacy leakage. Tagvisor[ZHR$^+$18] proposed three obfuscation approaches for protecting location privacy in case of hashtags: deletion or not publishing the hashtags, replacement of the original hashtag with another one, and replacing the location information with semantically broader category.

2.2 Blockchain for Internet of Things (IoT)

Understanding the limitations of the centralized model of IoT, recent research has shifted to develop decentralized architecture based on blockchain. The existing research efforts can be categorized into devising approaches to integrate blockchain into IoT using existing robust storage and computing resources [XZN+17, PWH+18, XFW+18, WvB14]. The research approaches have focused on node authentication and access control [NXN+07, ZN+15, Axo15, NXN+18, XCBC18, Nov18, OBS18, XNC+19, XCBC19], scalable data provenance methods[SRK+17, TSL+19a, ASN+19], trust management [MDB17, DGP17, YWN+18, AKKH18], information sharing framework for network systems [ARNK19], different security vulnerabilities in blockchains and their countermeasures [CPNX19, SSN+19, TSL+19b, SNKM19], and providing decentralized privacy in IoT systems [RNKK18, AMM+18, CVPC19, DKJG17]. The majority of these works have one thing in common: they either simply considered that IoT devices are equipped with enough storage and computing resources to hold and process blockchains, or utilized high end edge computing devices to manage the blockchain. The assumptions of having enough resources is hard to get on with IoT devices, making the applicability of the research works based on such assumptions questionable. For instance, trust and authentication management for wireless sensor networks using blockchain was proposed in [MDB17] without hinting how the sensors will manage the blockchain on their own local space. Likewise, the Block-VN architecture for distributed transport management system [SMP17], based on a permissioned blockchain, considered that at least some portion of the vehicles are capable of storing and processing an ever-growing blockchain. Another example is the IoT-based Machine-to-Machine payment system, known as IOTA [Fou18]. IOTA uses proof-of-work consensus protocol, which makes the new block creation

task both computationally expensive and time consuming. Thus, in IOTA the hardware requirement is too high and it is hard to meet such requirement for IoT sensor nodes.

Realizing the resource issues of the IoT devices, many research works proposed to offload the blockchain onto edge computing devices. The SpeedyChain[KXN+18] data sharing framework for intelligent vehicles suggested to use roadside infrastructure units (RSIs) to maintain blockchain. The RSIs are responsible for trust and authentication management, and trusted vehicles, verified by the RSIs, can append block to the blockchain. In a similar way, a Roadside Units (RSU) based blockchain trust management for vehicular network was proposed in [YYL+18]. In this work, each vehicle generates a rating for its neighboring vehicles and share the rating with nearby RSU. With all the most recently received ratings, RSUs calculate the trust value offsets of involved vehicles and gather these data into a block. In order to insert the new block into the blockchain, the authors proposed a combination of proof-of-work and proof-of-stake, improving each other. In contemporary works, Xiong et al.[XZN+17, XFN+17] proposed to deploy multiple access mobile edge computing devices to carryout the computationally expensive proof-of-work and introduced game theoretic approach for edge computing resource management. In these works, the sensors are considered as ordinary nodes, and the edge devices are responsible for the blockchain operations. The "EdgeChain" framework [PWH+18] extended this idea by introducing credit-based resource management system to control the edge server resource consumption by an individual IoT device. In [Nov18], a smart contract-based access mechanism was put forward with the aim of simplifying the process of blockchain management and reducing the communication overhead between the nodes. In this mechanism, the IoT devices are kept out of the blockchain as they cannot hold a large blockchain. Rather, a special node called management hub is

proposed to put as a link between IoT devices and blockchain. A blockchain framework was proposed for smart homes[LMNZ18], where the information produced by smart home devices are stored in the blockchain. In this architecture, the blockchain is maintained in the gateways and is isolated from the devices. Similar to the other works on blockchain based Internet of Vehicles, kang et al. [KXN+18] also considered RSUs as edge computing infrastructures for blockchain management. This approach utilized a modified Delegated Proof-of-Stake (DPoS) consensus scheme where instead of stake-based voting, reputation is used for miner selection.

It is evident that all these approaches tried to solve the storing and processing heavyweight blockchain problems by employing more powerful computing devices in the architecture. However, such structured deployment is hardly achievable, as the network topology is prone to changes very frequently in many IoT scenarios.

One viable solution to make blockchain "manageable" for sensors without using any edge or other devices is limiting the size of the blockchain. The "temporal blockchain" [DOA16] proposed a solution based on such concept. It was proposed to delete all the blocks older than a preset period (e.g. 30 days old). While this approach can reduce the size of the blockchain, it still lacks in guaranteeing limited storage capacity with the growth of the network in the long-run in IoT scenario. Moreover, how to deal with the loss of information due to the deletion of blocks was not addressed.

This study highlights that existing blockchain frameworks lack a clear understanding of the resource management issues for blockchain in IoT scenario. Lack of such understanding makes the frameworks highly impractical for IoT. The research on blockchain and IoT has a long way to go, and we emphasize that before taking further steps, we must have an efficient approach to make blockchain lightweight and scalable for IoT sensors.

2.3 Location Privacy Preservation in Blockchain

The goal of the existing trajectory privacy-preserving mechanisms (TPPMs) is to apply them to a node's current location before revealing it to the central authority. As an example, In case of pseudonym, before revealing the location, the mechanism changes the id of a node to make it untraceable [YMH15]. These approaches depend a trusted third party (TTP) to carry out the pseudonym changing steps. This is similar to the mixing approach[BNM+14] used in blockchain to improve privacy by changing the public key of a mobile node with a random public key such that the probability of linking multiple transactions will be reduced. However, in a permissioned version blockchain where short-range communication between the mobile devices form PoL for their whereabouts in spatiotemporal domain, such approach will not work.

Perturbation mechanisms, such as geo- indistinguishability [ABCP13], add statistical noise to a node's real location. Obviously, under a PoL, such mechanisms have limited impact [MDS+18]. On the other hand, spatial obfuscation reduces the precision of the actual location before releasing it to the authority. This is either done by infusing more locations or replacing the actual location with a realistic larger region. Similar to location perturbation, location obfuscation works only at a limited scale under the PoL. In a nutshell, the existing privacy preserving mechanisms, designed for centralized IoT systems, cannot be applied directly in the problem that we are trying to solve here.

In the scope of blockchain, the frequent change of public keys is the most explored solution to preserve the privacy[ZN+15, DSKJ17, SK17]. As an illustration, Dorri et al. [DSKJ17] proposed Lightweight Scalable Blockchain (LSB) architecture for smart-vehicle ecosystems. Here, each node uses a fresh unique public key

while communicating with other nodes to prevent linking attacks. In blockchain based centralized proof-of-location (PoL) generation, Brambilla et al.[BAZ16] also proposed to change the public keys frequently to preserve node's sensitive location privacy while generating proof of locations. Michelin et al. [MDL+18] proposed permissioned blockchain based SpeedyChain framework for vehicular network scenario. Similar to most of the other works in this context, SpeedyChain considers the fixed positioned roadside infrastructure units (RSIs) as the key to maintain the blockchain. Different from bitcoin or ethereum like blockchains, here, for each vehicle there exist exactly one block in the blockchain. In order to maintain privacy, this framework proposes the timely changing the public key of each vehicle. However, these frameworks do not fit completely into the scenario, considered in this paper, where the authority of the blockchain controls the private and public key distributions to the mobile nodes in the system.

From the perspective of efficacy, it is found that changing the public keys is not quite as bulletproof as expected[KKM14, BKP14]. The deanonymization of bitcoin users have gained significant attention from both law enforcement and security and privacy communities. As an unregulated market, it is estimated that almost half of the bitcoin transactions are related to illegal activities (e.g. illegal drug sales, ransomware)[FKP19]. Research efforts show that it is possible to map the public keys of bitcoin users to their unique identities (e.g. Ip addresses) [KKM14, RDJK18]. Such as, Roulin et al. [RDJK18] proposed an deanonymizing algorithm by exploiting only the input and output transactions of mixing services and identified a relationship between the input and output addresses at a very high accuracy. Recently, Dorri et al. [RDJK18] applied decision tree algorithms on smart home devices' data (e.g. smart things, nest smoke alarm) by utilizing off-chain information to classify IoT devices for understanding user's activity pattern. While the work is done in

smart home scenarios, similar to inference and and deanonymization can be done in the context of the mobility of the IoT devices. All these deanonymization works highlight that simply changing the public keys frequently is not the ultimately solution to provide privacy in the blockchain, even in the public version it.

Moving forward, our work is focused on a authority-based permissioned blockchain where privacy is tougher to acheive by default. It is closely related to the work proposed by Li et al.[LLC+18] for the context of vehicular network. Using their proposed framework, it is possible to achieve only conditional privacy as the trace manager can track anyone at anytime, if necessary. Similar to this, Yang et al.[YYL+18] presented a blockchain-based decentralized trust management framework for vehicles where each vehicle is registered with the system using its VIN number. Thus, it is only possible to achieve conditional privacy in this framework. Likewise, Sharma et al. [SC18] proposed a permissioned blockchain by incorporating traceability feature while maintaining privacy in Internet of Vehicle (IoV). However, they used a server for vehicle registration which would store all vehicle id in encrypted scheme and central authority can track any vehicle when needed.

To achieve complete location privacy, Yang et al. [YZL+19] proposed an obfuscation approach to protect location privacy in private blockchain for crowdsensing applications. In this work, a worker submits an obfuscated region to the system to protect exact location's privacy. However, in case of P2P communication of the nodes, such approach cannot be applied without the collaboration of the nodes. Jia et al. [YYL+18] designed a blockchain-based incentive mechanism for crowdsensing applications with a focus on preserving location privacy of the users. In their framework, a confusion layer was proposed in which a user's location is encoded in such a way that it can be confused with $k-1$ other users' locations. While this could be a solution to protect location privacy, it requires honest collaboration of other users.

CHAPTER 3

PRIVACY PRESERVING MECHANISM FOR CONTINUOUS LOCATION SHARING IN CENTRALIZED INTERNET OF THINGS (IOT) SYSTEMS

The ubiquitous use of Location-Based Services (LBS) through smart devices produces massive amounts of location data. An attacker, with access to such data, can reveal sensitive information about users. In this chapter, we study different location inference attacks based on the probability distribution of historical location data, travel time information between locations using knowledge of a map, and short and long-term observation of privacy-preserving queries. We show that existing privacy-preserving approaches are vulnerable to such attacks. In this context, we propose a novel location privacy-preserving approach, called KLAP, based on three fundamental obfuscation requirements: minimum k-locations, l-diversity, and privacy area preservation. KLAP models a user's preference for different locations based on the historical data available on the LBS and can attain personalized privacy-preservation by utilizing such a model for sporadic, frequent, and continuous LBS use cases. Specifically, KLAP generates a secure Concealing Region (CR) to obfuscate user's location and directs that CR to the service provider. For the first time, we propose a CR pruning technique to improve the delay between successive CR submissions significantly. We evaluate KLAP with two real-world datasets, and experimental results show that it can achieve better privacy, reduced delay, lower communication, and storage cost compared to existing state-of-the-art methods. The highlights of our contribution are as follows: 1. An obfuscation approach, named as KLAP, is proposed to protect location privacy against attacks based on the probability distribution, personal context, real-time traffic information, and short and long-term observation of obfuscated trajectories. 2. A concealing region (CR) prun-

ing technique is proposed which presents a significant improvement over delay-based approaches with a negligible reduction in privacy. To the best of our knowledge, this is the first technique to deal with the delay in privacy-preserving Location-Based Services. 3. We carry out a rigorous experiment with two real-world datasets provided by Foursquare for two cities (NYC and Tokyo). Experimental results show that KLAP can achieve better privacy, communication, and storage efficiency for sporadic, frequent, and continuous queries compared to existing approaches.

This chapter is organized as follows. section 3.1 presents the background of the problem that we are trying to solve in this chapter and section 3.2 discuss the detail of the problem. Sections 3.4 and 3.5 present the detail of the proposed privacy-preserving mechanism and its security analysis, respectively. The experimental analysis is covered in section 3.6. Finally, the chapter is concluded in section 3.7.

3.1 Introduction

Location-Based Services (LBS) have become an integral part of our smart life. According to Statista, the number of LBS users in the U.S. is approximately 197 million, with expected growth to 242 million by 2018 [Sta17]. A typical example of LBS use is point of interest (POI) search to obtain locations of restaurants, gas stations, movies, location-based coupons, and other consumer information. The query process poses severe threats to user's privacy which could lead to flooding users with personalized innocent location-based advertisements to more severe exposure such as revealing a user's interests, home address, relationships and more. For example, Liccardi et al. [LARC16] used a visual technique on Twitter data to show that people's most frequent and private locations, such as work and home, can be

deduced using only a small sample of location points (1-day worth). This highlights the vulnerability of location privacy in LBS, making this a highly critical issue.

Many approaches have been proposed in the literature to preserve users' privacy in LBS, including the class of user-centric approaches, which work on user's devices (e.g., smartphones, on-board unit (OBU) of smart vehicles, and smartwatches). One popular user-centric privacy-preserving approach is the *position dummy* which directs $k-1$ fake locations along with the original location to the LBS [HSI+16, LJY08]. Therefore, the maximum achievable privacy is $\frac{1}{k}$. However, the majority of the proposed dummy based approaches focus on single queries and do not consider the historical information related to users. Consequently, these approaches are vulnerable to a variety of attacks derived from historical and spatiotemporal information. Recently, a good number of works to generate dummy location using the notion of differential privacy have been proposed, which aim to make the dummy generation process 'independent' from the attacker's knowledge[ABCP13, BCP14]. However, they are still vulnerable to inferences based on historical data and spatiotemporal correlation. Another category of user-centric approaches is the disclosure of locations with reduced precision, called *location obfuscation* [LSTL13, XX15, SJZ+17, GDSB16]. The idea is to replace a user's precise location information with a larger region, called the concealing region (CR), with a compromise of Quality-of-Service (QoS). Unlike many other approaches, obfuscation approaches can be implemented in a variety of applications, including real-time navigation, POI search, and social network check-ins[LARC16]. However, similar to other approaches, existing obfuscation techniques also cannot guarantee strong privacy. In this chapter, we study the limitations of existing obfuscation-based location-privacy preserving approaches and proposed an obfuscation technique which is comparable with existing approaches regarding privacy preservation, delay management, and communication.

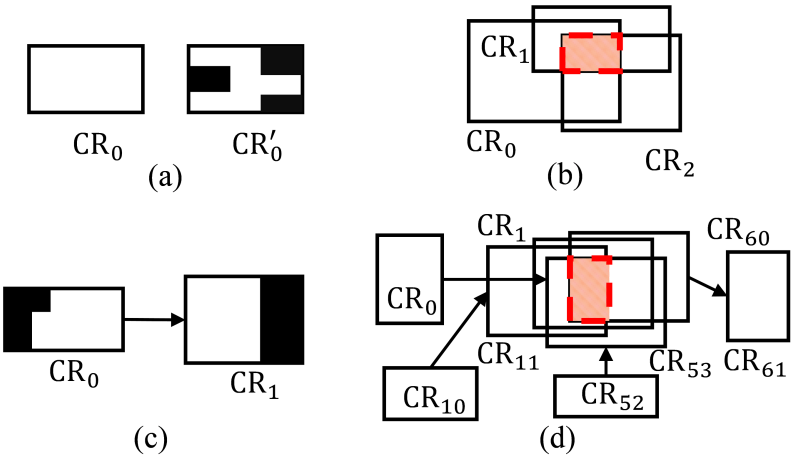


Figure 3.1: Location inference attacks considered In this chapter: (a) Map, probability distribution, and personal context linking attacks on single query, (b) region intersection attack on multiple short-term queries, (c) Real-Time Traffic Information (RTTI) based time-reachability, or maximum movement-boundary based attack, and (d) long-term obfuscated location tracking attack.

3.2 Motivation and Problem Statement

In this chapter, we assume an adversary has access to the following information. Information 1: user's historical queries to a Location-Based Service. Information 2: Knowledge of the map and Real-Time Traffic Information (RTTI) of the road network. Using this information, we study the following problems of existing obfuscation methods against such an adversary. **First**, many of the existing obfuscation techniques generate CRs at random, without considering any prior related to the user[LSTL13, SJZ+17]. Thus, from a given random CR_0, an adversary with Information 1 can exclude some locations (the dark regions in Figure 3.1(a)) for having very low probability to be the user's location and finds a smaller region, CR'_0. **Second**, majority of the methods focus on single query, and thus fail to provide privacy for multiple queries. Let us consider again, a user submitted multiple queries with different CRs in a short-period of time (Figure 3.1(b)). The time difference between the queries are small enough such that an adversary can guess with high

confidence that the user's location did not change significantly between T_0 and T_2. Thus, it can compute the intersection of those CRs and find a smaller region(the shaded region in the figure) containing user's real location. **Third**, the assumption of maximum velocity based free space movement[GDSB16] is unrealistic, as user's mobility is bounded by road networks and its condition in spatiotemporal domain. In figure 3.1(c) a user issued two CRs, CR_0 and CR_1, at time T_0 and T_1, using a maximum velocity based movement method [GDSB16]. However, using information 2, an adversary can exclude some locations in CR_0 from which it is not possible to reach any location in CR_1 in $(T_1 - T_0)$ time interval. It can also exclude some locations from CR_1 to which it is not possible to reach from any location in CR_0. **Fourth**, they cannot guarantee privacy for frequently visited locations in the long run. Let us consider four different obfuscated trajectories were published in $\langle T_0, T_1\rangle$, $\langle T_{10}, T_{11}\rangle$, $\langle T_{52}, T_{53}\rangle$, and $\langle T_{60}, T_{61}\rangle$ timestamp (Figure 3.1(d)).Through careful observation, the adversary can find that the CRs at T_1, T_{11}, T_{53}, and T_{60} were generated for a single location. Thus, it can compute the intersection of those CRs to get finer detail of that location. Motivated by these four limitations, in this dissertation, we seek answer to the following question,

How can the personalized historical mobility information, information on the privacy-sensitive locations, and a visit frequency of the locations will help to design an obfuscation mechanism to prevent against long-term observation attack with improved quality-of-service (QoS)?

We hypothesize that, modeling of a user's preference to different locations based on his/her historical data, a classification of the locations based on his/her visit frequencies, and a controlled relaxation of privacy requirements for infrequent locations will allow an obfuscation mechanism to mitigate in the process of privacy-preservation against long-term observation attack with reduced QoS loss.

Table 3.1: Table of Symbols

Symbol	Description	Symbol	Description
$\mathcal{O}_{\mathrm{A}}$	User's location at time $\mathrm{T_A}$	$\mathcal{P}_i$	Location i's popularity
$\mathcal{B}_i^u$	User u's bias towards location i	$\mathcal{S}_i^u$	user u's preference for i
$\mathcal{R}$	radius of privacy generating region(PGR)	α	privacy area factor
δt	Delay tolerance threshold	λ	privacy level deviation control factor
CR	Concealing region	W	List of related locations to construct CR
$\mathrm{E_{CR}}$	privacy level of a CR	$type(i)$	type of location i
r	radius of Region of Interest(ROI)	C	A CR's vertices

3.3 Background

In this section, we first discuss some necessary concepts related to the work presented In this chapter. Then, we present the proposed system model, including the attack strategies of an attacker to undermine trajectory privacy. The important symbols used in this chapter are listed in table 6.1.

3.3.1 Fundamental Concepts

1. User's Preference, $\mathcal{S}_i^u$: A user u's preference $\mathcal{S}_i^u$ for a location i depends on two factors: *popularity* of the location [NLZ+14], and user's *bias* in favour of the type of location i. We define preference $\mathcal{S}_i^u$ as follows,

$$\mathcal{S}_i^u = \text{Popularity, } \mathcal{P}_i \times \text{Bias, } \mathcal{B}_i^u \tag{3.1}$$

$$\text{Where, } \mathcal{P}_i = \frac{\#\text{of queries from } i}{\#\text{of queries in the LBS}}$$

$$\mathcal{B}_i^u = \frac{\#\text{of } u\text{'s queries from all the locations of type}(i)}{\#\text{of queries in the LBS}}$$

2. Accessible, Related, and Reachable Locations: A location i is ***accessible*** if $\mathcal{P}_i > 0$; otherwise it is ***inaccessible***. An accessible location i, is ***related*** to a user u, if, in terms of preference, it is considered to be close to the user's locations and is used to present user's location in the CR. A location B is ***reachable*** from location A if it is possible to reach B from A in $(\mathrm{T_B} - \mathrm{T_A} + \delta t)$ time using a real road network. Where, $\mathrm{T_A}$ and $\mathrm{T_B}$ are the timestamp of publishing locations A and B, respectively. δt is a delay tolerant threshold which either could be set by the user or the system. For example, $\delta t = 1$ minute means there could be maximum 1 minute delay between query generation and query submission.

3. Privacy Settings: In our approach, we use minimum required area, $\mathcal{A}$, of a concealing region, CR, as the primary parameter to generate a CR. However, this approach requires detail information of map, incurring high space and computational complexity [GDSB16], and cannot always guarantee desired privacy [LSTL13]. Thus, besides area, we incorporate two other privacy requirements: k and l, to define minimum number of related locations and minimum diversity of the selected related locations in the CR, respectively. We also realize that in the case of continuous queries, it may not always possible to meet a hard privacy requirements. Regarding this, we define two privacy settings:

Expected privacy, $priv^E = \langle \mathcal{A}^E, k^E, l^E \rangle$ and

minimum privacy, $priv^{min} = \langle \mathcal{A}^{min}, k^{min}, l^{min} \rangle$. Where, $priv^E > priv^{min}$. The proposed approach first selects $priv^E$ to generate a CR. If it fails then it selects $priv^{min}$.

4. Secure Concealing Region CR: A CR is a convex hull of a set of selected related locations, W, such that it covers user's real location. Given privacy settings,

$priv^E$ and $priv^{min}$, a CR is said to be secured if

$$\begin{aligned} &\{Area(\mathrm{CR}) \geq \mathcal{A}^E \wedge |\mathrm{W}| \geq k^E \wedge |type(\mathrm{W})| \geq l^E\} \text{ or} \\ &\{Area(\mathrm{CR}) \geq \mathcal{A}^{min} \wedge |\mathrm{W}| \geq k^{min} \wedge |type(\mathrm{W})| \geq l^{min}\} \end{aligned} \tag{3.2}$$

5. Privacy Generating Region(PGR), Settings of Privacy Parameters, and Frequently Visited Places: To have a bound to generate a CR, a disk with radius $\mathcal{R}$, centered at user's location $\mathcal{O}$, is considered as Privacy Generating Region (PGR) such that, the generated CR falls entirely inside of PGR. Here, the value of $\mathcal{R}$ is defined by KLAP and can be changed by the user. To set the minimum required area $\mathcal{A}$ of CR, a user uses α as a factor of $\mathcal{R}$ such that, $\mathcal{A} = \pi(\frac{\mathcal{R}}{\alpha})^2 (1 \leq \alpha \leq \mathcal{R})$. We also assume that the users will identify their frequently visited places, e.g. home, work, favorite coffee shop, and so on.

6. Delay in CR Submission: Let us consider a user moved to a new location in $(\mathrm{T_B} - \mathrm{T_A})$ time and the minimum required time to submit a secure CR to the LBS is $\mathrm{T'_B}$. If $\mathrm{T'_B} > (\mathrm{T_B} - \mathrm{T_A}) + \delta t$, then we call it *delaying* the CR submission.

3.3.2 Privacy Evaluation

We measure the privacy as the uncertainty in identifying real location for an adversary [NLZ+14]. From a given CR with the list W of all the related locations, the privacy level, $\mathrm{E_{CR}}$, is computed as follows,

$$\mathrm{E_{CR}} = -\sum_i \overline{\mathcal{S}}_i \log_2 \overline{\mathcal{S}}_i; \text{ where, } \overline{\mathcal{S}}_i = \frac{\mathcal{S}_i^u}{\sum_i \mathcal{S}_i^u}; \forall i \in \mathrm{W} \tag{3.3}$$

3.3.3 Problem Formulation

Based on privacy level and delay, we formulate the problem of generating a secure concealing region CR as follows according to the fact that, on the one hand, we

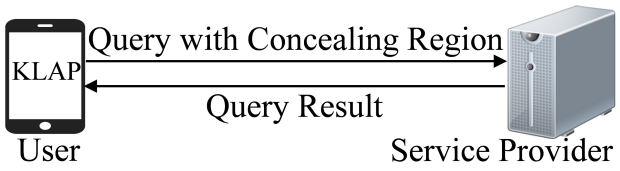


Figure 3.2: Proposed system model

want to generate a CR with maximum privacy level, on the other hand, we want to select the one with minimum delay to reach from previous CR.

$$\begin{gathered} \text{maximize}\{\text{privacy level } \mathrm{E_{CR}}, \frac{1}{e^{delay}}\} \\ s.t. \quad \mathrm{E_{CR}} \geq \lambda \mathrm{E_{Base}}; \{\lambda \in \mathbb{R}_{>0} | \lambda \leq 1\} \\ delay \leq \delta t; \{\delta t \in \mathbb{R}_{>0} | \delta t > 0\} \\ \{\alpha, k, l \in \mathbb{Z}^{+} | (1 \leq \alpha \leq \mathcal{R}) \wedge (k \geq 2) \wedge (1 \leq l < k)\} \end{gathered} \tag{3.4}$$

Here, $\mathrm{E_{Base}}$ is the privacy level of a baseline $\mathrm{CR_{Base}}$ and λ is a privacy level deviation control factor. Certainly, we want to make the value of λ as close as 1. From the experiment, it is observed that λ can be as large as 0.9. Note that, both λ and δt are system controlled parameters. On the other hand, $\alpha, k,$ and l are related to the user.

3.3.4 System Model

1. System Model

The general framework of a LBS system comprises a service provider and many mobile users. We assume that the users are registered to the system with a unique id, u. Our proposed framework, KLAP, resides in user's device (Figure 3.2). To find information about POIs of specific type ρ within a ROI with radius r, a user submits a query as follows, $Query_0 = \langle u, \langle (x, y), r \rangle, \rho \rangle$. Where, $\mathcal{O} = (x, y)$ denotes the user's original location in terms of latitude and longitude. Using the parameters

for $\mathcal{A}, k$, and l; the user generates a CR using KLAP algorithm. Then, the original query $Query_0$ is replaced by, $Query' = \langle u_{id}, \mathrm{CR}, r, \rho \rangle$. Instead of $Query_0$, $Query'$ is submitted to the LBS. Then, the LBS performs range query on the CR to find the POIs of type ρ, and sends them to the user.

2. Adversary Model

We consider an adversary knows information on users' historical queries. We also assume that the adversary knows how KLAP works but does not know the privacy parameters. Based on the available information, it can perform the following attacks to infer the user's location at a finer level.

a) Context linking attacks using knowledge of the map, popularity, and user's preference.

b) Region intersection attack on short-term multiple queries.

c) Real-Time Traffic Information (RTTI) based maximum movement boundary attack.

d) Long-term obfuscated location tracking attack.

3.4 Proposed Delay-Aware Privacy Preserving Approach

In this section, we describe the computational details of concealing region (CR) generation algorithm in KLAP. Based on the input, consists of privacy settings $priv^E$ and $priv^{min}$, previously published $\mathrm{CR_A}$ for a location $\mathcal{O}_\mathrm{A}$ at time $\mathrm{T_A}$, set $(\bigcup_\mathrm{x} \mathrm{CR_x})$ of all CRs for frequently visited places, user's current location $\mathcal{O}_\mathrm{B}$ at current time $\mathrm{T_B}$, the process of CR generation in KLAP comprises the following steps:

Algorithm 1: KLAP algorithm

Data: Previous $\mathrm{CR_A}$ at time $\mathrm{T_A}$, current time $\mathrm{T_B}$, user's current location $\mathcal{O}_\mathrm{B}$, $priv^E$, $priv^{min}$, $\mathcal{R}$, set $(\bigcup_x \mathrm{CR_x})$ of all CRs for frequently visited places, δt, λ, user's preference for all the locations $\mathcal{S}$, m

Result: Final concealing region CR

if $\mathcal{O}_B \in CR_A$ **then**
  Return $\mathrm{CR_A}$
if $\mathcal{O}_B \in CR_B$ //$CR_B \in (\bigcup_x CR_x)$ **then**
  $delay_i \leftarrow$delay between $\mathrm{CR_A}$ and $\mathrm{CR_B}$ using Equation 3.7
  if *($delay_i > \delta t$) $\wedge$ CR_A is not for frequent place* **then**
    $\mathrm{CR_A^-} \leftarrow$ Prune $\mathrm{CR_A}$ by excluding the related locations with $delay(a \in \mathrm{CR_A}, \mathrm{CR_B}) > \delta t$
    $delay_i \leftarrow$delay between $\mathrm{CR_A^-}$, and $\mathrm{CR_B}$ using Equation 3.7
  wait $delay_i$ amount of time and **Return** $\mathrm{CR_B}$
Select all the locations with $\mathcal{S}_i^u > 0$ from the region $(\mathrm{PGR} \setminus \{\mathrm{CR_A} \cup (\bigcup_x \mathrm{CR_x})\})$ in a list, *Loc*
if $(|Loc| \geq mk) \wedge (|type(Loc)| \geq ml)$ **then**
  $\mathcal{A}, k, l \leftarrow \mathcal{A}^E, k^E, l^E$
else
  $\mathcal{A}, k, l \leftarrow \mathcal{A}^{min}, k^{min}, l^{min}$
Compute the absolute difference of the locations with $\mathcal{O}_\mathrm{B}$ in terms of preference and sort them.
Seed $\leftarrow \{k$ random contiguous locations from sorted list which includes $\mathcal{O}_\mathrm{B}\}$
$\langle \mathrm{CR_{Base}}, \mathrm{E_{Base}} \rangle \leftarrow$ Generate-CR($\mathcal{O}_\mathrm{B}$)
List-of-CR $\leftarrow \emptyset$
for $(i = 1; i < k; i{+}{+})$ **do**
  $\langle \mathrm{CR}_i, \mathrm{E_{CR_i}}, delay_i \rangle \leftarrow$ Generate-CR($seeds(i)$)
  if $E_{CR_i} \geq \lambda E_{Base}$ **then**
    Put CR_i into *List-of-CR*
$\mathrm{CR_f}, delay_f \leftarrow$Select the CR with minimum *delay*
wait $delay_f$ amount of time and **Return** $\mathrm{CR_f}$

Algorithm 2: Generate-CR

Data: $seed, r, \mathcal{A}, k, l, m, \mathcal{S}, \mathcal{O}_{\mathrm{B}}, \mathrm{CR}_{\mathrm{A}}$

Result: CR, $\mathrm{E}_{\mathrm{CR}}, delay$.

Select all the locations with $\mathcal{S}_i^u > 0$ from the region $(\mathrm{PGR}_{seed} \setminus \{\mathrm{CR}_{\mathrm{A}} \cup (\bigcup_{\mathrm{x}} \mathrm{CR}_{\mathrm{x}})\})$ in a list, *Loc*

Compute the absolute difference of the locations with *seed* in terms of preference and sort them.

$LCL \leftarrow$ {first l-type locations from the sorted list} $\cup$ {first mk locations from the sorted list, including $\mathcal{O}_{\mathrm{B}}$}.

$SLCL \leftarrow$ Sorted LCL based on the locations' geographical distance with $\mathcal{O}_{\mathrm{B}}$, in ascending order.

$\mathrm{W} \leftarrow$ first k-locations from $SLCL$ including $\mathcal{O}_{\mathrm{B}}$; then delete them from $SLCL$.

while $|type(W)| < l$ *or* $Area(Convex\ Hull\ of\ W) < \mathcal{A}$ **do**

  $\mathrm{W} \leftarrow \mathrm{W} \cup SLCL(1)$

  delete $SLCL(1)$ from $SLCL$

$\mathrm{CR} \leftarrow$ Convex Hull of W

$delay \leftarrow$delay between CR_{A}, and CR using Equation 3.7

if $(delay > \delta t)$ **then**

  $\mathrm{CR}^{-} \leftarrow$Prune CR by excluding all the related locations with $delay(\mathrm{CR}_{\mathrm{A}}, b \in \mathrm{CR}) > \delta t$

  $delay \leftarrow$delay between CR_{A}, and CR^{-} using Equation 3.7

  if $(delay > \delta t)$ **then**

    $\mathrm{CR}_{\mathrm{A}}^{-} \leftarrow$ Prune CR_{A}

    $delay_i \leftarrow$delay between $\mathrm{CR}_{\mathrm{A}}^{-}$, and CR^{-} using Equation 3.7

  $\mathrm{CR} \leftarrow \mathrm{CR}^{-}$

$\mathrm{E}_{\mathrm{CR}} \leftarrow$ compute privacy level of W according to Equation 3.3

return $\langle \mathrm{CR}, E_{\mathrm{CR}}, delay \rangle$

In CR Test: If user's new location $\mathcal{O}_{\mathrm{B}}$ remains within CR_{A} then use it as the new CR (lines 1-2 of Algorithm 1).

Frequently Visited Place Check: If $\mathcal{O}_{\mathrm{B}}$ is in CR_{B}, a previously published CR for a frequently visited place, then use it as the final CR and skip steps 3, 4, 5, and pruning of it in step 7. Moreover, if $\mathcal{O}_{\mathrm{A}}$ is a frequently visited place, then also skip pruning CR_{A} in step 7 (lines 3-9 of Algorithm 1).

Seed Selection: First select all the locations with $\mathcal{S}_i^u > 0$ from the region $(\mathrm{PGR} \setminus \{\mathrm{CR}_{\mathrm{A}} \cup (\bigcup_{\mathrm{x}} \mathrm{CR}_{\mathrm{x}})\})$. Recall that, PGR is actually a circle with radius $\mathcal{R}$, centered at $\mathcal{O}_{\mathrm{B}}$. Then, for each location, compute the absolute difference with $\mathcal{O}_{\mathrm{B}}$, in terms of preference; and sort them based on that difference. From the sorted locations, select a list of locations as *seeds* as follows: $seeds = \{k$ random contiguous locations from the sorted list which includes $\mathcal{O}_{\mathrm{B}}\}$ (lines 9-15 of Algorithm 1).

One can argue that, we can consider $\mathcal{O}_{\mathrm{B}}$ as the only seed to generate the CR. However, it can leak user's privacy in two ways. First, locations are static and probabilities and preference do not change significantly over a long period of time. Thus, an adversary can apply reverse engineering process on a CR to map for which original location the vertices of that CR were generated and if there is only one such location is found, it is the user's original location. Second, as the CR was generated based on a fixed PGR, the adversary can perform geometric operations to shrink the CR.

Candidate Location Selection: For each seed, $seed_i$, consider a PGR_i, and select all the locations from the region $(\mathrm{PGR}_i \setminus \{\mathrm{CR}_{\mathrm{A}} \cup (\bigcup_x \mathrm{CR}_{\mathrm{x}})\})$. Again, compute their difference in preference with $\mathcal{O}_{\mathrm{B}}$; and sort them. From the sorted locations, select a list of candidate locations with {first mk locations} $\cup$ { first ml type locations}. Where, $m \geq 2$. Afterwards, calculate the physical distance of candidate locations from $\mathcal{O}_{\mathrm{B}}$, sort them, and store in a list LS (lines 1-4 of Algorithm 2).

CR Generation: From LS, select first k locations in a list W. Check if,$\{(|type(\mathrm{W})| \geq l) \wedge (Area(\text{convex hull of W}) \geq \mathcal{A})\}$ (lines 5-9 of Algorithm 2). If not, select more locations from LS in similar way in W. For consecutive queries, it may not always possible to generate a CR with $priv^E$. Thus, first check if it is possible to generate a CR with $prev^E$; if not, select $priv^{min}$ (lines 10-13 in Algorithm 1).

Time Reachability based Delay Calculation: For the generated CR_i, compute the (possible) delay from Real-Time Traffic Information(RTTI) based time to reach from CR_A to CR_i as follows:

$$\mathrm{RTTI}(a \in \mathrm{CR}_\mathrm{A}, b \in \mathrm{CR}_i) = \text{minimum required time to move from } a \text{ to } b \quad (3.5)$$

$$delay(a \in \mathrm{CR}_\mathrm{A}, \mathrm{CR}_i) = \begin{cases} 0, \text{if}(\mathrm{T_B} - \mathrm{T_A}) \geq \min(\mathrm{RTTI}(a, \forall b \in \mathrm{CR}_i)) \\ (\mathrm{T_B} - \mathrm{T_A}) - \min(\mathrm{RTTI}(a, \forall b \in \mathrm{CR}_i)), \text{otherwise} \end{cases} \quad (3.6)$$

$$X(m,n) = \max \left\{ \begin{array}{l} \min(delay(\forall a \in \mathrm{CR}_\mathrm{A}, \mathrm{CR}_i)), \\ \min(delay(\mathrm{CR}_\mathrm{A}, \forall b \in \mathrm{CR}_i)) \end{array} \right\} \quad (3.7)$$

Delay Management using CR Pruning: If $delay > \delta t$, one straightforward solution could be postponing the CR submission for at least $delay$ amount of time or generate a CR_i closer to CR_A without containing $\mathcal{O}_\mathrm{A}$[GDSB16]. However, this approach incurs reduction in quality-of-service. Thus, to deal with delay, we propose a CR pruning approach. To prune a CR, first identify the related locations causing the delay and then exclude them from it's list of related locations. If the new CR, reduces the delay then use that as new CR. Figure 3.3 depicts an example for CR pruning where W_A and W_i are the lists of related locations of CR_A and CR_i, respectively. In detail,

W_A	W_i				min(W_A)
	0	1	2	3	
0	2	0.5	2.6	1	0.5
1	0	0.2	5	0	0
2	2.5	2.9	4	2.3	2.3
min(W_i)	0	0.5	2.6	0	

Figure 3.3: CR pruning example with $\delta t = 1\ minute$: Each entry refers to the *delay*(in min.) to reach from $\mathrm{W_A}(m)$ to $\mathrm{W}_i(n)$. To reach from $\mathrm{W}_A(2)$ to any location in W_i minimum $(\mathrm{T_B} - \mathrm{T_A}) + 2.3$ minutes are required. On the other side, to reach $W_i(2)$ from any location in $\mathrm{W_A}$ minimum $(\mathrm{T_B} - \mathrm{T_A}) + 2.6$ minutes are required. Thus, we exclude both locations from W_i and $\mathrm{W_A}$, respectively, to get pruned CRs.

a) First prune CR_i and compute the delay. If it reduces the delay, then use the pruned version as CR.

b) If the pruned CR_i's delay regarding $\mathrm{CR_A}$ is still greater than δt, then, also prune $\mathrm{CR_A}$. Afterwards, for each seed, store the following information: $\{\mathrm{CR}_i^-, \mathrm{E}_{\mathrm{CR}_i^-}, delay\}$, where CR_i^- is the pruned version of CR_i.

Note that, one can argue to compute the delay to reach all the locations in $(\mathrm{PGR} \setminus \{\mathrm{CR_A} \cup (\bigcup_x \mathrm{CR_x})\})$ from $\mathrm{CR_A}$ in advance, and select locations based on minimization of delay. However, if the number of locations inside of $(\mathrm{PGR} \setminus \{\mathrm{CR_A} \cup (\bigcup_x \mathrm{CR_x})\})$ is large, it is not feasible to send large volume of requests in a short-period of time to a map service in a user-centric approach.

Final CR selection: To solve the Multi-Objective Optimization problem of selecting the final CR(equation 6.4), first select all the CRs based on the following relation, $\mathrm{E}_{\mathrm{CR}_i} \geq \lambda \mathrm{E}_{\mathcal{O}_\mathrm{B}}$, where $\mathrm{E}_{\mathcal{O}_\mathrm{B}}$ is the privacy level achieved from the CR with $\mathcal{O}_\mathrm{B}$ as the seed. Then return the CR with minimum delay.

3.5 Scheme Analysis

In this section we analyze the security aspects of our proposed mechanism KLAP. In particular, we discuss the impact of real-time traffic information based maximum movement boundary, long-term obfuscated location tracking, and probability distribution and personal context linking attacks.

Definition 3.5.1 *Given two CRs, CR_A and CR_B, submitted at T_A and T_B timestamps, with W_A and W_B as their corresponding set of related locations; a RTTI based maximum movement boundary attack on continuous queries is successful if*

$$W_A^* \cup W_B^* \neq \emptyset \tag{3.8}$$

$$W_A^* = \{\exists i \in W_A | RTTI(i, \forall j \in W_B) > (T_B - T_A)\}$$
$$W_B^* = \{\exists j \in W_B | RTTI(\forall i \in W_A, j) > (T_B - T_A)\}$$

Theorem 3.5.2 *RTTI based maximum movement boundary attack does not reveal user information in KLAP.*

Proof. In KLAP, using pruning and delay, $\mathrm{CR_B}$ is submitted only when all of its locations are reachable from $\mathrm{CR_A}$. Hence, $\mathrm{W_B^*} = \emptyset$. On the contrary, if $\mathrm{CR_A}$ is not for a frequently visited place and $\mathrm{W_A^*} \neq \emptyset$, then these locations must be excluded in the CR pruning step. That is, $\mathrm{W_A^*}$ does not leak any significant location information for $\mathrm{CR_A}$ and $\mathrm{CR_B}$, and therefore, $\mathrm{W_A^*} = \emptyset$. However, if $\mathrm{CR_A}$ is for a frequently visited place then certainly $\mathrm{W_A^*} = \emptyset$ through delay. Thus, RTTI based attacks on consecutive CRs does not reveal any location information in KLAP. □

Definition 3.5.3 *Given multiple CRs, $\langle CR_0, \ldots CR_n \rangle$, submitted from a frequently visited location $\mathcal{O}$ on different days, a long-term obfuscated location tracking attack*

is defined as,

$$CR_x = CR_0 \cap \ldots \cap CR_n \tag{3.9}$$

such that, $CR_x = \{\exists CR \in \{CR_0, \ldots CR_n\} | (CR \neq CR_x) \wedge (\mathcal{O} \in CR_x)\}$.

Theorem 3.5.4 *KLAP is resilient against long-term obfuscated location tracking attack.*

Proof. KLAP learns the frequently visited locations from user's input and once a CR is generated for such a location, it stores that CR for future queries and skip the pruning step for it. Thus, $\mathrm{CR_x} = \mathrm{CR_0} = \cdots = \mathrm{CR_n}$ □

Definition 3.5.5 *For a submitted CR with a list of related locations W, an approach is resilient against probability distribution and personal context linking attacks if*

$$Pr(w_i \in W | \mathcal{O} \in W) = Pr(w_j \in W | \mathcal{O} \in W) = \frac{1}{|W|}; \forall i \neq j \tag{3.10}$$

Theorem 3.5.6 *An attacker cannot infer user's location using probability distribution and personal context linking attacks.*

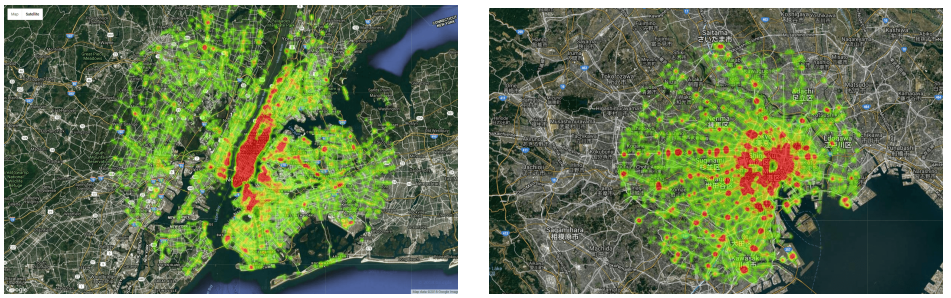
Proof. KLAP first defines user's preference to a location based on popularity and user's bias to that location. Then, its seed and candidate locations selection algorithm steps ensure that only the closely related locations, in terms of preference, defines the CR. Thus, every location in W has the equal probability($\frac{1}{|\mathrm{W}|}$) to be $\mathcal{O}$. □

3.6 Experimentation and Analysis

We evaluate the proposed KLAP framework with two real-world datasets provided by Foursquare for NYC and Tokyo cities[YZZY15]. The important statistics of the datasets are provided in table 3.2.

Table 3.2: Dataset Statistics

Dataset	#Check-ins	#Locations	#Types	#Users
NYC	227428	38333	400	1083
Tokyo	573703	61858	385	2293



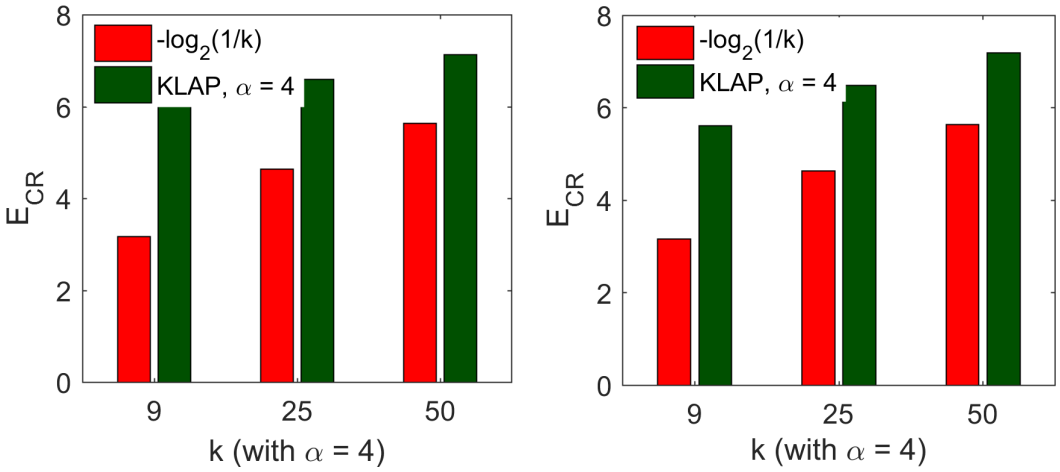
(a) (b)

Figure 3.4: Locations in (a) NYC and (b) Tokyo datasets.

For each check-in we consider five information types: user id, location id, location using GPS coordinates, location type, and time of check-ins. Figure 3.4 presents the locations used in the two datasets. The parameters and their different values used in the experiment are as follows: $r = 1000$ meter, $k = \{9, 17, 25, 50\}, \alpha = \{4, 10, 12\}, l = \{3, 5, 7\}, \mathcal{R} = 1000$ meter, $\delta t =$ 1 minute, $\delta = 0.9, m = 2$

In this chapter, we use the following baseline approaches for comparison:

1. Rand-CR: It Generates a random CR based on area $\mathcal{A}$ requirement.
2. k-DLCA: It covers user's ROI with n-number of circles with equal radii such that the intersection region of those circles construct a CR[LHA+16].
3. Cont-Dummy: Based on spatiotemporal correlation, this approach generates dummies considering real-time traffic information[LLL+17].
4. NoCor-CR: This is a variation of our proposed obfuscation approach KLAP without considering spatiotemporal correlation between the CRs.



(a) NYC (b) Tokyo

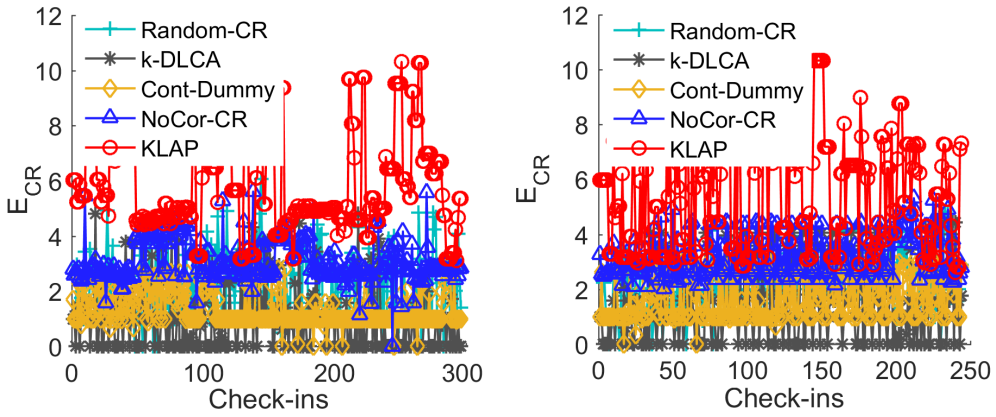
Figure 3.5: Relationship between privacy area requirement $\mathcal{A}$ and k; and their impact on privacy level $\mathrm{E_{CR}}$.

5. NoPrune-CR: This is another variation of KLAP without the pruning step. That is, when a CR cannot be submitted in current time, it delays the process. This approach generates a CR with privacy setting $priv^E$. This approach is similar to PROBE under spatiotemporal constraints[GDSB16].

With different combinations of the parameters, we ran these approaches on 40 trajectories of different users from both datasets, each having $\approx$ 15 check-ins on average, to generate different statistics. Without loss of generality, we use home address of each user as the frequently visited place.

3.6.1 Privacy Level

We first evaluate the privacy level, $\mathrm{E_{CR}}$, achievable from a given CR; and the CR depends on the three privacy parameters k, l, and $\mathcal{A}$ (by means of α). Obviously, with the increase of k, $\mathrm{E_{CR}}$ increases. While the parameter l ensures the required level of diversity, we observe its little impact on the overall privacy level. Thus, we are more interested in checking how the area requirement affects privacy in KLAP.



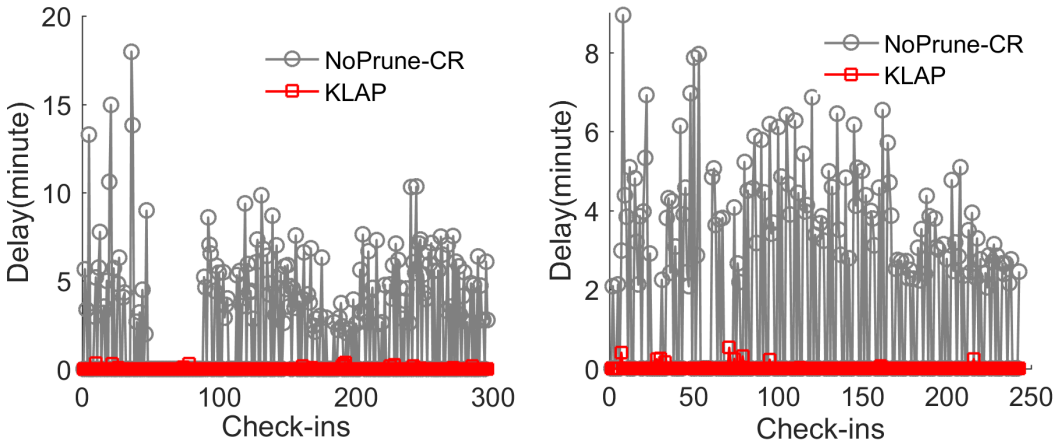
(a) NYC (b) Tokyo

Figure 3.6: Comparison in terms of privacy level, E_{CR} among different approaches. KLAP settings: $k = 9, l = 5, \alpha = 10$

In figure 3.5 we depicts the relationship between $\mathcal{A}$(in terms of α), k, and E_{CR}. Evidently, with the privacy area requirement, KLAP can achieve higher privacy than ideal privacy level with just k locations ($-\log_2(\frac{1}{k})$). Thus, to achieve a high degree of privacy, a user can set a higher value for $\mathcal{A}$, leaving the values of k and l with some default small numbers.

Comparison

The comparison among the mentioned approaches in terms of privacy level, E_{CR}, is shown in figure 3.6. k-DLCA shows the lowest value for E_{CR} because it neither considers user's preference nor any privacy requirement and also, the generated CR cannot guarantee a certain area of it. With Random-CR, it is possible to achieve better privacy than k-DLCA and Cont-Dummy because it generates the CR ensuring privacy area $\mathcal{A}$. However, it does not consider the relationship between user and locations. Although NoCor-CR improves this limitation, it cannot guarantee privacy in continuous check-in cases. With the consideration of Spatio-temporal correlation between successive check-ins, CR pruning approach, and (expected, minimum) privacy requirements, KLAP achieves the highest degree of privacy. More precisely, we



(a) NYC (b) Tokyo

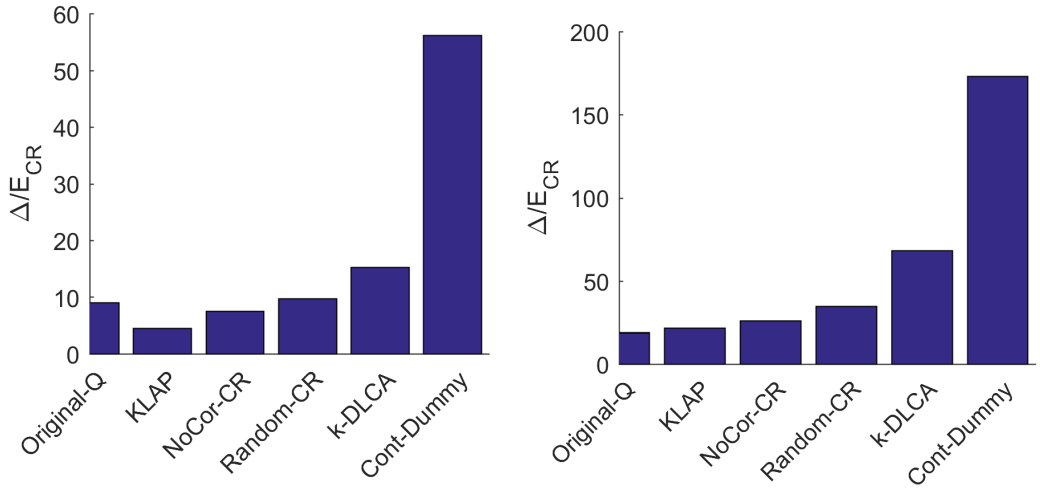
Figure 3.7: Impact of CR Pruning on delay management for (a) NYC and (b) Tokyo.

found only 4% and 7% check-ins in NYC and Tokyo, respectively, for which privacy level of KLAP is less than NoCor-CR. This can also be improved by defining multiple privacy settings between $priv^E$ and $priv^{min}$, and select the one closest to $priv^E$, with an increase in computation cost. The oscillating nature of KLAP's result is effected either by regions' location density or the CR pruning step. At regions, where the density of locations is high, a generated CR may cover a large number of related locations; yielding a high degree of privacy, even if k is small.

3.6.2 Impact of Pruning

Delay Management

We compare the amount of delay imposed by NoPrune-CR and KLAP in figure 3.7 to show the improvement achieved by KLAP with the pruning technique. In the experiment we found that the delay in KLAP is always less than 1 minute. This large improvement in KLAP is achieved with the concept of pruning and minimum privacy, $priv^{min}$ settings.



(a)NYC (b) Tokyo

Figure 3.8: ($\Delta/\mathrm{E_{CR}}$) based comparison among different methods for (a) NYC & (b) Tokyo. Original-Q refers to the original query without any privacy-preservation. For comparison purpose we use $E_{\mathrm{CR}} = 1$ for Original-Q.

Impact of Pruning on Privacy Level

In case of pruning previously published $\mathrm{CR_A}$, we observe a small reduction in privacy level for $\mathrm{CR_A}$ which is always $\leq 10\%$ in both datasets.

3.6.3 Communication Cost

The communication cost, Δ, is computed using both of upstream, and downstream cost[LSTL13] as $\Delta = |C| + N$. Where, $|C|$ is the total number of vertices of the CR and N is the number of POIs returned by the LBS. To measure Δ, we consider one type of location to query. In figure 3.8, we present $\Delta/\mathrm{E_{CR}}$ to show the cost per privacy level in different approaches. It is evident that to achieve 1 unit of privacy level in KLAP; the cost is the lowest. On the other hand, Cont-Dummy yield the highest amount of cost, indicating the advantage of obfuscation approaches over dummy based approaches in terms of both privacy and communication efficiency.

3.6.4 Storage Cost

In KLAP, storage cost is mainly influenced by 1) number of locations in the dataset, 2) CRs of frequently visited locations, and 3) previously published CR. Recall that, our approach is a user-centric one. On each user's device, for each location we need to store $\{id, latitude, longitude, type, preference\}$. If we use flat-file to store this information we would need no more than 200byte for each location. Thus, it is required 8MB and 13MB for NYC and Tokyo, respectively, to store all the locations' information. Each CR comprises two information: its vertex list and the ids of related locations inside of it. Even if a CR contains all the locations in city, then it would take 1MB space for a simple file to store their ids. Therefore, if a city has 1 million locations, then KLAP requires less than 250MB space to store necessary information.

3.7 Discussion and Summary

In this chapter, we introduced an obfuscation mechanism for LBS, called KLAP. Based on a user's privacy settings, KLAP defines a concealing region (CR) to obfuscate the user's real location. Here, we made three significant contributions: first, through the usages of location's popularity and user's preference, KLAP can achieve strong privacy against personal context linking and probability distribution based attacks; second, the utilization of real-time traffic information allows KLAP to be practical in real-world scenarios and protect against maximum movement boundary-based attack; and finally, with the proposed CR pruning technique, KLAP reduces delay to a great extent compared to delay-based mechanism. Evaluation results of privacy, communication, and storage costs, based on two real datasets, mark the advantage of the proposed mechanism over existing approaches.

CHAPTER 4

LOCATION INFERENCE ATTACKS ON GEO-TAGGED AND NON GEO-TAGGED DATA AND THEIR COUNTERMEASURES

In this chapter, we study the privacy issues associated with the trajectories containing both geo-tagged and non-geo-tagged data. Without loss of generality, we limit our study to checkins and geo-tagged and non-geo-tagged photos. We propose a probabilistic inference model by considering both checkin and photo probabilities for each location. With Foursquare's New York City and Tokyo datasets, we first implement the inference model on three variations of dummy-based obfuscation mechanisms, and show that a straightforward application of existing dummy approaches can leak location privacy for trajectories containing both geo-tagged and non geo-tagged data. To the best of our knowledge, this is the first work to investigate the impact of historical shared photos on location privacy. After observing the negative impact of the inference attack, we also propose an improved version of [NLZ+14] to negate such an inference. Our contribution also includes a visualization technique to visual represent trajectory privacy-preserving checkins and photos in location-based social networks (LBSN), another form of LBS.

The chapter is organized as follows. Sections 4.1 and 4.2 provide a formal introduction to the problems. The necessary background information, fundamental concepts, the proposed system and attack models are discussed in section 4.3. Finally, the proposed privacy-preserving mechanism is detailed in section 4.4. The chapter is concluded in section 4.6.

4.1 Introduction

The advancement of location technology and smart devices is fading the gap between the physical world and online social network; making Location-Based Social

Network (LBSN) a popular platform to enjoy different location-dependent services, e.g., friend finder, point-of-interest(POI) search, check-in, and geo-tagged photo sharing. There exist many LBSNs offering different services, such as Facebook, Foursquare, and Google place. In Foursquare, more than 160 million check-ins and 5 million photos were recorded for New York City alone [fou]. While it prompts technological and societal advantages, the vast collection of location information of both check-ins and geo-tagged photos poses serious privacy concerns. Studies show that it is possible to reveal a user's home, workplace, lifestyle, health condition, and political views. from the location information [DMHVB13, LARC16, TPI17]. For instance, Liccardi et al. [LARC16] proposed a visual technique based location inference model, using twitter check-in data, to show that people's most frequent and private locations, such as work and home, can be deduced using only a small sample of location points (1-day worth). To counter such an inference attack, several Trajectory Privacy-Preserving Mechanisms (TPPMs) have been proposed, including dummies-based obfuscation[LLL$^+$17].

We identify that existing inference models, as well as TPPMs, mostly consider only the location information. These TPPMs can guarantee strong privacy, only if we limit inference to check-ins. However, besides check-in, photo sharing is another prevalent form of interaction in LBSN and the impact of historical shared photos on location privacy, specifically their distributions over locations, is not studied yet. In this paper, we aim to show that, it is possible to design inference model to deduce location information in the spatiotemporal domain based on the distribution of historical check-ins and photos. The novelty of our work is that we designed an inference model which is oblivious to contents of the photo. We experimentally show that, under specific spatiotemporal constraint, such an inference model can be a useful tool to infer a user's location at a higher resolution, even if its lo-

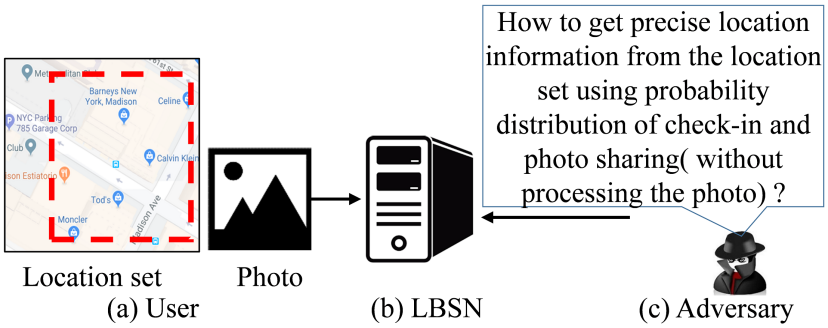


Figure 4.1: Location inference motivation:(a) a user generates a set of locations using an existing location-privacy preserving mechanism (TPPM) on a mobile device (e.g., smartphone, smartwatch, and so on)to conceal her original location in a geo-tagged photo and (b) submits them to an LBSN. (c) With access to the LBSN, an adversary's goal is to find a user's original location from the location set without processing the content of the photo.

cation information is "well" protected using an obfuscation approach. While our analysis is concentrated on dummy-based obfuscation, it can be extended for differential privacy-based obfuscation, such as geo-indistinguishability[ABCP13], which we intend to explore in our future work. The generic framework of the inference is depicted in Figure 4.1. With our experimental results signifies our argument that, for LBSs, where users interaction covers heterogeneous contents, TPPM should be designed by considering the impact of all the type of data, not only the geo-tagged content.

4.2 Motivation and Problem Statement

For a better understanding of how spatiotemporal analysis of the distribution of historical photos influences location privacy, we illustrate a couple of examples in Figure 4.2. We first discuss a more straightforward form of inference, so-called "0/1-probability-based inference", where an adversary checks whether a location has any probability from where a photo can be shared. Let's consider a user decided to

share a check-in with a photo in LBSN. To protect privacy of current location $\mathcal{O}$, she shared $\mathcal{O}$ with $k-1$ dummy locations using any of the dummy based methods in [NLZ+14, NLZ+15, LLL+17, YLX+17](Figure 4.2(a)). An attacker knows that the user is at any of the k locations, and a photo is also shared from that location. Without considering the content of the photo, the attacker can look into the historical data and can eliminate the dummies with zero (0) photo sharing probability to infer more precise user's location information. The definitions of check-in and photo sharing probabilities are discussed in section 4.2. Using data, collected from Foursquare through their public API, we apply this inference model on three different algorithms, which generate dummies based on historical check-in distribution. The details of the experiment are discussed in section 4.5. Figure 4.2(b) presents results of the inference model for Random[NLZ+14] and Baseline [NLZ+15] Algorithms. It is evident that overlooking the impact of photos makes these algorithms highly vulnerable against the proposed inference model. Definitely, this inference model can be further improved with the consideration of spatiotemporal correlation between successive events (we use 'event' as the generic term for check-ins and photos). Let us consider two events, E_0 (a photo without any geo information) and E_1 (a check-in), generated at time T_0 and T_1, respectively. As both of the events are independent of each other, one straightforward solution to protect location privacy of E_1, is to generate a set of dummies using check-in probability (Figure 4.2(c)). However, $(T_1 - T_0)$ might be small, indicating that a user's position did not change significantly between E_0 and E_1. In other words, if $(T_1 - T_0) \leq \delta$, then an adversary can guess with high confidence that the user's location remained the same and thus can apply the discussed inference attack. Motivated by this example, we seek the answer to the following problem: *How can the incorporation of heterogeneous contents' information in the process of obfuscation allows a user to protect his/her*

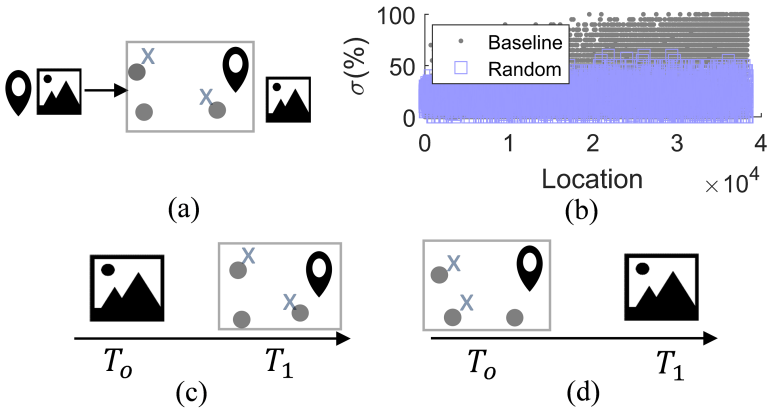


Figure 4.2: (a) 0/1-probability based inference: User hides his location of geo-tagged photo with 3 different dummy locations using a check-in probability-based(P_c) existing TPPM. Two of those locations have zero photo-sharing probability(P_ρ)(x marked), thus excluded by an adversary. (b) Example on real dataset: adversary's success rate $\sigma(\%)$ on Random[NLZ+14] and Baseline[NLZ+15] algorithms(using $k = 20$) with 0/1-probability based inference, performed using a real dataset. (c) and (d) Inference based on spatiotemporal correlation: Although dummy locations set and photo are submitted in different time, as $(T_1 - T_0)$ is small, an adversary can guess with high confidence that both events were generated from the same location. Thus, can exclude some dummies in a similar way.

trajectory privacy in the spatiotemporal domain?

We hypothesized that by considering the probabilities of content sharing at different locations based on their historical data and by utilizing delay and drop in the process of obfuscation allow a user to protect his/her trajectory privacy.

That is, we argue that it is also required to consider photo-sharing probability while generating the dummies (from our previous paragraph, it is E_1) to protect trajectory privacy. However, it is not always possible to find enough candidate locations nearby (due to spatiotemporal and regions' location density constraints) with photo-sharing probability. In that case, **delaying** E_1 can protect privacy, not posting dummies instantly. Let us consider another example from Figure 4.1(d), where E_0 is a check-in, E_1 is a photo, and E_0's privacy is protected using a set of dummies $\mathcal{S}_0$, generated based on check-in sharing probability. If $(T_1 - T_0) \leq \delta$, then

Table 4.1: Table of Symbols

Symbol	Description	Symbol	Description
$\mathcal{O}$	User's real location	k	#of locations to hide $\mathcal{O}$
L	Set of all locations	$\mathcal{P}_i$	checkin-photo sensitivity of a location
P_c	check-in probability	δ	permanence threshold
$\mathcal{E}$	Degree of privacy	P_r	Generic term for P_c and $\mathcal{P}$
$\mathcal{S}$	Set of k locations including $\mathcal{O}$	m	Candidate location selection parameter
σ	#of dummies with zero checkin-photo sensitivity	λ	Degree of privacy selection parameter

we argue that **dropping** the photo is the solution to protect privacy. However, if $(T_1 - T_0) > \delta$, then the temporal relationship between E_0 and E_1 does not reveal a user's location information, even if user's location remained the same. Thus, the user can safely **post** E_1. Now, let's consider another event E_2 (a check-in) such that $(T_2 - T_1) \leq \delta$. That is, the user's location did not change between T_0 and T_2. If we use $\mathcal{S}_0$ for E_2, then it can leak privacy for E_1 as $\mathcal{S}_0$ was generated based on check-in sharing probability. Thus, we propose to select the closest location with photo-sharing probability with a **spatial error** as a user's location and generate dummies based on that location.

4.3 Background

In this section, we describe first different fundamental concepts related to our work. We then present the system model and attack strategies of the adversary.

4.3.1 Fundamental Concepts

1. Event: A generic name for check-in, geo-tagged photo, and photo. An event generated at time T_t is denoted by E_t.

2. Check-in Probability, P_{c_i} If the number of historical check-ins, shared from location i, is C_i, then $P_{c_i} = \frac{C_i}{\sum_j C_j}; \quad \forall j \in L$, Where L is the set of all locations.

3. Photo Sharing Probability, P_{ρ_i}**:** If the number of historical shared photos from location i is ρ_i, probability to share a photo from i is, $P_{\rho_i} = \frac{\rho_i}{\sum_j \rho_j}; \quad \forall j \in L$

4. Checkin-photo Sensitivity, $\mathcal{P}_i$**:** A location, loc_i, is said to be checkin-photo sensitive, if $P_{c_i} > 0$ and $P_{\rho_i} > 0$. Then, checkin-photo sensitivity of location Loc_i is, $\mathcal{P}_i = P_{c_i} \times P_{\rho_i}$. Thus, a location is checkin-photo sensitive, if $\mathcal{P}_i > 0$.

5. Probability, P_r**:** A generic term for P_c and P_ρ. If check-in probability is used to generate dummies, then $P_r = P_c$. In case of checkin-photo sensitivity, $P_r = \mathcal{P}$.

6. Timing Error: If posting of an event is delayed for certain amount of time, then timing error occurs. It is define as, timing error $= \frac{n_{de}}{n}$.

7. Dropping Error: If an event (specifically, photo) is dropped without posting, then dropping error occurs; and it is defined as, dropping error $= \frac{n_{dr}}{n}$.

8. Spatial Error: If instead of user's original location, another location is considered as user's 'real' location then we call such phenomena as spatial error, which is defined as, spatial error $= \frac{n_{sp}}{n}$.

Here, n, n_{de}, n_{dr}, and n_{sp} refer to total number of events, delayed events, dropped events, and spatial errors; respectively.

4.3.2 System Model

Our proposed photo-check mechanism works on user's device without relaying on a third party. Figure 4.3 depicts the proposed system model. Based on the spa-

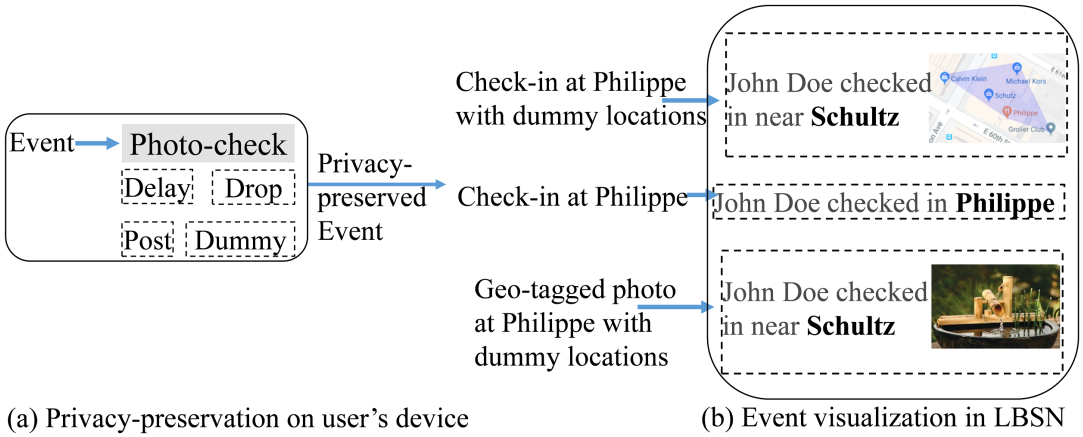


Figure 4.3: Proposed system model: (a) Photo-check's system model on user's device with 4 possible actions to preserve location privacy of an event, (b) visualization of example privacy-preserved events in LBSN.

tiotemporal correlation between new and previously generated events,photo-check selects the appropriate action from an array of 4 possible actions to preserve location privacy. We assume that the check-in and photo-sharing probabilities are publicly available. If an event contains dummies, user's location is computed from the convex hull of those dummies (Figure 4.3(b)). The detail of the proposed mechanism is discussed in section 4.4.

4.3.3 Adversary Model

We consider an adversary with access to the historical check-in and photo distributions. We assume that the adversary does not use any image content analysis method to localize a user from a given photo. Instead, it considers only the probability distribution of historical check-in and shared photos to infer the user's location. We also assume that it has users' approximate permanence information at different locations. For example, Google search shows that people typically spend 15 min at McDonald's of 4217 Genesee St, Cheektowaga, NY 14225. Without loss of general-

ity, we take the time difference between two consecutive events as the threshold (δ) to determine whether a user remains at the same location.

4.4 Proposed Privacy Preserving Geo-Tagged and Non Geo-Tagged Data Sharing Approach

Our proposed photo-check mechanism comprises two modules: privacy preservation on an event before sending it to a LBSN and visualization of the privacy-preserved event in LBSN.

Privacy Preservation: The main goal of photo-check is to maximize user's privacy against an adversary while posting an event. To achieve this goal, photo-check uses the the following 4 actions: **post, delay, drop,** and **dummy generation**. The flowchart of the mechanism is presented in Figure 4.4. Using information of previously posted event E_{prv}, user's current event E_{cur}, user's current location $\mathcal{O}$, and user's privacy setting (number of required dummies, k) as input, photo-check works in the following main steps:

1) Check whether E_{cur} contains location information(either a check-in or a geo-tagged photo). If so, then go to step 2; otherwise go to step 5.

2) If $\mathcal{O}$ is not checkin-photo sensitive, then post E_{cur} with $k-1$ dummies using P_c; otherwise, go to step 3.

3) Check whether enough candidates ($\geq mk$) within the proximity of $\mathcal{O}$ such that $dist(\mathcal{O}, x) \leq \mathcal{R}$, where x is a candidate location and $\mathcal{R}$ is a distance control parameter (e.g. 1000 meter). If yes, then generate $(k-1)$ dummies using $\mathcal{P}_{\mathcal{O}}$ and post E_{cur}. If not, go to step 4.

4) If the following condition (Eq.4.1) satisfies, delay E_{cur}; otherwise, generate dum-

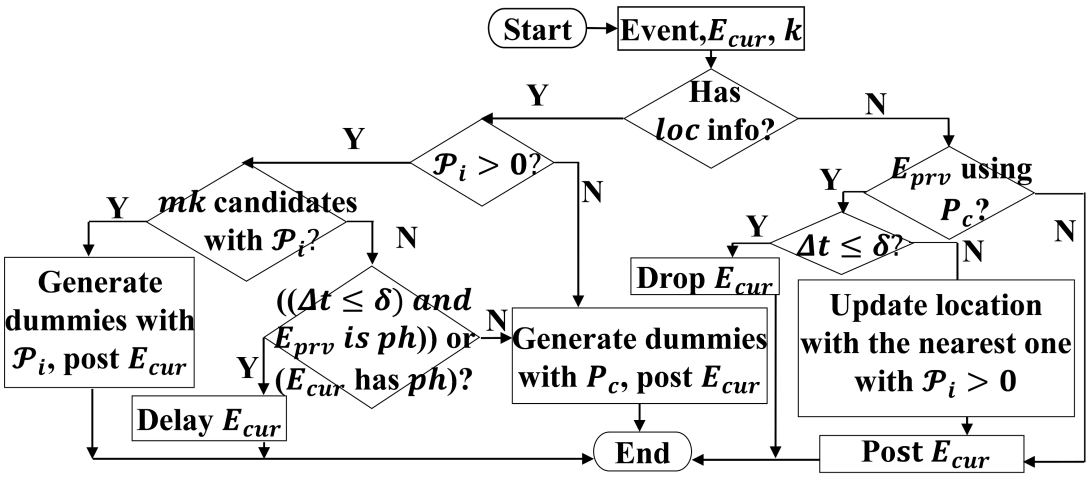


Figure 4.4: Photo-check flowchart. Here loc, ph, and Δt refer to location, photo, and $(T_{prev} - T_{cur})$ respectively.

mies using P_c and post E_{cur} with the dummies.

$$(((t_{cur} - t_{prv}) \leq \delta) \wedge (E_{prv} \text{ is } photo)) \vee (photo \in E_{cur}) \tag{4.1}$$

The proposed dummy generation algorithm is discussed in the later part of this paper.

5) If E_{prv} was a photo or dummies were not generated for it using P_c, then directly post E_{cur}. Otherwise check check whether $(t_{cur} - t_{prv}) \leq \delta$. If yes, then drop E_{cur}. Else, post E_{prv} and update current location with the closest location i with $\mathcal{P}_i > 0$ such that $dist(\mathcal{O}, i) \leq \mathcal{R}$.

Visualization in LBSN: After receiving the privacy-preserved E_{cur}, to visualize it, the LBSN works as follows. First, check whether E_{cur} contains a set of dummies. If so, then compute their convex hull and it's centroid from the location set. The closest location in the dummy set from the centroid is treated as the approximate location of the user (Figure 4.3(b)). Otherwise, post E_{cur} as it is.

4.4.1 Dummy Generation

In dummy generation, our main objective is to select a set $\mathcal{S}$ of $k-1$ dummies and $\mathcal{O}$, which ensures highest degree of privacy, $\mathcal{E}$. This goal can be achieved if we select locations as dummies having least difference with $\mathcal{O}$ in terms of P_r, which will maximize $\mathcal{E}$. $\mathcal{E}$ of a set $\mathcal{S}$ is computed as follows,

$$\mathcal{E}_s = -\sum_{i=0}^{k-1} P'_{ri} \log_2 P'_{ri} \tag{4.2}$$

where, $P'_{ri} = \frac{P_{ri}}{\sum_{i=0}^{k-1} P_{ri}}$ is the normalized probability. Certainly, $\mathcal{E}_s$ will be maximum, if all the k locations' probability are same, i.e. $\frac{1}{k}$. That is, optimum privacy, $\mathcal{E}_{opt} = -\log_2(\frac{1}{k})$. $\mathcal{E}_s$ can be maximized to $\mathcal{E}_{opt}$ if we consider the whole map as the search space for dummy generation. In that case, the dummies distance could be far from each other. However, to visualize the user's approximate location in LBSN post, it is indeed not a feasible solution. Moreover, the spatiotemporal analysis on consecutive events will also reveal that dummies cannot be far from each other. Thus, we constrain the maximum allowable distance between a dummy and $\mathcal{O}$ with a distance control parameter $\mathcal{R}$. On the flip side, this constraint imposes another problem: we may not find enough candidate locations as dummies which are closely related to $\mathcal{O}$, in terms of P_r. That is, it might not always be possible to achieve optimal privacy. Therefore, our goal is to select a set of dummies as the final set $\mathcal{S}_f$ whose $\mathcal{E}_f$ is close to $\mathcal{E}_{opt}$. Formally,

$$\begin{gathered} \text{maximize}\{\text{degree of privacy, } \mathcal{E}_s\} \\ s.t. \quad \mathcal{E}_s \geq \lambda \times \mathcal{E}_{opt} \end{gathered} \tag{4.3}$$

Indeed, we want to have λ as close as 1 and in our experiment we found its value can be as large as 0.9. Now, we discuss our proposed randomized greedy dummy generation algorithm. In the first step, it selects all the locations within $\frac{\mathcal{R}}{2}$ distance

Algorithm 3: Dummy Generation Algorithm

Data: user's real location $\mathcal{O}$, set of all locations L, P_r of all locations in L, k

Result: Final location set $\mathcal{S}_f$ and degree of privacy, $\mathcal{E}_f$

$L_{selected} \longleftarrow$ Select each location l from L such that, $(P_{rl} > 0) \wedge (dist(\mathcal{O}, l) \leq \frac{\mathcal{R}}{2})$

$L_{sorted} \longleftarrow$Compute $|P_{rl} - P_{ro}|, \forall l \in L_{selected}$; and sort them in ascending order

$\mathcal{C} \longleftarrow$ Select mk random contiguous locations including as candidates from L_{sorted}

$i \longleftarrow 0$

$\mathcal{E}_s \longleftarrow 0$

$\mathcal{E}_{opt} \longleftarrow -\log_2(\frac{1}{k})$

while $((\mathcal{E}_s < \lambda \times \mathcal{E}_{opt}) \ \wedge \ (i < |\mathcal{C}|))$ **do**

  $c \longleftarrow \mathcal{C}_i$

  $L_{selected} \longleftarrow$ Select each location l from L such that, $(P_{rl} > 0) \wedge (dist(c, l) \leq \frac{\mathcal{R}}{2})$

  $L_{sorted} \longleftarrow$Compute $|P_{rl} - P_{rc}|, \forall l \in L_{selected}$; and sort them in ascending order

  $\mathcal{S} \longleftarrow \mathcal{O} \cup \{$ First $(k-1)$ locations from L_{sorted}, excluding $\mathcal{O}\}$

  $\mathcal{E}_s \longleftarrow$ Compute degree of privacy of $\mathcal{S}$ using Equation 4.2

  $i \longleftarrow i + 1$

$\mathcal{S}_f \longleftarrow S, \mathcal{E}_f \longleftarrow \mathcal{E}_s$

with $P_r > 0$, computes $|P_r - P_{ro}|$, and sorts them in ascending order. Then, it selects $\mathcal{C} = mk$, $(m \geq 2)$, random contiguous locations from the sorted locations, which includes $\mathcal{O}$. This set, $\mathcal{C}$, is used as candidate locations for dummy selection. For each location c in $\mathcal{C}$, again selects all the locations within $\frac{\mathcal{R}}{2}$ distance with $P_r > 0$, compute $|P_r - P_{rc}|$, and sorts them in ascending order. From the sorted locations, select a set $\mathcal{S}$ as {first k-1 locations $\cup \mathcal{O}$} and compute its degree of privacy using eq.4.2. Once a set is found whose degree of privacy $\mathcal{E}_f$ satisfies $(\mathcal{E}_f \geq \lambda \times \mathcal{E}_{opt})$, then it is selected as the final set of dummies. The overall process of dummy generation using *photo-check* is presented in Algorithm 3.

4.4.2 Privacy Analysis

Assume a user posted n events $(E_0, \ldots E_{n-1})$ in $(T_0 \ldots T_{n-1})$ time stamps. Out of these, p events were dropped or delayed, q events were posted using dummies, and $w = n - (p + q)$ events were posted directly. Then, the total privacy is,

$$total\ \mathcal{E} = \sum_{i=0}^{p-1} \mathcal{E}_i^{drop/delay} + \sum_{i=0}^{q-1} \mathcal{E}_i^{post} + \sum_{i=0}^{w-1} \mathcal{E}_i^{dummy} \tag{4.4}$$

Here, $\mathcal{E}_i^{type}$ refers to the privacy achieved by applying the *type* of action on i-th event. Now we analyze total privacy for different cases.

Case 1 (None of the events contains location information): In this case no privacy is leaked for each event. That is, $total\ \mathcal{E} = \infty$

Case 2 (All the events contain location information): If location information of all the events are concealed with dummies using the proposed dummy generation algorithm, then, $total\ \mathcal{E} = \sum_{i=0}^{n-1} \mathcal{E}_i^{dummy}$

Case 3 (Some of the events contain dummy locations and others either dropped, delayed, or directly posted): Let us consider three events $E_0, E_1,$ and E_2; where E_0 contains dummy locations, E_1 was dropped, and E_2 was posted directly. E_0's degree of privacy $\mathcal{E}_0^{dummy}$ can be calculated using Eq.4.2. For E_1 and E_2, $\mathcal{E}_1^{drop} \geq \mathcal{E}_0^{dummy}$ and $\mathcal{E}_2^{post} \geq \mathcal{E}_0^{dummy}$, respectively. Thus, $total\ \mathcal{E} \geq \sum_{i=0}^{n-1} \mathcal{E}_i^{dummy}$. By considering all the 3 cases the bound for total privacy,

$$\sum_{i=0}^{n-1} \mathcal{E}_i^{dummy} \leq total\ \mathcal{E} \leq \infty \tag{4.5}$$

4.5 Experimental Evaluation

In this section, we present the detail of the experiment. We use a PC with Intel Core i7 CPU(2.5 GHz), 8GB RAM, and Microsoft Windows 10-64bit operating system

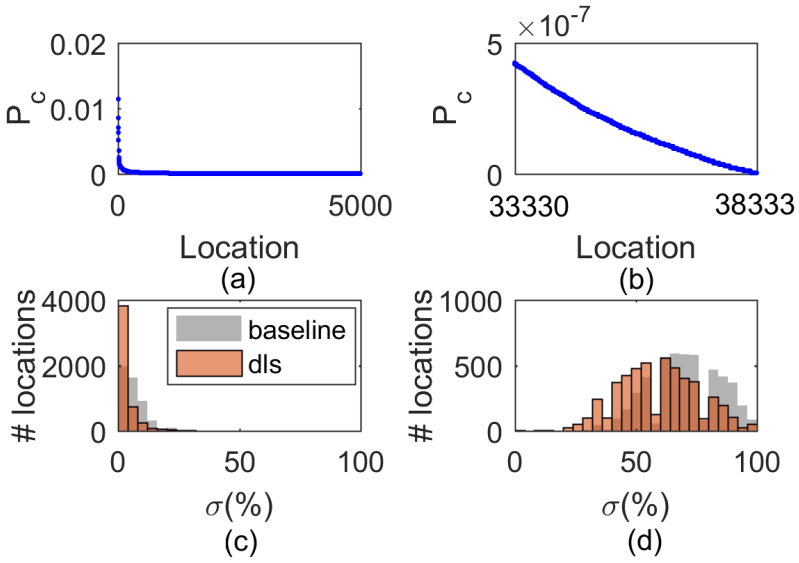


Figure 4.5: Impact of 0/1-probability based inference on DLS and baseline algorithms at locations with different check-in probability ($k = 20$). Here, (a) and (b) are the locations with the highest and lowest check-in probability, respectively.

to carry out the experiment. To get location information, we use Foursquare NYC dataset [YZZY15] which contains 38, 333 unique locations. For each location, we collect check-in and shared photo count using Foursquare developer API. In total, these counts are 162.72 and 4.94 millions, respectively.

For comparison, we consider four different dummy generation approaches: Random, Optimal, DLS[NLZ+14], and Baseline[NLZ+15]. We use the following parameters and their values in the experiment: $k = [8, 12, 16, 20]$, $\delta = [5, 20]$ minutes, $m = 2$, $\lambda = [1, 0.9, 0.8]$, and $\mathcal{R} = 1000$ meter. We use 100 users' data to generate different statistics. Using these parameters, we generate the statistics for privacy analysis and computation cost in two ways: we run each approach on all the events 1) generated by each user and 2) generated from each location, and compute the corresponding mean.

4.5.1 Privacy Analysis

Impact of 0/1- probability based inference model on existing TPPMs

We apply the 0/1 probability based inference model(discussed in section 4.1) on DLS and Baseline approach, and observe a monotonic relationship with location's check-in probability, P_c. For better understanding, we sort the locations based on P_c in ascending order and for the first 5000 locations(Figure 4.5(a)), the inference rate(σ) is least successful(Figure 4.5(c)). However, with the decrease in P_c(Figure 4.5(b)), σ increases(Figure 4.5(d)). Specifically, for DLS and baseline, Spearman's rank-order correlations[McD09] are -0.521 and -0.6, respectively (for last 5000 locations). These results show that existing mechanisms cannot be used for all type of locations to protect location privacy for both check-in and photo sharing in LBSN.

Degree of Privacy

Now, we analyze the degree of privacy $\mathcal{E}$ which measures the uncertainty of an attacker to infer user's real location from a given set $\mathcal{S}$. The comparison of all the approaches in terms of degree of privacy is presented in Figure 4.6 and 4.7. From the above discussion, it is understandable that at locations with high P_c, DLS may sometimes achieve privacy close to our proposed approach. However, it's distribution(Figure 4.6) show that this number is quite small. The comparison plots (Figure 4.7), in terms of different k and users, further justify the superiority of our approach.

Timing, Dropping, and Spatial Errors

We found that the total timing,dropping, and spatial errors are $\approx 5\%$ and 17% for $\delta = [5, 20]$, respectively.

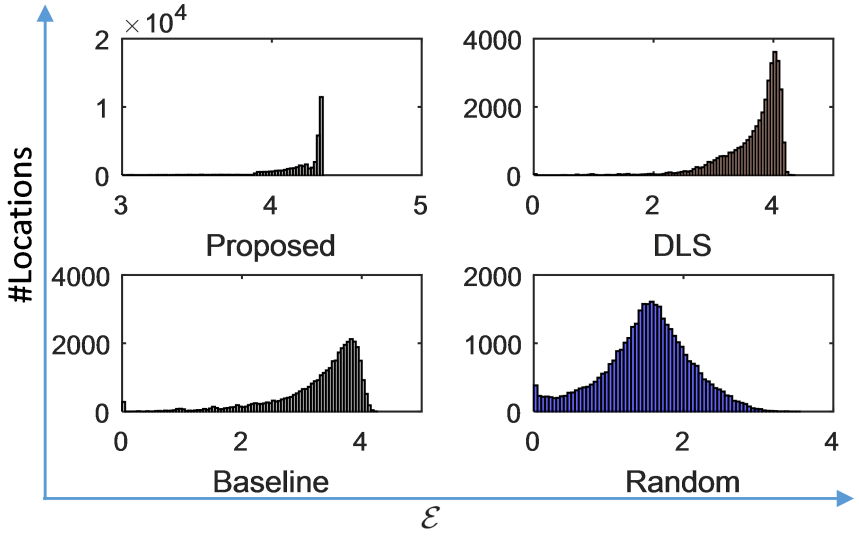


Figure 4.6: Distribution of degree of privacy $\mathcal{E}$ for all the locations with $k = 20$.

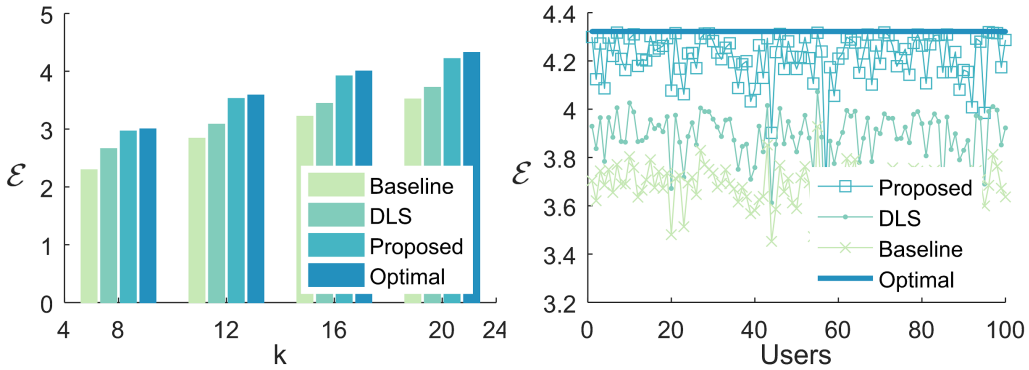


Figure 4.7: Degree of privacy (ϵ) comparison: (a) for different k's and (b) for 100 users with $k = 20$.

Table 4.2: Average Computation Cost(in second) for Different k

Approaches	$k = 8$	$k = 12$	$k = 16$	$k = 20$
Baseline	0.0015	0.0018	0.0017	0.0016
DLS	0.0491	0.0537	0.0535	0.0523
Proposed	**0.0015**	**0.0020**	**0.0024**	**0.0029**

4.5.2 computation cost

The comparison in terms of computation cost among different approaches in shown in table 4.2. As all the approaches have the sorting step, we focus on the number of steps required to generate the dummies. In this context, baseline's computation cost is $O(1)$. The randomized greedy algorithm in proposed approach requires at most mk steps, yielding $O(mk)$ computation cost. However, the major reduction in computation cost in the proposed approach, compared to DLS, is achieved mainly with the introduction of the relation between gained ($\mathcal{E}_s$) and optimal privacy($\mathcal{E}_{opt}$) in the algorithm(line 7 in algorithm 3).

4.6 Discussion and Summary

In this chapter we show that, similar to check-ins, the spatial distribution of historical shared photos can influence user's privacy in LBSN. Based on this observation, we design a probabilistic location inference model for both check-in and photos, and apply it on different existing dummy-based approaches. While this model exposes the vulnerability of existing approaches, it also shows that it is possible to infer user's location without analyzing the photo itself. To solve this problem, we propose a novel privacy-preserving mechanism, called *photo-check*, to preserve location privacy while sharing check-in and photos in LBSN. By analyzing spatiotemporal correlation between consecutive events(generic term for check-in and photo), photo-

check decides whether to delay, post, drop, or insert dummies to an event. In case of dummy generation, we propose a randomized greedy approach which is computationally less expensive than state-of-the-art approach and practical for LBSN scenarios. We believe the findings presented in this work will lead to further research on this issue.

CHAPTER 5

SPATIOTEMPORAL BLOCKCHAIN MANAGEMENT FOR RESOURCE-CONSTRAINED IOT DEVICES TO ACHIEVE DECENTRALIZED PRIVACY

Formation of a peer-to-peer (P2P) network of the IoT devices such that an IoT user can query and get the required information from its peers, rather than a centralized entity, is a promising approach for achieving trajectory privacy from any centralized entity. Recently, blockchain has gained significant attention as a solution to design a P2P network of mobile IoT devices in a way that is designed to be secure, transparent, highly resistant to outages, auditable, and efficient. However, before realizing the promise of blockchain, there are significant challenges to address. One fundamental challenge is the scale issue around data collection, storage, and analytics as IoT sensor devices possess limited computational power and storage capabilities. In particular, since the chain is always growing, IoT devices require more and more resources. Thus, an oversized chain poses storage and scalability problems.

With this in mind, the goal of this chapter is to present the design and implementation of a lightweight, scalable blockchain framework for mobility-centric IoT systems. We focus on certain mobility mobility-centric applications, such as mobile crowdsensing, where an IoT device is not required to have a global view of the whole network. This framework, coined as "Sensor-Chain", promises a new generation of lightweight blockchain management, a significant reduction in resource consumption, and at the same time capable of retaining critical information about the IoT systems of mobile devices. We compare the performance of proof-of-concept implementation of sensor-chain with 3 other schemes, and the results justify its efficacy. Also, the proposed framework does not involve any fixed positioned powerful edge devices, which makes it more flexible with a variety of mobility-based IoT

applications. We further verify Sensor-Chain by implementing it on top of Babble hashgraph blockchain.

The chapter is organized as follows. Section 5.1 introduces the problem and section 5.2 provides the problem statement and our hypothesis. The required background information is discussed in section 5.3. The proposed Sensor-Chain is presented in section 5.4. The evaluation results on its proof-of-concept implementation is presented in 5.5. Finally, section 5.7 concludes the chapter. Last but not least, important symbols used in this chapter are described in table 5.1.

5.1 Introduction

Blockchain is a distributed P2P way of recording digital interactions in a way that it provides built-in integrity of information, and security of immutability by design, making it very useful to ensure trust, security, and transparency in P2P trustless networks of huge number of devices[Nak08, Fra14]. Although blockchain is considered as the key to redesign IoT systems, they cannot be directly integrated into IoT systems. Since the chain is always growing, IoT nodes require more and more resources in order to manage it on their local spaces. Similarly, scalability with constrained computing power and battery also poses a challenge. With an integration of blockchain, each node needs to perform large amount of tasks at different stages of the blockchain with their constrained computing power and battery life. The growth of the network further aggravates the problem. These two issues are rooted to the problem of managing the number of transactions required to be stored and processed by a single IoT node at any time instance, as transactions are the main building blocks of a blockchain. For better understanding of the problem, let us consider a conventional blockchain for a P2P network of n number of nodes. At

any time instance, there could be at most $\frac{n(n-1)}{2}$ number of transactions in the the network. If we express the size of a blockchain (BC) as the number of transactions stored it, then it is quite understandable that, over time, the IoT devices will run out of storage space to store all the transactions.

Understandably, in some applications, the transactions may happen at a longer interval (e.g. 30 minutes). Also, at every timestamp the number of transactions can be less than $\frac{n(n-1)}{2}$. Despite that, it is not going to change the fact that, existing blockchains are not feasible for resource-constrained IoT sensor devices in the long-run. To manage blockchain in a mobility-centric IoT system, many research works proposed to deploy more powerful edge devices and offload the blockchain on such edge devices[XZN+17, MDL+18, PWH+18, KAG18]. These devices are costly and are deployed in a predefined structured way. Such a fixed structure of the static edge devices is hard to be acceptable for several reasons. First, the devices are costly and their deployment over a large region will result into very high cost. Second, in some scenarios, such as in military applications, such fixed structure cannot be attained. Third, we also argue that the management of blockchain using fixed positioned edge devices naturally makes the system more vulnerable, as compromise of few edge devices will affect a large portion of the IoT nodes in the IoT network. Hence, we need a lightweight, scalable blockchain framework without relying on a fixed infrastructure of external edge computing devices. Another promising way of making blockchain lightweight is to delete "historical" blocks from the chain [DOA16]. However, such an approach still consumes large storage space and cannot retain any information of the deleted portion of the chain.

5.2 Motivation and Problem Statement

In this work, we focus on certain mobility-related scenarios where a mobile node is not really required to have a "global" view of a blockchain. Let us consider an environment monitoring mobile crowdsensing application where aggregated data (e.g. temperature, humidity, air quality, and so on) from a small region at a certain time is more important than individual's data. In such a scenario, the mobile nodes at a location may contact each other in a P2P way to collect each other's 1 dimensional (1-d) environmental sensor's value for some time. Then, one node is selected to send the aggregated information in a certain form (e.g. max, mean, average, median, etc.). As the nodes are mobile, the trust value computed for some nodes may not be important at a different location and time for a certain node. Also, the environmental data varies from one region to another; thus instead of having a single network, region based multiple smaller networks, as well as blockchains, are more feasible. Motivated by such a scenario, in this dissertation, we seek answer to the following question: *how can the spatiotemporal mobility of the IoT users allow to make blockchain lightweight for IoT devices?*

We hypothesize that breaking down a traditional blockchain into smaller "local" blockchains based on spatial subdivision of a region, imposing a temporal constraint on their lifecycle, and mobility-based blockchain deletion from storage will allow a blockchain to be lightweight, and scalable for resource-constrained IoT devices.

5.3 Background

In this section, we provide a formal definition of the size of blockchain, a description to the system model, and important assumption made in the work.

Table 5.1: Table of Symbols

Symbol	Description
TX	Abbreviation of transaction
BC	Abbreviation of blockchain
$\mathcal{C}$	Voronoi diagram or set of Voronoi cells
C_i	i-th Voronoi cell or simply cell
n	Total number of sensor nodes
m	Number of sensors in a single cell
G_i^t	Local network in i-th cell at time t
V_i^t	Set of vertices of local network G_i^t
E_i^t	Set of edges between the nodes in V_i^t
S	A sensor node
B_i^t	Local blockchain generated by G_i^t
T_{chain}	Temporal constraint for blockchain
T_{block}	Block generation time constraint

5.3.1 Size of a blockchain

If we take a look at the different elements of a block, we observe that it contains some elements which take constant storage space (the elements of block header and transaction counter). Since each transaction size can be within a limit, its their number which is the only dominating variable in a block. Thus, a size of a blockchain, $size(\text{BC})$, can be expressed as a function of number of transactions, TX, as follows.

$$\begin{aligned} size(\text{BC}) &= \text{TX}^1 + \ldots + \text{TX}^{\text{t}} \\ &= \text{number of TX} \times \text{size of a TX} \\ &= O(\text{number of TX}) \end{aligned} \tag{5.1}$$

Here, TX^t is used as an abbreviation of transaction happened at t-th timestamp. At any time instance, there could at most $\frac{n(n-1)}{2}$ transactions in network of n nodes. Thus, if the lifespan of a blockchain is T,

$$size(\text{BC}) = O\Big(T \times \frac{n(n-1)}{2}\Big) \tag{5.2}$$

This defines the upper bound of the required storage space on a IoT device to hold an entire blockchain in total T time.

5.3.2 System Model and Important Assumptions

The proposed system model has two major entities: 1. a region, divided into a set of smaller cells, and 2. a set of sensor nodes. Some of the sensors are static and others are mobile. The mobile nodes are moving over the region based on Random Waypoint Mobility model[HV07]. Each sensor node is capable of performing lightweight aggregate operations, such as e.g. max, mean, min, weighted average [PPJP12] and so on. Furthermore, the proposed system does not require any additional resources. We assume that the distribution of the sensed data within a cell is approximately same. The proposed blockchain can be either a public blockchain or a permissioned-blockchain. If it is public blockchain, there is no authority in the blockchain and nodes can join and leave the network with random cryptographic key pairs. In such a blockchain, we assume that the nodes are using a lightweight consensus algorithm, such as Proof-of-Stake (PoS). Our work is also applicable in permissioned-blockchain where an authority assigns each IoT Node a private key and a private key and to join a network a node needs to reveal its identity to all the other nodes in the network. In order to achieve conditional privacy from the peers, an IoT node can anytime request the authority for new key pairs. In such a case, we assume that the nodes are using a Byzantine Fault tolerant algorithm for reaching consensus in the network. Devising mechanisms for Key management and authentication are beyond the scope of this work.

5.4 Proposed Sensor-Chain Framework

This section presents the Sensor-Chain framework. We first discuss 3 different frameworks: Conventional and our proposed improved temporal, and spatial blockchains. We analyze their strength and limitations to highlight the motivation behind the design of Sensor-Chain framework.

5.4.1 Naive Approach: Conventional Blockchain

In the conventional blockchain frameworks [XZN+17, MDB17] a blockchain is managed by all the nodes in the network and continues to grow with the lifespan of the network. Thus, with a $T = \infty$ lifespan, according to our discussion on the size of blockchain, the size of a conventional blockchain becomes,

$$size(\text{conventional}) = \sum_{t=1}^{\infty} \frac{n(n-1)}{2} \tag{5.3}$$

Obviously, this blockchain will impose a high storage requirement which cannot be met by sensor nodes. To improve this, we then design an improved version of temporal blockchain[DOA16] in the context of mobile IoT.

5.4.2 Our First Approach: Improved Temporal Blockchain

In the original temporal blockchain[DOA16], it was proposed to keep a portion of the blockchain after certain time period. However, we propose to replace the blockchain with an aggregated version of it after certain a time period. In detail, in the preprocessing step of our scheme, we consider a specific time at the "genesis time", and a time period is set as the temporal constraint for blockchain deletion. For example, if 00:00 in 24-hour format is taken as the genesis time and the temporal constraint is 2 hours, then the deletion operation will take place at 02:00, 04:00,

06:00, ... of each day. This genesis time information and temporal constraint are preset onto the IoT devices. Another way to set this information is to have smart contract on the blockchain. We leave this for our future research. Every time the lifetime of the blockchain meets the temporal constraints, through the consensus mechanism,a node will be selected as an aggregator node which performs aggregation over the whole blockchain and creates an aggregated block. This block includes the ID of the aggregator node. This block is then broadcasted over the network by the aggregator. This aggregation could be anything lightweight for IoT sensor devices to perform (e.g. min, max, mean, weight average [PPJP12]).

Upon receiving this block, the nodes in the network replaces the whole existing blockchain with this block on their local storage. That is, it will considered as the genesis block of a new blockchain. Even though as a consequence the newly restarted blockchain's size becomes relatively small, we still need to look into the size of the blockchain between two consecutive restarts so as to ensure that it is withing the storage space capacity of the IoT sensor node. If the temporal constraint is T_{chain}, then in the the worst case scenario, the maximum size of the blockchain can be,

$$size(\text{improved-temporal}) = \sum_{t=1}^{t=T_{\text{chain}}} \frac{n(n-1)}{2} \tag{5.4}$$

Clearly this scheme outperforms the conventional blockchain schemes. However, with higher T_{chain} and a large number of nodes in a network, the nodes still need to hold a large blockchain, making it quite impractical for IoT devices. Thus, despite the fact that a temporal blockchain can reduce the size of a chain, the size of a chain must be further improved when dealing with IoT nodes. This is done using the following spatial blockchain technique.

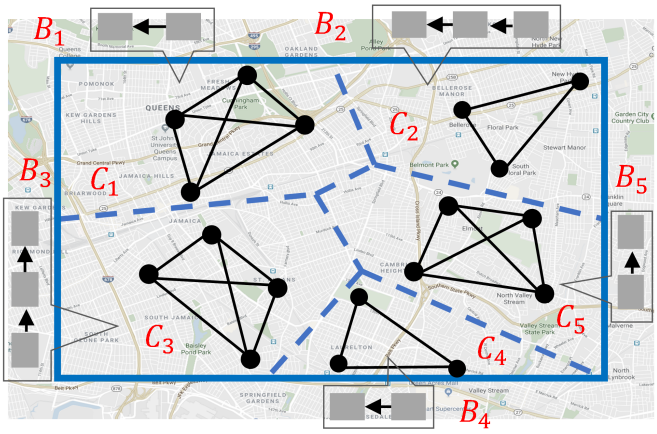


Figure 5.1: Proposed blockchain-based IoT architecture: A region is transformed into a Voronoi diagram where C_i is the i-th Voronoi cell and the graph inside of it is a local network G_i. '•'s and'−'s represent nodes and edges between the nodes, respectively. B_i refers to the local blockchain in cell C_i.

5.4.3 Our Second Approach: Spatial Blockchain

In our spatial blockchain framework, a global blockchain is broken down into smaller disjoint *local blockchains* with the aim of reducing the number of transactions performed by a node at any given time than in conventional blockchain frameworks. To achieve this objective, we translate a region into a Voronoi diagram[AIRX08]. Voronoi diagram $\mathcal{C}$, is a partitioning of a plane into non-overlapping smaller convex regions, called Voronoi cells $\mathcal{C}$.

Based on this partitioning of the plane, we define two different structures: local networks and local blockchains (figure 5.1 depicts these structures). A *local network* refers to the graph $G_i^t = (V_i^t, E_i^t)$ formed by the nodes in the cell $C_i \in \mathcal{C}$ at time t. Here, V_i^t and E_i^t are the set of the nodes and the edges between them. Any two local networks of two different cells at the same time are disjoint. That is,

$$V_i^t \cap V_j^t = \emptyset, \quad E_i^t \cap E_j^t = \emptyset \tag{5.5}$$

A *local blockchain* B_i, is the blockchain managed by the nodes in cell C_i and B_i^t is

the snapshot of B_i at time t. Any two local blockchains from two different cells have the following property: a block of a local blockchain in a cell is neither a parent nor a child of a block of another local blockchain in another cell at any time instance. That is,

$$\begin{aligned}(\exists b_i^x \in B_i | b_i^x \text{ is a parent of a block in } B_j) \cup \\ (\exists b_j^y \in B_j | b_j^y \text{ is a parent of a block in } B_i) = \emptyset; \forall t\end{aligned} \tag{5.6}$$

The two properties imply that a sensor node in G_i works only on local blockchain B_i. Hence, it needs to store only the copy of B_i at any given time as long as it remains in G_i. While this definitely improves the storage issue than in conventional blockchain, this scheme further enhance its efficacy by considering mobility of the nodes. In case of mobility, if a node moves from cell C_i to C_j, at first it deletes the copy of local blockchain B_i from its memory and then, after joining G_j, it downloads the copy of B_j from its peers. Thus, a node is required to store only one local blockchain at any time instance, which significantly reduces the required space to store a blockchain. We quantify the storage requirement of this scheme as follows. Let us consider that at any time instance, there could be at most m number of nodes in a cell, where $m < n$ and the time difference between the creation of genesis block and current time is $\approx \infty$. Let us also assume that a mobile node's permanence in a cell is at most T_{per}. At the first glance, it seems $size(\text{spatial}) = \sum_{t=1}^{t=T_{per}} \frac{m(m-1)}{2}$. However, consider the worst case scenario where there exists at least one node in a particular cell C_i all the time (if some nodes are static or the cell is never empty). That is, the local blockchain continues to expand forever. In that case,

$$size(\text{spatial}) = \sum_{t=1}^{t=\infty} \frac{m(m-1)}{2}; \quad m < n \tag{5.7}$$

From the analysis of temporal and spatial blockchains, it is not clear which one offers the best solution. For static nodes, the temporal blockchain with a small

temporal constraint could be the better solution in the long run. On the other hand, in mobile environment, the spatial blockchain will be the winner. To address the limitations of both approaches, we propose Sensor-Chain approach.

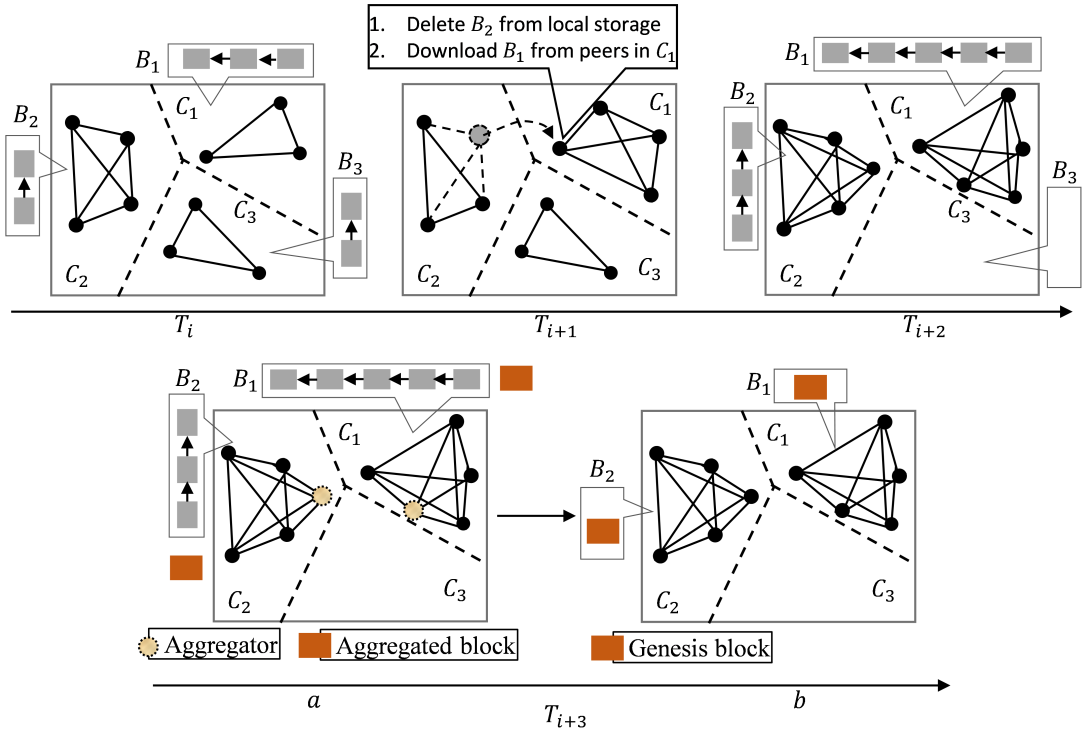


Figure 5.2: Illustrated Sensor-Chain:-T_{i+1}: A mobile node moves from cell C_2 to C_1. First, it deletes the copy of local blockchain B_2 from its memory and then downloads B_1 from its peers in G_1^{i+1}. T_{i+2}: local blockchain B_3 does not exist anymore as C_3 is empty. T_{i+3}: as temporal constraint is met, (a) aggregator node from each local network is selected. The selected nodes compute aggregation over their respective local blockchains and generate aggregated blocks. (b) using the aggregated blocks as the genesis, the local blockchains are regenerated.

5.4.4 Our Best Approach: Sensor-Chain

Sensor-Chain is a fusion of both temporal and spatial blockchain approaches. Similar to spatial blockchain, in this framework, a complete region is first divided into a number of Voronoi cells. Using those cells, the nodes in a cell form a local network

and maintain a local blockchain. All the local networks and local blockchains follow the properties defined for spatial blockchain. Among different information, each nodes holds the following tuple: {*current cell id* C_{cur}*, copy of the local blockchain* B^t_{cur}}. In order to manage the size of a blockchain, this framework has two important constraints: temporal constraint T_{chain} and block creation time constraint T_{block}. The storage management of blockchain is done in two ways: spatiotemporal and mobility-based.

Spatiotemporal-based blockchain management is detailed in algorithm 4. In this framework, the block creation and insertion are done at a fixed time interval (lines 1-6), a similar approach of bitcoin. At first, in each local network G^t_i a $Miner$ is selected through consensus. Then the $Miner$ gathers all the recent transactions and creates $NewBlock$. Upon verification, the new block is inserted into B^t_i. The temporal constraint is used to reset the local blockchains at a fixed time interval. Every time the temporal constraint is met (line 8), an *Aggregator* node is selected from each local network. This *Aggregator* node computes aggregation of its local blockchain, creates an *AggregatedBlock*, and broadcasts it over its local network (lines 9-13). Upon receiving the *AggregatedBlock*, the nodes in the local network first delete their copy of the existing local blockchain (line 14) and then regenerate the local blockchain using the aggregated block as the genesis block (line 15).

Algorithm 5 presents the mobility-based blockchain management. Every time a node moves from one cell C_{cur} to another C_{new_cell} (line 1), it deletes the copy of the local blockchain B_{cur} of previous cell from its memory. Then it joins the The work flow of Sensor-Chain is illustrated in figure 5.2.

Algorithm 4: Spatiotemporal Blockchain Management

Input : Current time T_t, set of all local networks $\mathcal{G}$ at T_t, set of all local blockchains $\mathcal{B}^t$, genesis time T_{gen}, temporal constraint T_{chain}, block creation time constraint T_{block}

Output: Updated local blockchains $\mathcal{B}^t$

```
if (T_gen − T_t)%T_block == 0 then
    for each G_i^t ∈ 𝒢^t do
        Miner ← Select-Miner(V_i^t)
        NewBlock ← Create-Block(Miner)
        Insert-Block(B_i^t, NewBlock)
    end
end
if (T_gen − T_t)%T_chain == 0 then
    for each G_i^t ∈ 𝒢^t do
        Aggregator ← Select-Aggregator(V_i^t)
        AggregatedBlock ← Compute-Aggregation(B_i^t, Aggregator)
        Broadcast(AggregatedBlock)
        for each node v ∈ V_i^t do
            Delete(B_i^t) from local storage
            B_i^t ← Re-generate(B_i^t, AggregatedBlock)
        end
    end
end
```

Algorithm 5: Mobility-Based Blockchain Management

Input : Voronoi diagram $\mathcal{C}$, sensor node S

Output: Updated node S

```
if S.C_cur ≠ C_new_cell then
    Delete(B_cur) from local storage
    S.C_cur ← C_new_cell
    Join(G_cur^t)
    Download(B_cur^t) from peers in V_cur^t
end
```

5.4.5 Analysis

We argue that, with such spatiotemporal and mobility-based blockchain management, Sensor-Chain provides the best solution. To prove its validity, we now analyze the space requirement to store a blockchain in this scheme. Referring to the discussion on spatial blockchain, with the space partitioning, the size of a local blockchain in Sensor-Chain can be at most,

$$size(\text{Sensor-Chain}) = \sum_{t=1}^{t=\infty} \frac{m(m-1)}{2} \tag{5.8}$$

However, as the temporal constraint T_{chain} is applied to all the local blockchains, according to the discussion on temporal blockchain, the size of a local blockchain can be further reduced as follows,

$$size(\text{Sensor-Chain}) = \sum_{t=1}^{t=T_{chain}} \frac{m(m-1)}{2} \tag{5.9}$$

This analysis gives us the required storage space in Sensor-Chain. Next, we analyze the scheme case by case and draw comparison with our proposed improved temporal and spatial blockchain frameworks.

Case 1: In the first case, all the nodes are assumed as static. Also, the partitioning of the region is such that all the nodes reside in a single cell. In such a case, $m = n$.

$$size(\text{Sensor-Chain}) = size(\text{improved-temporal}) = \sum_{t=1}^{t=T_{chain}} \frac{n(n-1)}{2} < size(\text{spatial}) \tag{5.10}$$

Where $size(\text{spatial}) = \sum_{t=1}^{t=\infty} \frac{n(n-1)}{2}$.

Case 2: All the nodes are moving in such a way that each local blockchain becomes empty (more correctly, it doesn't exist anymore) every time before the

temporal constraint is satisfied. This case is depicted in figure 5.2(T_{i+2}) where cell C_3 is empty so that B_3 does not exist anymore. In such a case,

$$size(\text{Sensor-Chain}) = size(\text{spatial}) = \sum_{t=1}^{t<T_{chain}} \frac{m(m-1)}{2} < size(\text{improved-temporal}) \quad (5.11)$$

Where $m < n$ and $size(\text{improved-temporal}) = \sum_{t=1}^{t=T_{chain}} \frac{n(n-1)}{2}$.

All other cases: In all other cases,

$$(size(\text{Sensor-Chain}) < size(\text{spatial})) \& (size(\text{Sensor-Chain}) < size(\text{improved-temporal})) \quad (5.12)$$

5.5 Proof-of-Concept Evaluation

This section presents the experimental results. To carry out the experiment we use synthetic data. The parameters and their different values used in the experiment are presented in table 5.2. We implemented all the four (conventional, improved-temporal, spatial, and Sensor-Chain) approaches. We ran the simulation for 6 hours and generated statistics for all the approaches. Specifically, we compared the approaches in terms of number transactions needed to be stored on a single IoT sensor device, as it defines the size of a blockchain. The evaluation is done from three different points of view: 1. duration of the simulation, 2. number of cells, and 3. number of sensors to analyze the benefit of Sensor-Chain in the long-run and scalability. The detail of the evaluation results are discussed below.

Figure 5.3(a) shows the result of the simulation for Sensor-Chain. In every hour, the curve moves upward. As $T_{chain} = 1$ hour, the size of the blockchain becomes 1 (with the aggregated block) at the end of each hour. It is also clear that in Sensor-Chain, using the temporal constraint, it is possible to keep the size of the blockchain

Table 5.2: Parameters used in the Experiment

Parameter	Values
Area of the region	$5000m \times 5000m$
Number of Voronoi cells	$50, 100, 150, 200, 1000$
Number of sensor nodes	$1000, 3000, 5000, 7000$
Speed of the nodes	$[0, 50]$ km/h
Temporal constraint T_{chain}	1 hour
Block creation time constraint T_{block}	10 minute

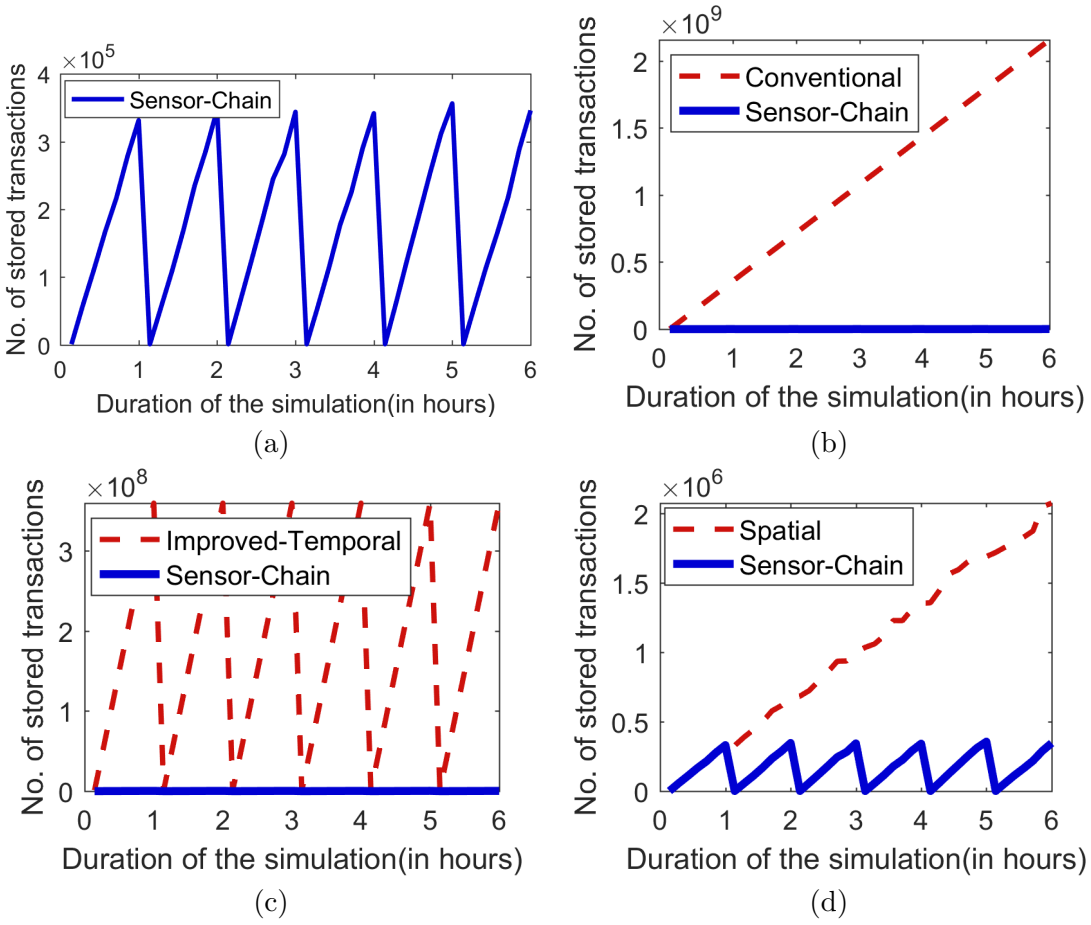


Figure 5.3: Evaluation results: (a) Sensor-Chain, (b) conventional, (c) improved-temporal, and (d) spatial blockchains (experiment Settings: number of cells = 50, number of sensors = 1000).

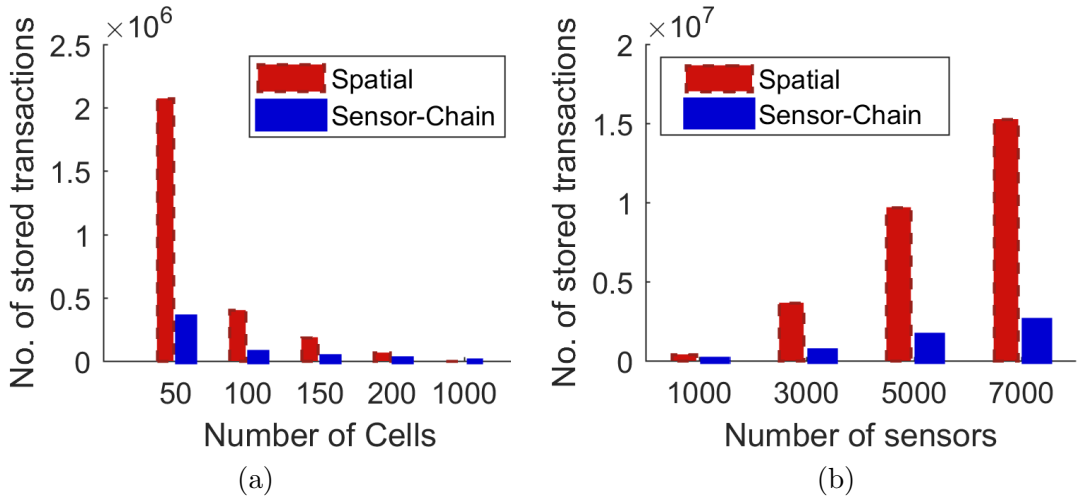


Figure 5.4: Comparison between Sensor-Chain and spatial approaches in terms of number of (a) cells and (b) sensors.

within a limit. Figure 5.3(b) shows the comparison between Sensor-Chain and conventional approaches. From nearly the beginning of the simulation, the required storage space in Sensor-Chain is far less than in conventional approach. Next, we evaluate how Sensor-Chain, with the fusion of spatiotemporal and mobility-based blockchain management, outperforms the improved temporal and spatial schemes. For both of the improved temporal and Sensor-Chain, we used the same temporal constraint. Although the improved temporal blockchain shows a trend similar to Sensor-Chain, its required storage space is much higher than Sensor-Chain. Figure 5.3(d) shows more interesting results on the comparison with spatial blockchain. In the 1st hour, both spatial and Sensor-Chain approaches go toe-to-toe. However, just after the 1st hour (as $T_{chain} = 1$ hour), the local blockchains in Sensor-Chain restore to genesis block, while spatial blockchain continues to grow over the time.

Then, we analyze the impact of number of cells and sensors on the size of the blockchain. As only spatial and Sensor-Chain use cell-based partitioning, here we analyze their comparison. Figure 5.4(a) presents the comparison result in terms

of number of cells. It is understandable that with the increase in the number of cells, the size of a local blockchain decreases. Furthermore, it seems that when this number is relatively high (e.g. 1000 in the figure), both approaches require similar storage capacity. However, it is the number of sensors that makes the difference in such a particular case. With the increase in the number of sensors, the required storage space increases rapidly in spatial approach than in Sensor-Chain. Figure 5.4(b) shows the results for 1000 cells with different number of sensors.

5.6 Implementation Detail of Sensor-Chain

While the theoretical analysis and simulation results highlight the efficiency and efficacy of Sensor-Chain, we are developing the platform for P2P networks of mobile devices. In this section we present the detail of the implementation steps of Sensor-Chain. The development is being carried out with Go programming language, an open source programming language. For P2P communication, we use `go-libp2p-pubsub` library [why19], an open source golang implementation of pub-sub system with flooding and gossiping variants. Figure 5.5 presents the key components of the Sensor-Chain platform. Figures 5.6 5.7, 5.8, and 5.9 illustrate the class, architecture, sequence, and use case diagrams of the platform.

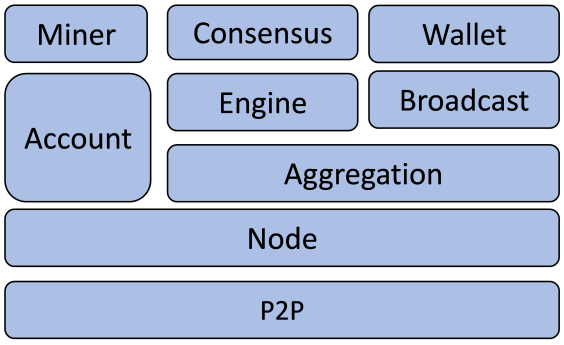


Figure 5.5: Key components of Sensor-Chain platform.

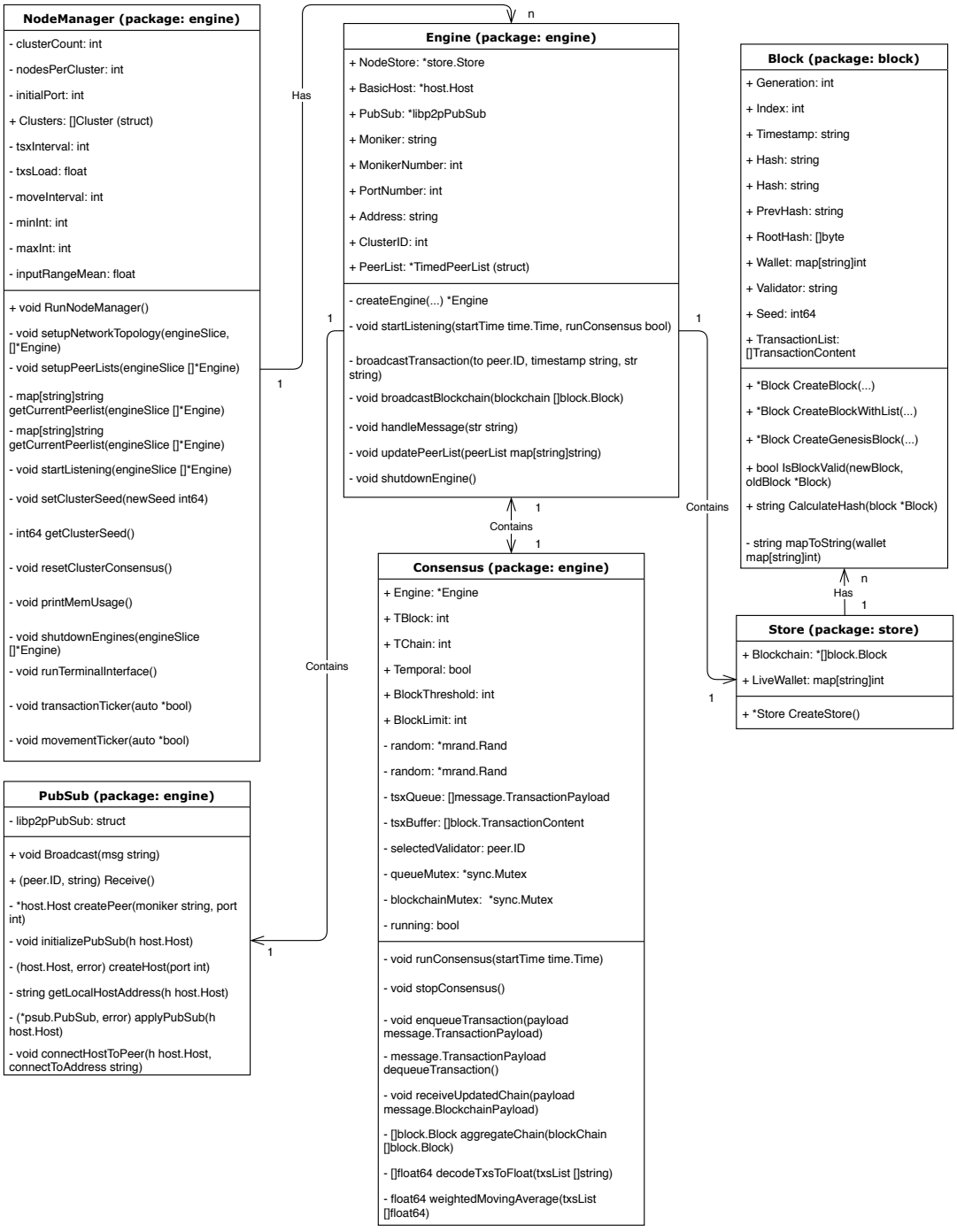


Figure 5.6: Class diagram of Sensor-Chain.

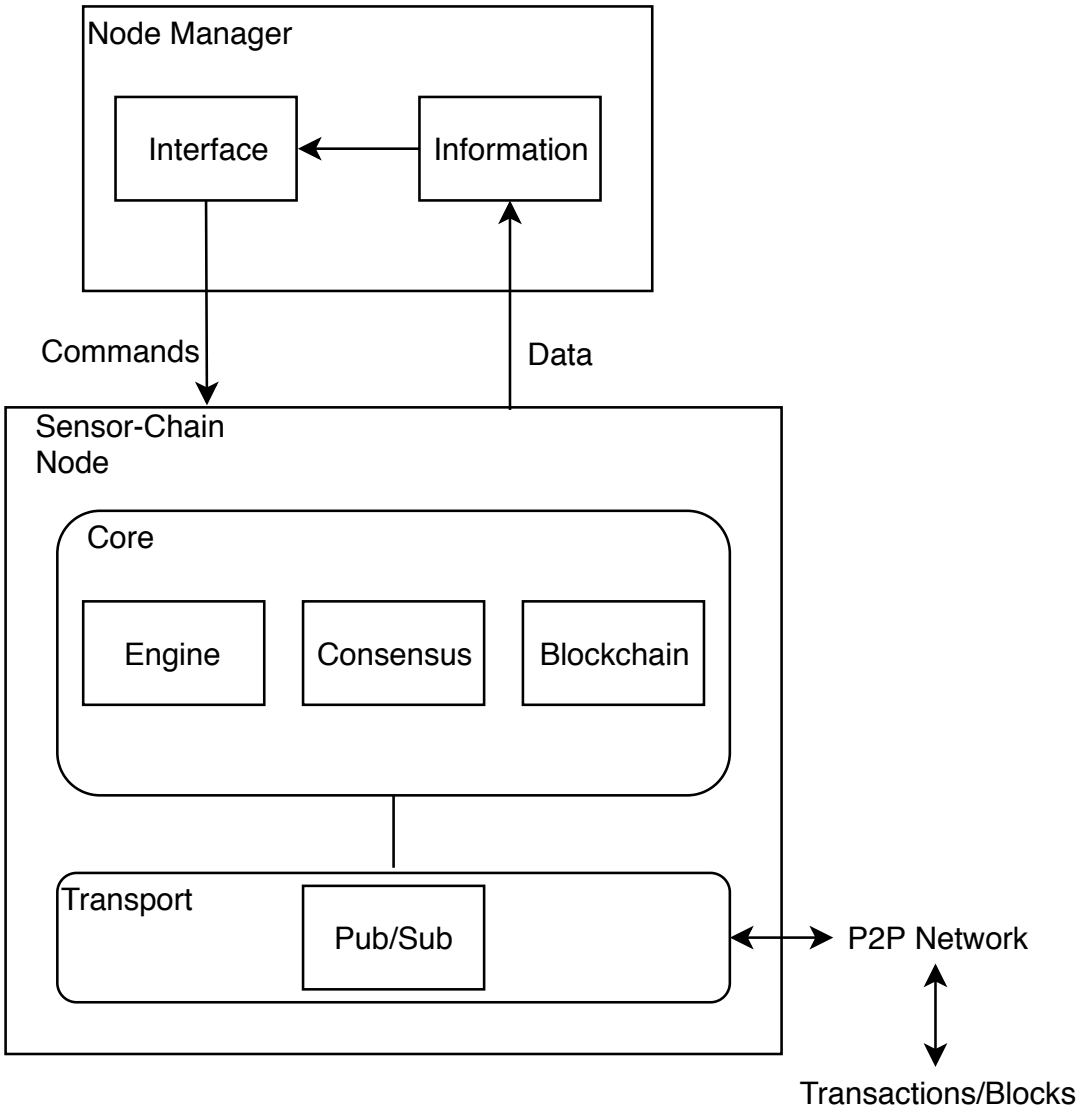


Figure 5.7: Architecture diagram of Sensor-Chain.

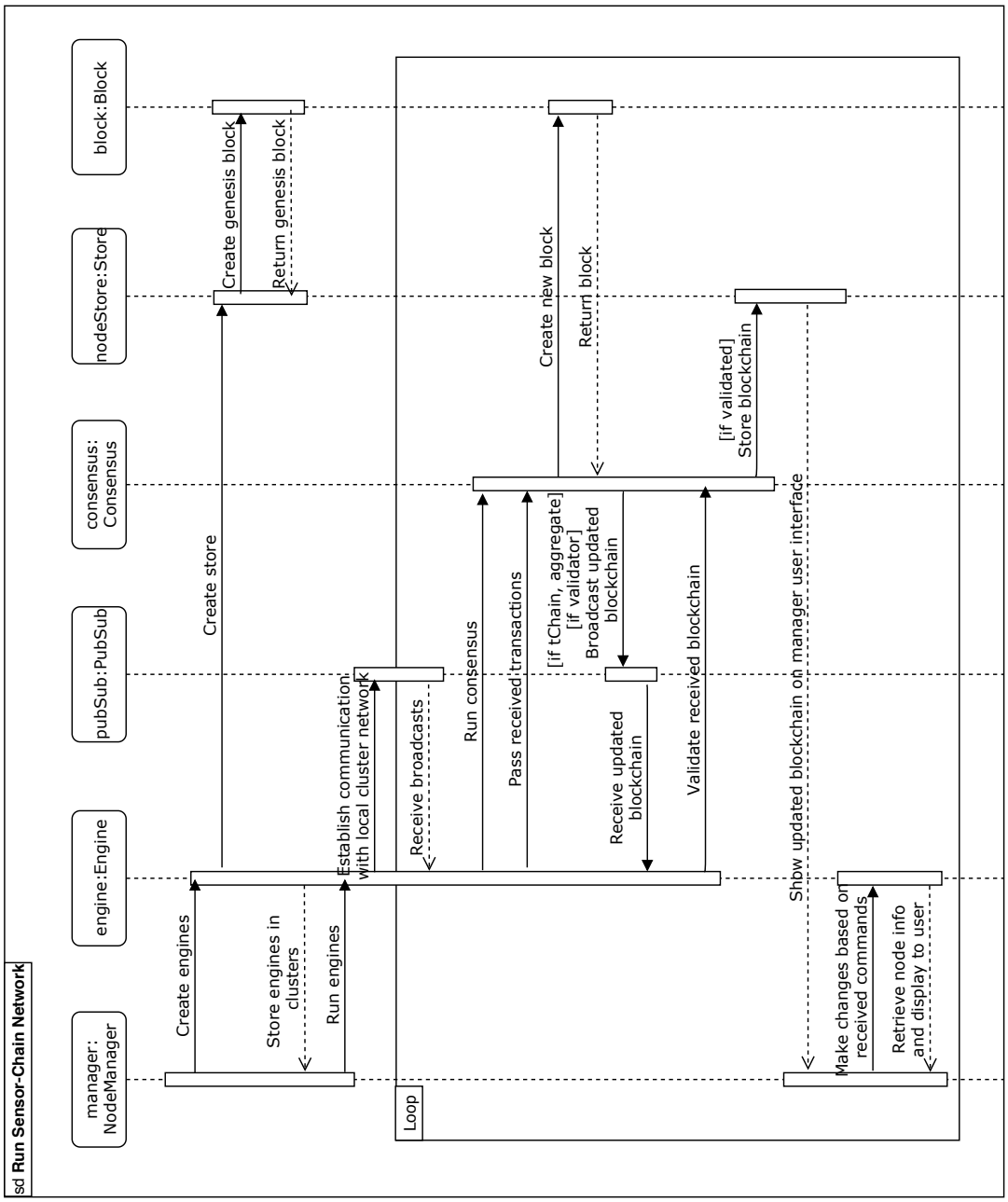


Figure 5.8: Sequence diagram of Sensor-Chain.

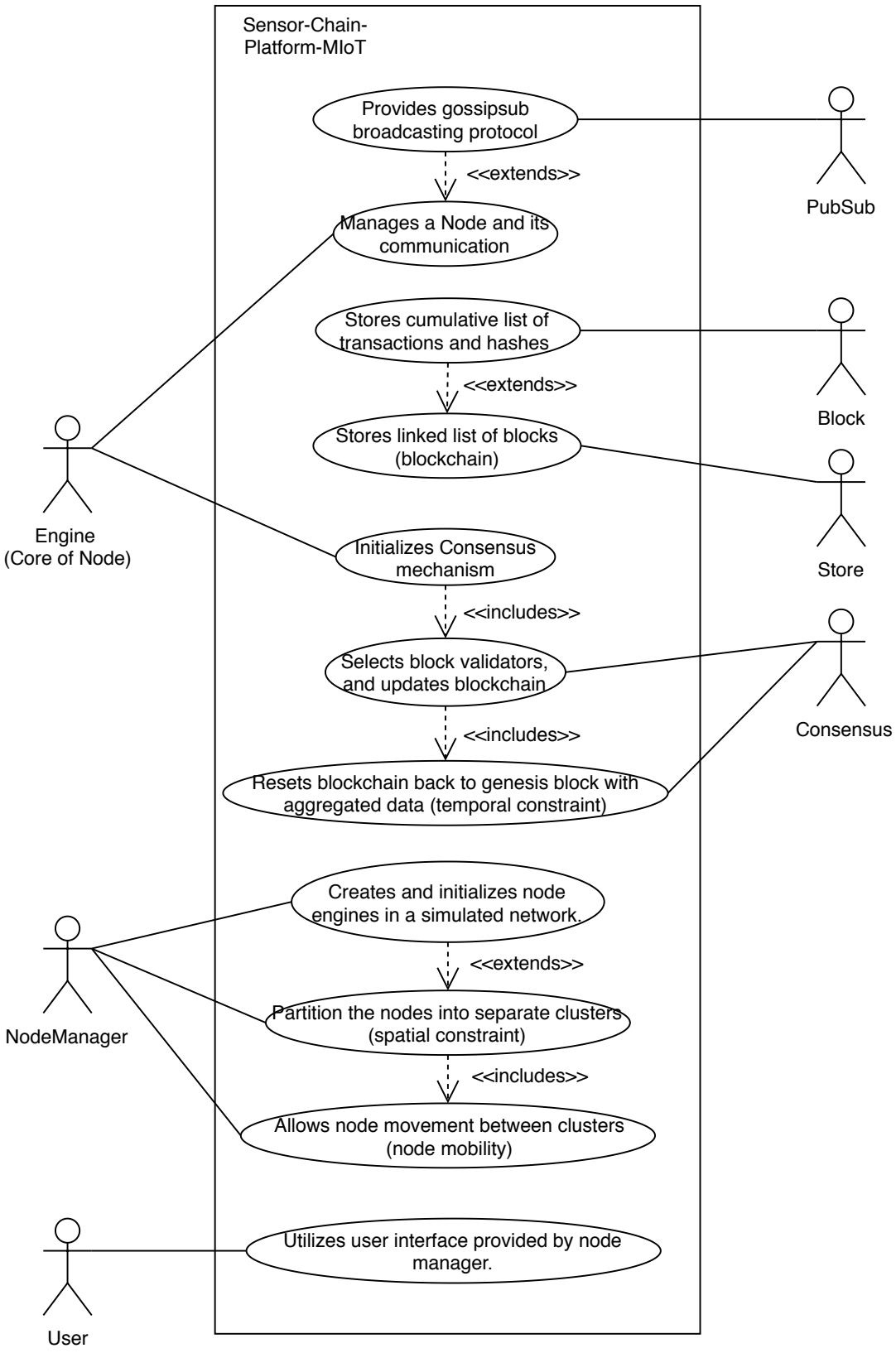


Figure 5.9: Use case diagram of Sensor-Chain.

5.6.1 Key Components

Node

The `Node` struct has two fields: `Blockchain` and `LiveWallet`. The `Blockchain` field is a pointer to the current blockchain that a `Node` is holding in its memory. The `LiveWallet` is a `map` object containing the new transactions happened in the Blockchain by the nodes which are not included in the `Blockchain` yet.

```
type Node struct {
        Blockchain *[]block.Block
        LiveWallet map[string]int
}
func CreateNode() *Node {
        var newNode Node
        newNode.Blockchain = &[]block.Block{}
        newNode.LiveWallet = make(map[string]int)
        *newNode.Blockchain = append(*newNode.Blockchain,
        ↪  *block.CreateGenesisBlock(0, nil, "", nil, "", 0))
        return &newNode
}
```

Transaction Content

`TransactionContent` implements the Content interface provided by merkletree and represents the content stored in the tree. There are three functions to facilitate different operations related to transactions: `CalculateHash()`, `Equals()`, and `String()`. Specifically, `CalculateHash()` hashes the values of a `TransactionContent`, `Equals()` tests for equality of two contents, and `String()` returns a string representation of the content.

```
type TransactionContent struct {
        From        string
        To          string
        Timestamp   string
        Transaction string
```

```
}
func (t TransactionContent) CalculateHash() ([]byte, error) {
        h := sha256.New()
        if _, err := h.Write([]byte(fmt.Sprintf("%s%s%s%s", t.From, t.To,
        ↪  t.Timestamp, t.Transaction))); err != nil {
                return nil, err
        }
        return h.Sum(nil), nil
}

func (t TransactionContent) Equals(other merkletree.Content) (bool, error)
↪  {
        return t.From == other.(TransactionContent).From && t.To ==
        ↪  other.(TransactionContent).To &&
                t.Timestamp == other.(TransactionContent).Timestamp &&
                ↪  t.Transaction ==
                ↪  other.(TransactionContent).Transaction, nil
}
func (t TransactionContent) String() string {
        return fmt.Sprintf("From: %s, To: %s, Timestamp: %s, Transaction:
        ↪  %s", t.From, t.To, t.Timestamp, t.Transaction)
}
```

Block

A blockchain is defined by `Block` struct type. The `Block` contains `BlockHeader` and `TransactionContent`. `BlockHeader` holds the Block struct contents which are hashed for blockchain integrity. Among different variables, the `BlockHeader` contains `RootHash` Merkle tree root hash which is composed of the concatenated hashes of all transactions in block.

```
type Block struct {
        Header          BlockHeader
        TransactionList []TransactionContent
}
```

```
type BlockHeader struct {
        Generation int
        Index      int
```

```
        Timestamp  string
        Hash       string
        PrevHash   string
        RootHash   []byte
        Wallet     map[string]int
        Validator  string
}
```

The `CreateBlock()` function is used to create a new block using the previous block hash, and appends one payload.

```
func CreateBlock(oldBlock *Block, from string, to string, timestamp
↪  string, transaction string, wallet map[string]int, validator peer.ID,
↪  seed int64) *Block {
        var newBlock Block
        var contentList []merkletree.Content
        for _, c := range oldBlock.TransactionList {
                contentList = append(contentList, c)
        }
        contentList = append(contentList, TransactionContent{From: from,
        ↪  To: to, Timestamp: timestamp, Transaction: transaction})
        tree, err := merkletree.NewTree(contentList)
        if err != nil {
                log.Fatal(err)
        }
        validRootHash, err := tree.VerifyTree()
        if err != nil {
                log.Fatal(err)
        }
        if !validRootHash {
                err := tree.RebuildTree()
                if err != nil {
                        log.Fatal(err)
                }
                validRootHash, err = tree.VerifyTree()
                if err != nil {
                        log.Fatal(err)
                }
                if !validRootHash {
                        log.Fatalln(" Failed to build correct merkle
                        ↪  tree.")
                }
        }
        var tsxContentList []TransactionContent
```

```
        for _, c := range contentList {
                tsxContentList = append(tsxContentList,
                ↪  c.(TransactionContent))
        }
        newBlock.TransactionList = tsxContentList

        newBlock.Header = BlockHeader{
                Generation: oldBlock.Header.Generation,
                Index:      oldBlock.Header.Index + 1,
                Timestamp:  time.Now().Format(time.RFC1123),
                Hash:       "",
                PrevHash:   oldBlock.Header.Hash,
                RootHash:   tree.MerkleRoot(),
                Wallet:     wallet,
                Validator:  validator.Pretty(),
                Seed:       seed,
        }
        newBlock.Header.Hash = CalculateHash(&newBlock)

        return &newBlock
}
```

A node uses the `IsBlockValid` function to make sure block is valid by checking index, and comparing the hash of the previous block. The hashing of the contect of the `Block` struct is performed using SHA256.

```
func IsBlockValid(newBlock, oldBlock *Block) bool {
        if oldBlock.Header.Index+1 != newBlock.Header.Index {
                return false
        }
        if oldBlock.Header.Hash != newBlock.Header.PrevHash {
                return false
        }
        if CalculateHash(newBlock) != newBlock.Header.Hash {
                return false
        }
        var contentList []merkletree.Content
        for _, c := range newBlock.TransactionList {
                contentList = append(contentList, c)
        }
        tree, err := merkletree.NewTree(contentList)
        if err != nil {
                log.Fatal(err)
```

```
        }
        validRootHash, err := tree.VerifyTree()
        if err != nil {
                log.Fatal(err)
        }
        if !validRootHash {
                err := tree.RebuildTree()
                if err != nil {
                        log.Fatal(err)
                }
                validRootHash, err = tree.VerifyTree()
                if err != nil {
                        log.Fatal(err)
                }
                if !validRootHash {
                        log.Fatalln(" Failed to build correct merkle tree
                        ↪  .")
                }
        }
        if fmt.Sprintf("%x", newBlock.Header.RootHash) !=
        ↪  fmt.Sprintf("%x", tree.MerkleRoot()) {
                return false
        }
        return true
}
```

```
func CalculateHash(block *Block) string {
        record := strconv.Itoa(block.Header.Index) +
        ↪  block.Header.Timestamp + fmt.Sprintf("%x",
        ↪  block.Header.RootHash) +
                block.Header.PrevHash + MapToString(block.Header.Wallet)
        h := sha256.New()
        h.Write([]byte(record))
        hashed := h.Sum(nil)
        return hex.EncodeToString(hashed)
}
```

Consensus

`Consensus` is the struct which handles the consensus mechanism of the platform. It defines the `TBlock`, the time for blockchain creation in seconds, `TChain`, the time for blockchain aggregation in seconds, and `BlockThreshold`, minimum number of

blocks needed before aggregation. This also implements a `transactionQueue`, a transaction pool to holds the new transactions in the network. In the current version, a randomized mechanism is implemented as a consensus using `RunConsensus()` function. The `ReceiveUpdatedChain()` accepts broadcasted blockchains, validates them, and checks if they were sent by the validator. The `NewConsensus()` function creates a new `consensus` which manages the consensus mechanism. The nodes run the function `RunConsensus()` to run a consensus.

```
type Consensus struct {
        Engine         *Engine
        TBlock         int
        TChain         int
        BlockThreshold int
        random           *mrand.Rand
        transactionQueue []message.TransactionPayload
        electedValidator peer.ID
        queueMutex       *sync.Mutex
        blockchainMutex  *sync.Mutex
        running          bool
}
```

```
func NewConsensus(engine *Engine, tBlock int, tChain int, blockThreshold
↪  int) *Consensus {
        consensus := &Consensus{
                Engine:         engine,
                TBlock:         tBlock,
                TChain:         tChain,
                BlockThreshold: blockThreshold,
                random:           mrand.New(mrand.NewSource(0)),
                transactionQueue: []message.TransactionPayload{},
                electedValidator: "",
                queueMutex:       &sync.Mutex{},
                blockchainMutex:  &sync.Mutex{},
                running:          false,
        }

        return consensus
}
```

```
func (c *Consensus) RunConsensus() {
        c.running = true
        c.blockchainMutex.Lock()
        blockchain := c.Engine.HostNode.Blockchain
        lastBlock := (*blockchain)[len(*blockchain)-1]
        c.blockchainMutex.Unlock()
        lastBlockTime, err := time.Parse(time.RFC1123,
        ↪  lastBlock.Header.Timestamp)
        if err != nil {
                log.Fatal(err)}

        nextTBlockTime := lastBlockTime.Add(time.Second *
        ↪  time.Duration(c.TBlock))
        nextTChainTime := lastBlockTime.Add(time.Second *
        ↪  time.Duration(c.TChain))

        c.random.Seed(lastBlock.Header.Seed)

        go func() {
                for c.running {
                if (time.Now().After(nextTBlockTime) ||
                ↪  time.Now().Equal(nextTBlockTime)) &&
                ↪  len(c.transactionQueue) > 0 {
                                buffer := []block.TransactionContent{}
                                c.electedValidator = ""
                                c.queueMutex.Lock()
                                for len(c.transactionQueue) > 0 {
                                t := c.dequeueTransaction()
                                if _, found := c.Engine.
                                HostNode.LiveWallet
                                        [t.From.Pretty()];
                                        found {
                                                c.Engine.HostNode.
                                                LiveWallet
                                                [t.From.Pretty()]++
                                        } else {
                        c.Engine.HostNode.LiveWallet
                                                [t.From.Pretty()] = 1
                                        }

                                        buffer = append(buffer,
                                        ↪  block.TransactionContent
                                        {From: t.From.Pretty(), To:
                                        ↪  t.To.Pretty(), Timestamp:
                                        ↪  t.Timestamp, Transaction:
                                        ↪  t.Transaction})
```

```
}
c.queueMutex.Unlock()
candidates := []string{}

for key := range c.Engine.
PeerList.Peers {
        candidates = append(candidates,
        ↪  key)
}
sort.Strings(candidates)
validator := candidates[c.random.Intn
(len(candidates))]

pid, err := peer.IDB58Decode(validator)
if err != nil {
        log.Fatal(err)
}
c.electedValidator = pid

c.blockchainMutex.Lock()
lastBlock =
↪  (*blockchain)[len(*blockchain)-1]
c.blockchainMutex.Unlock()

lastBlockTime, err =
↪  time.Parse(time.RFC1123,
↪  lastBlock.Header.Timestamp)
if err != nil {
        log.Fatal(err)
}

if nextTBlockTime.Before
(lastBlockTime.Add(
time.Second * time.Duration
(c.TBlock))) {
c.random.Seed
(lastBlock.Header.Seed)
nextTBlockTime = lastBlockTime
}

if nextTChainTime.Before
(lastBlockTime.Add(time.Second *
time.Duration(c.TChain))) {
nextTChainTime = lastBlockTime
}
```

```
go func() {
c.blockchainMutex.Lock()
if c.electedValidator ==
↪  (*c.Engine.BasicHost).ID() {
candidateBlock :=block.
CreateBlockWithList
        (&lastBlock, buffer,
        c.Engine.HostNode.
        LiveWallet, (*c.Engine.
                BasicHost).ID(),
                ↪  time.Now().
                UTC().UnixNano())
                currentChain := c.Engine.
                HostNode.Blockchain

        if (time.Now()
        .After(nextTChainTime) ||
        ↪  time.Now().
        Equal(nextTChainTime)) &&
        len(*currentChain) >=
        ↪  c.BlockThreshold
        && len(lastBlock.
        TransactionList) > 1 {

        if c.electedValidator ==
        ↪  (*c.Engine.BasicHost).ID() {
aggregatedChain :=
↪  c.aggregateChain(append(*currentChain,
↪  *candidateBlock))

if aggregatedChain != nil {
c.Engine.broadcastBlockchain
(aggregatedChain)
}
}
nextTChainTime =
↪  nextTChainTime.Add(time.Second *
↪  time.Duration(c.TChain))

} else {
if block.IsBlockValid(candidateBlock,
↪  &lastBlock) {
c.Engine.broadcastBlockchain(
append(*currentChain, *candidateBlock))
} else {
if c.Engine.Verbose {
```

```
                                fmt.Println("Block Creation: Block
                                ↪  validation failed!")
                                }
                                }
                                }
                                }
                                c.blockchainMutex.Unlock()
                                }()

                                nextTBlockTime =
                                ↪  nextTBlockTime.Add(time.Second *
                                ↪  time.Duration(c.TBlock))
                        }
                        time.Sleep(time.Second)
                }
        }()
}
```

```
func (c *Consensus) ReceiveUpdatedChain(payload message.BlockchainPayload)
↪  {
        c.blockchainMutex.Lock()
        currentChain := c.Engine.HostNode.Blockchain
        updatedChain := payload.Blockchain

        if ((len(updatedChain) > len(*currentChain) &&
        ↪  updatedChain[0].Header.Generation ==
        ↪  (*currentChain)[0].Header.Generation) ||
                updatedChain[0].Header.Generation >
                ↪  (*currentChain)[0].Header.Generation) &&
                updatedChain[len(updatedChain)-1].Header.Validator ==
                ↪  c.electedValidator.Pretty() {

                if len(updatedChain) == 1 ||
                ↪  block.IsBlockValid(&updatedChain[len(updatedChain)-1],
                ↪  &updatedChain[len(updatedChain)-2]) {
                        *c.Engine.HostNode.Blockchain = updatedChain
                        currentChain = c.Engine.HostNode.Blockchain
                        lastBlock := (*currentChain)[len(*currentChain)-1]
        c.Engine.HostNode.LiveWallet = make(map[string]int)
                        for key, value := range lastBlock.Header.Wallet {
                                c.Engine.HostNode.LiveWallet[key] = value
                        }
                        bytes, err :=
                        ↪  json.MarshalIndent(*c.Engine.HostNode.Blockchain,
                        ↪  "", "  ")
                        if err != nil {
```

```
                                log.Fatal(err)
                        }
                        if c.Engine.Verbose {
                                fmt.Printf("\x1b[32m%s\x1b[0m\n",
                                ↪  string(bytes))
                        }
                } else {
                        if c.Engine.Verbose {
                                fmt.Println("ReceiveUpdatedChain: Block
                                ↪  validation failed!")
                        }
                }
        }
        c.blockchainMutex.Unlock()
}
```

Chain Aggregation

The `aggregateChain()` functions takes a blockchain as input and returns an aggregated block which is the genesis block of a new blockchain. The aggregation is performed using `weightedMovingAverage()` functions which implements the weighted moving average of the sensing data, stored in the blockchain.

```
func (c *Consensus) aggregateChain(blockChain []block.Block) []block.Block
↪  {
        tempList := blockChain[len(blockChain)-1].TransactionList
        newBlockChain := []block.Block{}

        if len(tempList) > 1 {
                txsList := []string{}

                for _, c := range tempList {
                        txsList = append(txsList, c.Transaction)
                }

                aggregate :=
                ↪  weightedMovingAverage(decodeTxsToFloat(txsList))

                newGenesisBlock := block.CreateGenesisBlock(
                blockChain[0].Header.Generation+1,
                ↪  &blockChain[len(blockChain)-1],
```

```
                        strconv.FormatFloat(aggregate, 'f', 6, 64),
                        ↪  blockChain[len(blockChain)-1].Header.Wallet,
                        (*c.Engine.BasicHost).ID(),
                        ↪  time.Now().UTC().UnixNano())
                newBlockChain = append(newBlockChain, *newGenesisBlock)

                if c.Engine.Verbose {
                        fmt.Println("Blockchain aggregated.")
                }
        } else {
                newBlockChain = nil
        }

        return newBlockChain
}
```

```
func weightedMovingAverage(txsList []float64) float64 {
        alpha := 0.1
        wAvg := 0.0

        if len(txsList) == 1 {
                return txsList[0]
        }

        for i := range txsList {
                wAvg = (1.0-alpha)*wAvg + alpha*float64(txsList[i])
        }

        return wAvg
}
```

Engine

`Engine` manages the host node and its communication over the network. It contains a PeerList, which is a `TimedPeerList` struct type that stores a node's peer list alongside its update time. It assigns the current clusterID, Moniker, and address to a node. The function `CreateEngine()` creates a new `Engine` thereby running a new node. The `Engine` handles the transaction listening, transaction and blockchain broadcasting, and peer list updating through the `startListening()`,

`broadcastTransaction()`, `broadcastBlockchain()`, and `updatePeerList()` functions, respectively.

```
type Engine struct {
        HostNode      *node.Node
        BasicHost     *host.Host
        PubSub        *libp2pPubSub
        Moniker       string
        Address       string
        ClusterID     int
        PeerList      *TimedPeerList
        NodeConsensus *Consensus
        Verbose       bool
        shutdown bool
}
type TimedPeerList struct {
        Timestamp time.Time
        Peers     map[string]string
}
```

```
func CreateEngine(moniker string, clusterID int, listenPort int) *Engine {
        pubsub := new(libp2pPubSub)
        host := pubsub.createPeer(moniker, listenPort)
        pubsub.initializePubSub(*host)
        engine := Engine{
                HostNode:      node.CreateNode(),
                BasicHost:     host,
                PubSub:        pubsub,
                Moniker:       moniker,
                Address:       getLocalHostAddress(*host),
                ClusterID:     clusterID,
                PeerList:      &TimedPeerList{time.Now(),
                ↪  make(map[string]string)},
                NodeConsensus: nil,
                Verbose:       false,
                shutdown: false,
        }

        engine.NodeConsensus = NewConsensus(&engine, tBlock, tChain,
        ↪  blockThreshold)
        return &engine
}
```

```
func (e *Engine) startListening() {
        e.NodeConsensus.RunConsensus()

        for !e.shutdown {
                sender, message := e.PubSub.Receive()

                if e.Verbose {
                        fmt.Println("\nIncoming broadcast...")

                        data := &msg.Message{}
                        if err := json.Unmarshal([]byte(message), &data);
                        ↪  err != nil {
                                log.Fatal(err)
                        }

                        fmt.Printf("Node %s sent Message of type: '%s'\n",
                        ↪  sender.Pretty(), data.Type)

                        if data.Type == msg.TransactionType {
                                fmt.Println(message)
                        }
                }

                e.handleMessage(message)
        }

        e.NodeConsensus.StopConsensus()

        err := (*e.BasicHost).Close()
        if err != nil {
                log.Println(err)
        }
}
```

```
func (e *Engine) broadcastTransaction(to peer.ID, timestamp string, str
↪  string) {
        from := (*e.BasicHost).ID()

        rawData, err := json.Marshal(msg.TransactionPayload{From: from,
        ↪  To: to, Timestamp: timestamp, Transaction: str})
        if err != nil {
                log.Fatal(err)
        }
```

```
        message := msg.Message{Type: msg.TransactionType, RawPayload:
        ↪  rawData}

        bytes, err := json.Marshal(message)
        if err != nil {
                log.Println(err)
        }

        e.PubSub.Broadcast(string(bytes))
}
```

```
func (e *Engine) broadcastBlockchain(blockchain []block.Block) {
        rawData, err := json.Marshal(msg.BlockchainPayload{Blockchain:
        ↪  blockchain})
        if err != nil {
                log.Fatal(err)
        }

        message := msg.Message{Type: msg.BlockchainType, RawPayload:
        ↪  rawData}

        bytes, err := json.Marshal(message)
        if err != nil {
                log.Println(err)
        }

        e.PubSub.Broadcast(string(bytes))
}
```

```
func (e *Engine) updatePeerList(peerList map[string]string) {
        e.PeerList.Timestamp = time.Now()
        e.PeerList.Peers = peerList

        for key := range e.PeerList.Peers {
                if _, found := e.HostNode.LiveWallet[key]; !found {
                        e.HostNode.LiveWallet[key] = 0
                }
        }
}
```

5.6.2 Running the Demonstration

To run the demonstration, we created a `RunNodeManager()` function which initializes the node creation and management process. In this function, to simulate the different regions, nodes are created as cluster and each cluster is considered as a different spatial region. The function defines the both number of clusters and number of nodes per cluster. Each cluster is created from struct `Cluster` with an unique ID. The function `setupNetworkTopology()` sets up a random and sparse network topology. Finally, the function `main()` runs the platform.

```
var (
        clusterCount    = 4
        nodesPerCluster = 5
        initialPort     = 10000
        clusters        = make([]Cluster, clusterCount)
        txInterval  = 5000
        txPerMinute = (txInterval / 1000) * 60
        txLoad      = 0.2
        maxInt = 1000
        minInt = 0
        inputRangeMean = (float64(maxInt-minInt) / 2.0) + float64(minInt)
)
type Cluster struct {
        clusterID int
        engines   []*Engine
}
func RunNodeManager() {
        counter := 0
        for i := range clusters {
                clusters[i] = Cluster{clusterID: i, engines: []*Engine{}}
                clusters[i].engines = make([]*Engine, nodesPerCluster)
        }
        for _, cluster := range clusters {
                engines := cluster.engines
                defer shutdownEngines(engines)
                for i := range engines {
                        engines[i] = CreateEngine(fmt.Sprintf("node %d",
                        ↪  counter), cluster.clusterID,
                        ↪  initialPort+counter)
                        counter++
                }
```

```
                setupNetworkTopology(engines)
                setupPeerLists(engines)
                startListening(engines)
        }
        runTerminalInterface()
}
func setupNetworkTopology(engineSlice []*Engine) {
        mrand.Seed(time.Now().UTC().UnixNano())
        if len(engineSlice) > 1 {
                edges := "Graph Topology: {"
                for i := range engineSlice {
                        var n = i
                        for n == i || (len(engineSlice) > 2 &&
                        ↪  len((*engineSlice[i].BasicHost).Network().
        ConnsToPeer((*engineSlice[n].BasicHost).ID())) != 0) {
                                n = mrand.Intn(len(engineSlice))
                        }
        connectHostToPeer(*engineSlice[i].BasicHost,
        ↪  getLocalHostAddress(*engineSlice[n].BasicHost))
                        edges += fmt.Sprintf("(%d, %d), ", i, n)
                }
                edges = strings.TrimSuffix(edges, ", ")
                edges += "}"

                fmt.Println(edges)
        }
        time.Sleep(time.Second * 2)
}
```

```
func main() {
        engine.RunNodeManager()
}
```

Figures 5.10 to 5.14 present some screenshots of the demonstration. In the demonstration, the following commands are implemented: `SwitchCluster`, `Send`, `SwitchNode`, `AutoSend`, `ManualSend`, `ShowActivity`, `PrintBlockchain`, `Shutdown`. Table 5.3 lists the description of the commands.

Table 5.3: Table of Commands

Command	Description
`SwitchCluster`	Shows a different Cluster
`Send`	Sends transaction with manual input to transaction value and receiver
`AutoSend`	Automatic transaction creation
`ManualSend`	Resets transaction sending to `Send` from `AutoSend`
`ShowActivity`	Visualizes the activities in the network
`PrintBlockchain`	Prints the content of a blockchain
`Shutdown`	Shutdowns the demonstration

Network Initialization

The Unix command `go run src/main/sensor-chain.go` creates the nodes and clusters (in Figure 5.10, it creates 4 clusters, each with 5 nodes). In each cluster, the connections between the nodes are set in random.

Genesis Block in a Newly Created Blockchain

Figure 5.11 depicts the content of a genesis block of a newly created blockchain for `cluster` 0. The field `Generation` is 0, as no aggregation has been done yet in the network. Similarly, the `wallet` and `TransactionList` are both empty.

Broadcasts and transactions in Blockchain

By using the `ShowActivity` command, figure 5.12 displays the detail of the new transactions happened in a network. On the other hand, the figure 5.13 shows the committed transactions in a block.

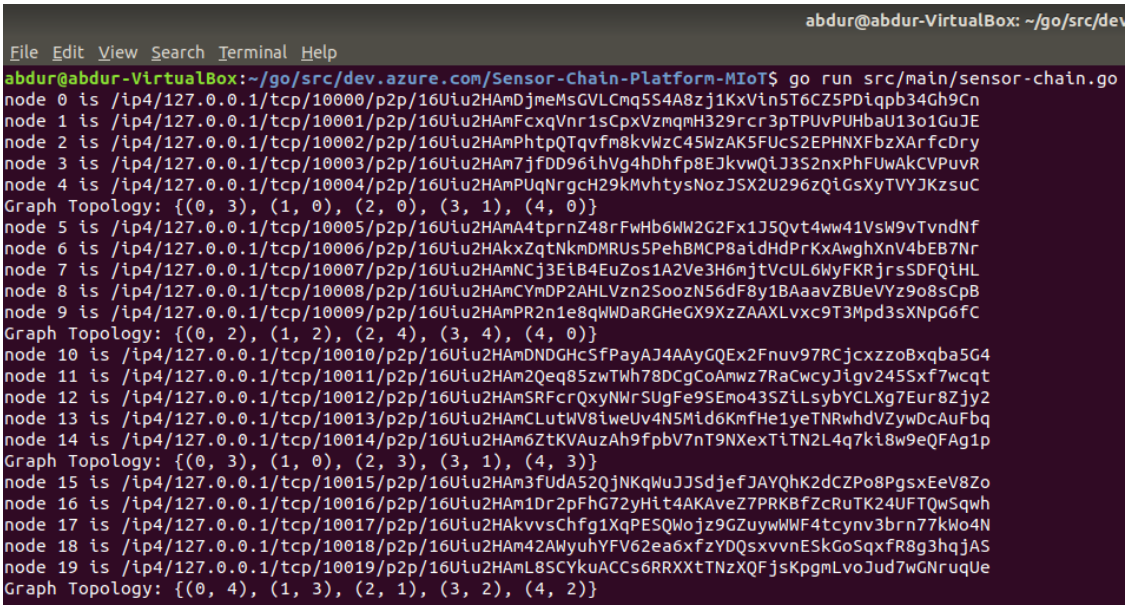


Figure 5.10: Network initialization in the demonstration.

Figure 5.11: Genesis block in a newly created blockchain.

Figure 5.12: Transaction broadcasting in the demonstration.

Genesis Block After Aggregation

Figure 5.14 shows the content of the genesis blockchain after an aggregation. The field `Generation` is set 2, meaning there two aggregation happened on a blockchain so far. The `Wallet` contains the number of transactions made by the different nodes in a cluster. In this version of genesis block, The `Transaction` field in `TransactionList` is not empty. Rather, it contains the weighted average value of the blockchain.

5.7 Discussion and Summary

In this chapter, we proposed "Sensor-Chain", a lightweight scalable blockchain framework for resource-constrained IoT sensor devices. In this framework, a conventional blockchain is made lightweight in three steps. First, a global blockchain is divided into smaller disjoint local blockchains in spatial domain such that the

Figure 5.13: Transactions recorded in a blockchain.

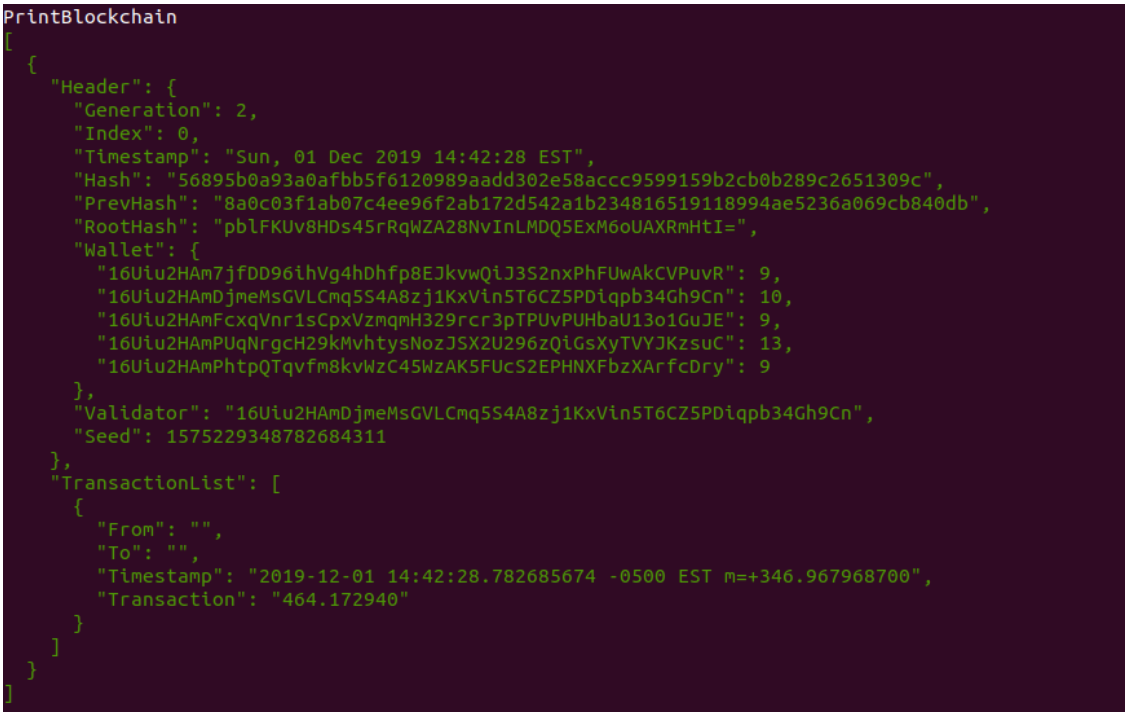


Figure 5.14: Genesis block after aggregation.

required storage space to hold a local blockchain for an IoT device is always smaller than that in conventional blockchain. Second, a temporal constraint is imposed on the life span of the local blockchains to limit their size in temporal domain. Finally, a sensor node is required to keep at most one local blockchain in its memory at any time instance. We analyzed and tested Sensor-Chain in terms of both long-run performance and scalability; and compared with other approaches. Experimental results show that it consumes far little storage space than other approaches.

CHAPTER 6

QUANTIFYING TRAJECTORY PRIVACY IN BLOCKCHAIN-BASED INTERNET OF THINGS (IOT)

In this chapter, we study the trajectory privacy issue in a permissioned blockchain. In such a blockchain, the authority of the system has control over the identities of its users. Such information can allow an authority to map identities with their spatiotemporal data, undermining the location privacy of a mobile user. We study the problem under three conditions. First, the authority holds the public and private key distribution task in the system. Second, there exists a spatiotemporal correlation between consecutive location-based transactions. Third, users communicate with each other through short-range communication technologies such that it constitutes a proof of location (PoL) on their actual locations. We show that, in a permissioned blockchain with an authority and a presence of a PoL, existing approaches cannot be applied using a plug-and-play approach to protect location privacy. In this context, we propose BlockPriv, an obfuscation technique that quantifies the relationship between privacy and utility in order to dynamically protect the location privacy in the permissioned blockchain. The chapter is organized as follows. Sections 6.1 and 6.2 introduce the problem statement and hypothesis. The system model and attack strategies are discussed in 6.3. The BlockPriv approach is detailed in section 6.4 and its several important security, privacy, and utility aspects are analyzed in section 6.5. Experimental evaluation with real-world data of the proposed approach is covered in section 6.6.

6.1 Introduction

To date, two main categories of blockchain have been studied in a variety of IoT applications: public and permissioned. In the public version, the blockchain has

no authority; a node can join and leave the network at any point with random pseudonyms and can also change its public keys at any time instance (e.g., for every transaction). This approach was first proposed by Nakamoto [Nak08], the creator of blockchain and later widely adapted for almost all kinds of applications, including smart home and vehicular network[ZN+15, DSKJ17, SK17]. It is considered that, the certificate authority (CA) and the public blockchain are independent of each other. Thus, the frequent pseudonym scheme makes the IoT nodes untraceable and provides high privacy. In fact, in order to undermine trajectory privacy of a target node, other nodes need to either collude with a large number of nodes over a large region or track the target node physical, which is really hard to do, if not impossible, in mobile environment.

However, in a permissioned blockchain, such a high level of privacy is not easily attainable, as the authority of the blockchain controls the blockchain network with a variety of access controls spanning from control over joining the network to perform consensus mechanisms. Amazon's Quantum Ledger Database (QLDB)[ama], J.P. Morgan's Quorum blockchain [Mor19], and Microsoft's Azure blockchain [Azu] are just a few examples of industry standard permissioned blockchains. Similar to many other fields, permissioned blockchain is also being studied in the IoT of mobile devices. The authority of the blockchain holds the public and privacy key distribution task in the system which gives it an upper hand of tracking the the mobile devices. The problem can be exacerbated if the communication between the nodes are based on short-range communication. Such communication can be used to localize a target node in spatiotemporal domain by generating proof-of-location (PoL) [ABMZ18] on the node's locations. In this context, We first discuss the limitations of existing location privacy-preserving mechanisms under a PoL in the context of permissioned blockchain. Then, we present an effective trajectory privacy-preserving approach,

coined as BlockPriv and quantify the trade-off between privacy and utility theoretically and empirically using two factual datasets.

6.2 Motivation and Problem Statement

In this work, we study the location privacy issue in the context of permissioned blockchain, where: 1. the authority of the blockchain holds the public and privacy key distribution task in the system, 2. a transaction can be considered as a proof of location (PoL) for a user's temporal whereabouts, and 3. there is a spatiotemporal correlation between the locations. For better understanding, we draw motivation of a permissioned blockchain from CreditCoin, a privacy-preserving blockchain framework for the Vehicular Ad Hoc Network (VANET) [LLC+18]. In this framework, the vehicles are required to be registered with the authority. This authority is responsible for generating and providing the vehicles with cryptographic keys, and keep track of the relationship between the vehicles and the provided keys. A set of trace managers at different locations also aids the authority in tracking malicious vehicles/users. In this framework, only road-side units (RSUs) and authorized vehicles are responsible for managing the blockchain. This framework is built around the short-range communication technology-based P2P network of the vehicles. Here, the vehicles make transactions with their peers such that each transaction is signed by each of the peer vehicles by their public keys. As these transactions are made through a short-range communication technology (e.g., Wi-Fi, Bluetooth), they can be treated as a proof-of-location (PoL) for the vehicles' whereabouts in the spatiotemporal domain. In some frameworks, such as the one proposed in [ABMZ18], the proof of location is explicitly defined in the design. Based on the transaction information, the vehicles generate a rating about each other and forward them to the

nearest RSU. The RSUs then compute the overall rating of each vehicle and append the new rating into the blockchain. Similar motivation can also be drawn from the work presented in [YYL+18]. Obviously, these frameworks can be integrated into many other mobility-centric IoT scenarios, such as mobile crowdsensing. The RSUs and smart vehicles can be replaced with Wi-Fi access points and low powered mobile devices (e.g., smart phones and smart watches), respectively. In terms of location privacy, these frameworks only guarantee conditional privacy to IoT users. That is, the devices can enjoy privacy from their peers by using the public keys provided by the authority. However, as the authority holds the mapping between real identity and the public keys, the privacy of sensitive locations from a malicious authority cannot be preserved using only a key changing mechanism. A malicious authority can perform a spatiotemporal analysis of the disclosed locations of a user and can reveal sensitive information. Against this backdrop, we seek answer to the following problem: *How can a user's privacy preference for different locations and spatiotemporal mobility information over a map will help an obfuscation approach to prevent a malicious blockchain authority from revealing trajectory of sensitive locations?*

We hypothesize that a combination of spatiotemporal silence with spatial obfuscation approach will allow to protect trajectory privacy of mobile users from a malicious authority under PoL.

6.3 Background

In this section, we present the details of the system model and the behavior and attack strategies of the malicious entities in the system. We then formulate the problem and state the goals we set out to achieve in the design of its solution.

Table 6.1: Table of Symbols

Symbol	Description
MU	Mobile user or mobile node
$\mathcal{N}_x$	Privacy parameter for a location l_x
$\mathcal{P}(l_h)$	Privacy level achieved for location l_h
$Pr^t_{MU}(l)$	MU's probability of being at location l at time t
l^s	A sensitive location
S	Set of all sensitive locations of a MU
$\mathcal{U}(l)$	Loss of utility for location l
T_r	A trajectory
n	Total number of sensitive locations in a T_r
δt	Time difference
$\mathcal{L}_a$	Set of all locations reachable to/from location l_a
$\Phi(a, b)$	Required time to reach from location l_a to l_b
$\|X\|$	Size/number of elements in a set X
α	% of location types selected as sensitive
r	Privacy region radius

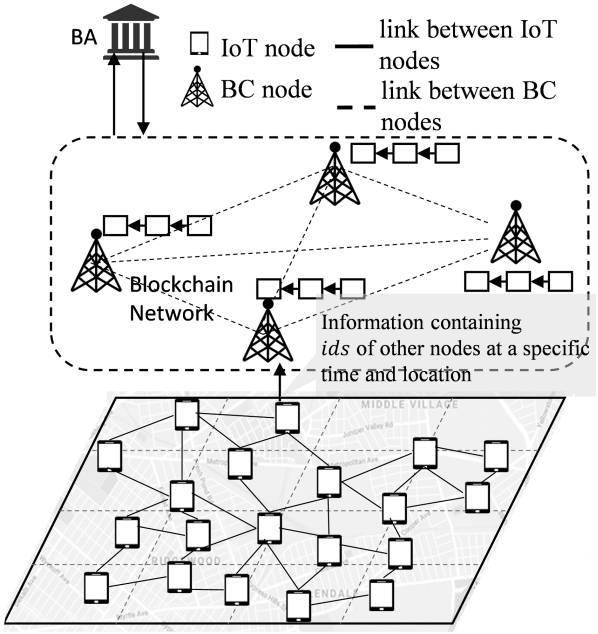


Figure 6.1: System model of permissioned-blockchain where BC and BA refer to blockchain and blockchain authority, respectively. The BA also acts as certificate authority and trace manager. The mobile IoT nodes are connected with each other in a P2P network using a short-range communication technology. They make transactions with each other and send information on the transactions (e.g., rating about other mobile nodes at a specific location and time) to the nearest blockchain node. Here, each grid refers to a specific location.

6.3.1 Blockchain System Model

We consider a permissioned blockchain, where its authority also acts as the certificate authority to provide the public and private key pairs to the mobile nodes. The mobile nodes are registered with the system and communicate with each other using the preassigned key pairs. Communication between the nodes takes place using a short-rage communication technology. The nodes can request the authority for new key pairs at any point of time. The blockchain is managed by preassigned mobile edge computing devices (e.g., RSU, Wi-Fi access points, and so on), distributed over a large region. These devices constitute the blockchain nodes and are connected with each other in a P2P network over the internet. The transactions among the IoT nodes are broadcasted to the blockchain nodes in the blockchain network. The blockchain nodes aggregate and insert the new transactions into the blockchain through a consensus mechanism (e.g. practical byzantine fault tolerance, proof-of-stake) in a timely fashion (e.g. every 30 minutes). We consider a blockchain architecture similar to the one presented in CreditCoin [LLC+18] without considering the rewarding phase. We assume that the mobile nodes have internet capability to compute the time to reach one location from another with the help of a traffic information provider in real time, e.g., Google Maps. We also assume that the information between the traffic information service provider and a node is anonymous and the provider is independent from the blockchain authority.

6.3.2 Attack Model

Malicious Entities

In the system, we consider the authority of the blockchain as the malicious entity. It follows the honest-but-curious adversary model in the system. That is, it tries

to predict a target node's sensitive spatiotemporal information without violating any protocol of the system or dismantling the way blockchain works. Furthermore, it is not going to hack into the device of a target node. We also consider that, in order to compute the time reachability information, the authority also uses a traffic information service provider. From this point on, we refer the authority as an attacker. It is important to note that some of the mobile nodes can be malicious. However, as we mentioned earlier in the system model, the mobile nodes can change their public keys at any point of time; the malicious mobile nodes cannot track a target node from their transactions without colluding with the authority. This is a fundamental privacy feature of blockchains. Thus, we focus on the attack strategies of the blockchain authority.

Attacker's Goal and Strategies

The goal of an attacker is to understand a mobile node's presence at different locations in the temporal domain. In order to do so, it utilizes the time-reachability-based spatiotemporal correlation between a node's disclosed locations in the blockchain as its fundamental strategy. Let the random variable O^t_{MU} represents the actual location of a mobile node MU at time t. Given a node's locations l_i, l_j at time t_a, t_b respectively, the node's probability of being at a location l_h at a discrete time t_q $(t_a < t_q < t_b)$ is

$$Pr^q_{MU}(l_h) = \mathbb{P}\mathrm{r}(O^q_{MU} = l_h | O^a_{MU} = l_i, O^b_{MU} = l_j) \tag{6.1}$$

The attacker computes $Pr^{q}_{MU}(l_h)$ using the time reachability correlation as follows.

$$Pr^{q}_{MU}(l_h) = \begin{cases} 1 & \text{If } l_h \text{ is reachable to and from } l_i \\ & \text{and } l_j \text{ in } (t_b - t_a) \text{ time} \\ 0 & \text{Otherwise} \end{cases} \tag{6.2}$$

Obviously, it is possible to have multiple locations with $Pr^{q}_{MU}(l_h) = 1$. Thus, the ultimate goal of the attacker is to minimize the number of such locations. That is,

$$\text{minimize} \left(\sum Pr^{q}_{MU}(l_h) \right) \tag{6.3}$$

This forms the core of an attacker's strategy. Based on this, we consider mainly the following attacks that can be exploited by the attacker to infer a target node's location information.

(1) *Collusion with malicious mobile nodes*: Malicious nodes collude with the authority and provide it with the location information of a target node for profit.

(2) *Map matching attack*: The authority employs the map information to understand spatially reachable and unreachable location information. A spatially unreachable location refers to a location that cannot be reached at any time using a map service (e.g., the middle of a lake). Thus, $Pr^{\infty}_{MU}(l) = 0$.

(3) *Time-reachability-based path reconstruction attack*: In order to reconstruct the actual path between two revealed locations, the authority can use the time reachability information to construct the valid paths that can be traveled between the two locations within a time limit.

We also analyze the impact of transaction dropping attack on location privacy. Note that, the scope of this work encompasses the analysis of location privacy invading attacks from a user's point of view and thus different blockchain related attacks, such as DDoS, Sybil, 51% attack, and eclipse attack are not covered here.

6.3.3 Problem Formulation and Design Goals

It is clear that there is an important trade-off between location privacy and utilization of the system. The problem lies with the short-range communication technology-based transactions between the mobile nodes that form proof of locations (PoL) for the nodes. Thus, in order to protect a sensitive location's privacy, a mobile node must remain silent in the network: that is, it must not make any transaction in the network. This leads to the question of how long in both spatial and temporal domains a node must remain silent to protect a sensitive location's privacy. Remaining silent infinitely results in a location privacy of 100%, but a system utilization of 0%. In other words, an indefinite silence will incur a 100% *loss of utility*. Hence, the goal of this work is to formulate, design, implement, and evaluate a location privacy-preserving mechanism, called BlockPriv, for mobile nodes in the context of permissioned blockchain by solving the following problem:

$$\text{minimize } \{\mathcal{P}^{-1}(l^s), \mathcal{U}(l^s)\} \tag{6.4}$$

Here, $\mathcal{P}(l^s)$ and $\mathcal{U}(l^s)$ refer to the achieved privacy for sensitive location l^s and the loss of utility due to privacy preservation for l^s, respectively.

To summarize, in the design of the BlockPriv mechanism, we intend to achieve the following goals: 1. achieve privacy without collaborating with any other entity in the system, and 2. achieve a quantifiable balance between privacy and utility.

6.4 The BlockPriv Approach

For the sake of clarity and to maintain coherence with the blockchain concept, we first discuss the public key changing technique adapted in BlockPriv. In our scheme, we adapt the temporal public key changing concept proposed by Michelin et al.

[MDS+18]. Here, at a fixed time interval t^{key}, a mobile node will change its public key in order to nullify the possibility of spatiotemporal linkage attack from malicious nodes. Note that, in our problem, public key changing can only provide privacy to a mobile node against its peers, not against the authority that distributes the keys. Also, this scheme is vulnerable against colluding attack between the authority and malicious mobile nodes, which is one of the focus of our work.

At this point, we present the formal definition of location privacy and utility from the perspective of a mobile node. The definition of privacy can be derived from the formulation of the attacker's objective, defined by equation 6.3, as follows.

$$\mathcal{P}(l^s) = \text{maximize} \left(\sum Pr^q_{MU}(l_h) \right) \tag{6.5}$$

Let us consider: a node's last revealed location in the blockchain is l_i at time t_a, and it was at a sensitive location l^s_h at time t_q. It should reveal its location, also known as making a transaction, at an insensitive location l_j at time t_b ($t_a < t_q < t_b$) if and only if

$$\mathcal{P}(l^s_h) = \left(\sum Pr^q_{MU}(l^s_h) \right) \geq \mathcal{N}_h \tag{6.6}$$

To explain, a node should reveal its location l_j at time t_b in the network to the authority when there exists at least $\mathcal{N}_q$ number of locations, including l^s_h, which are both reachable from and to l_i and l_j in ($\delta t = t_b - t_a$) time. Here, $\mathcal{N}_h$ is a user defined privacy parameter for location l^s_h. This formulation is applicable only for a single sensitive location. It is also possible that, after l^s_h, the node was also at another sensitive location l^s_p at time t_r ($t_a < t_q < t_r < t_b$) such that, after $\delta t = t_b - t_a$ time, $\mathcal{P}(l^s_h) \geq \mathcal{N}_h$, but $\mathcal{P}(l^s_p)) < \mathcal{N}_p$. In such a case, the node should not make any transaction at location l_j at time t_b. Formally, if there are m number of sensitive locations visited by a node between time t_a and t_b, then it will make a transaction

with its peers at an insensitive location at time t_b in the network if and only if

$$\mathcal{P}(l_i^s) = \left(\sum Pr_{MU}^{q_i}(l_i^s) \right) \geq \mathcal{N}_i; \quad \forall i = 1, \ldots m \tag{6.7}$$

Note that, from t_a to t_b, the node was continuously silent in the network. We call it single or 1 round silence to maintain privacy of the m number of sensitive locations. If a trajectory T_r contains n number of sensitive locations, then the average privacy of each sensitive location in that trajectory is defined as

$$\mathcal{P}(T_r) = \frac{1}{n} \sum_i \mathcal{P}(l_i^s), \quad i = 1, \ldots n \tag{6.8}$$

From the formulation of privacy, we can also define the loss of utility due to the application of privacy preservation. Let us consider: at i-th round silence, the node opted not to make any transaction at $\mathcal{P}(l_h^s)$ number of locations. In our definitions, this number is the loss of utility of BlockPriv. If a node maintained k rounds of silence to preserve privacy of a trajectory T_r with n number of sensitive locations, then the average loss of utility for each sensitive location is

$$\mathcal{U}(T_r) = \frac{1}{n} \sum_{i=1}^{i=k} \mathcal{U}_i \tag{6.9}$$

This allows us to reconstruct the multi-objective optimization problem, presented in equation 6.4, as a single objective optimization problem as follows:

$$\begin{aligned} &\text{minimize } \mathcal{U}(T_r) \\ s.t. \quad &\mathcal{P}(l_i^s) \geq \mathcal{N}_i; \forall l_i^s \in T_r \end{aligned} \tag{6.10}$$

Now we present in detail the mechanism of BlockPriv to solve this problem.

In this mechanism, the mobile nodes are responsible for labeling their sensitive locations and assigning level of privacy to each of them. The nodes utilize radius r to specify the level of privacy for a sensitive location as $\mathcal{N} = \pi r^2$. Let us consider: a node MU made a transaction in the network at time t_a at location l_i. Then, it

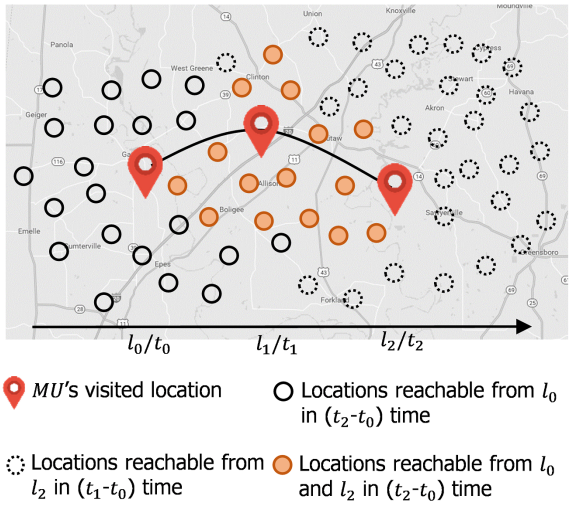


Figure 6.2: Illustrated BlockPriv: The curve refers to a mobile node (MU)'s actual path between l_0, l_1, and l_2 locations at times t_0, t_1, and t_2, respectively. The location l_1 is privacy sensitive for the MU. Thus, it remained silent at location l_1. It will make a blockchain transaction at l_2 at time t_2 only when the number of locations reachable from both l_0 and l_2 in $t_2 - t_0$ time, meets the privacy requirement for l_1.

moved to a privacy sensitive location l_h^s at time t_q and did not make any transactions. Then, after every Δt time at location l_j, different from both l_i and l_h^s, it checks the number of locations that are reachable to and from l_i and l_j. Let current time and location be t_b and l_j, respectively. The node first computes the set of all the locations $\mathcal{L}_i$ that are reachable from l_i in $\delta t = t_b - t_a$ time. Next, it computes the set of all the locations $\mathcal{L}_j$ from which location l_j is reachable. Then, $\mathcal{L} = \mathcal{L}_i \cap \mathcal{L}_j$ forms the set of all locations from which both l_i and l_j is reachable in δt time. In other words, each of the location in $\mathcal{L}$ creates a valid 1-hop route from l_i to l_j in δt time. That is, based on the time reachability information, the node can move from l_i to any location $l_l \in \mathcal{L}$ and then move to l_j in δt time. Thus,

$$\mathcal{L} = \{\forall l | (\Phi(l_i, l) + \Phi(l, l_j)) \leq \delta t\} \tag{6.11}$$

Here, $\Phi(a, b)$ refers to the time to get from location a to b. The size of $\mathcal{L}$ defines the privacy level achieved for sensitive location l_h^s in δt time. That is, $\mathcal{P}(l_h^s) = |\mathcal{L}|$. The

node will make a transaction at time t_j at location t_b only when $|\mathcal{L}| \geq \mathcal{N}_h$. If there is a total m number of sensitive locations visited by the node in δt time, according to equation 6.7, it will make a transaction at time t_j and location l_b if and only if

$$|\mathcal{L}| \geq \mathcal{N}_i; \quad \forall i = 1, \ldots m \tag{6.12}$$

It is understandable that, in the case when all the sensitive locations have the same level of privacy, comparing $\mathcal{L}$ with the level of privacy of the latest sensitive location is enough to check whether the condition in equation 6.7 is valid. However, for sensitive locations with different levels of privacy, the MU is required to check whether all the previous sensitive locations' levels of privacy are met before making any transaction.

For a single sensitive location l^s, the maximum loss of utility $\mathcal{U}_{max}(l^s)$ is bounded by the value of its privacy parameter $\mathcal{N}$. The higher the value of $\mathcal{N}$, the higher the $\mathcal{U}_{max}(l^s)$. More specifically, $\mathcal{U}_{max}(l^s) \leq \mathcal{L}$. Certainly, from equation 6.10, we do not want any "extra" loss in utility of the blockchain. Let t_a is the last time when a node's location was revealed in the blockchain. After that, at every Δt ($\Delta t \in \mathbb{Z}_{\geq 0}$) time, it computes $\mathcal{L}$ and checks whether it meets the privacy requirement of a set of sensitive locations. That is, after checking $\mathcal{L}$ at time $(t_a + x \times \Delta t)$, it will check $\mathcal{L}$ at time $(t_a + (x+1) \times \Delta t)$. Here, $x \in \mathbb{Z}_{\geq 0}$. Let, t', where $(t_a + x \times \Delta t) < t' < (t_a + (x+1) \times \Delta t)$, is the time when $\mathcal{L} \simeq \mathcal{N}$. Then, computing $\mathcal{L}$ at $(t_a + (x+1) \times \Delta t)$ time will certainly impose some extra loss of utilities. Thus, $\mathcal{U}_{max}(l^s) \leq \mathcal{N} + \mathcal{U}'$. Here, $\mathcal{U}'$ refers to the set of insensitive locations at which the MU opted not to make any transaction between time t' and $(t_a + (x+1) \times \Delta t)$. With the higher value of Δt, the value of $\mathcal{U}'$ will be higher. Thus, Δt should remain as small as possible. However, for resource-constrained mobile nodes, a very small Δt means very frequent computation of the time reachability, which affects the energy of the

Algorithm 6: BlockPriv

Input: Current location l_{cur}, current time t_{cur}, last revealed location in the blockchain l_{prev} and time t_{prev}, list of sensitive locations S, list of level of privacy for the sensitive locations $\mathcal{N}$, previous time of key change t^{key}_{prev}, key expiration time t^{key}

Output: Decision on making transactions.

```
if (t_cur − t^key_prev) ≥ t^key then
    Request new key pair from the authority.
    t^key_prev = t_cur
if l_cur is a sensitive location then
    Append l_cur to S and do not make any transaction.
else
    δt ← t_cur − t_prev
    L_prev ← select all the locations that are reachable from l_prev in δt time
    L_cur ← select all the locations from which l_cur is reachable in δt time
    L ← L_prev ∩ L_cur
    for (i = 1; i ≤ |S|; i++) do
        if |L| ≥ N(l^s_i ∈ S) then
            Delete l^s_i from S
    if S ≠ ∅ then
        Do not make any transactions in the network.
    else
        Free to make transactions.
```

device. Thus, the compromise between the capability of the device and loss of utility is an issue that needs to be examined: we leave it for our future work. The detail of BlockPriv is presented in Algorithm 6.

6.5 Scheme Analysis

In this section, we present an analysis of the important privacy, utility, and security aspects of BlockPriv.

6.5.1 Privacy Analysis

Privacy Bound

Lemma 6.5.1 *If there are multiple numbers of sensitive locations between two revealed insensitive locations, then each of the sensitive locations achieves a privacy level of* $(\max \mathcal{N})$.

Proof. Let us suppose that a mobile node MU has visited m number of sensitive locations between l_{prev} and l_{cur} in $\delta t = (t_{cur} - t_{prev})$ time. According to equation 6.12, it will make a transaction at location l_{cur} and time t_{cur} only when all of the sensitive locations' privacy requirements are met. That is, a new transaction will take place only when the length of the set $\mathcal{L} \geq (\max \mathcal{N} = \max\{\mathcal{N}_1, \ldots \mathcal{N}_m\})$. Thus, even if a sensitive location's privacy requirement is much lower than $(\max \mathcal{N})$, the achieved privacy for i-th sensitive location l_i^s in the set is $\mathcal{P}(l_i^s) = |\mathcal{L}| \geq (\max \mathcal{N})$. □

Obfuscating Paths

Lemma 6.5.2 *If there are any sensitive locations between two revealed insensitive locations l_i and l_j, then, at a minimum, there are* $(\max \mathcal{N})$ *number of* 1*-hop obfuscating paths between the two revealed locations.*

Proof. Equation 6.11 implies that each location in the set $\mathcal{L}$ is reachable to and from l_{priv} and l_{cur} in δt time. Thus, from the point of reachability, each i-th location in $\mathcal{L}$ forms a 1-hop path between l_{priv} and l_{cur} in δt time.

As a result, each path formed by each sensitive location $l_i^s \in \mathcal{L}$ is obfuscated with $(|\mathcal{L}| - 1)$ number of different other paths in δt time. □

6.5.2 Utility Analysis

Loss of Utility Bound.

Lemma 6.5.3 *If there are multiple numbers of sensitive locations between two revealed insensitive locations, then the maximum loss of utility* $\mathcal{U}_{max}(l^s)$ *in BlockPriv to preserve privacy of a sensitive location* l^s *is proportional to* $(\max \mathcal{N})$.

Proof. Lemma 6.5.1 states that whatever the expected level of privacy assigned to a specific sensitive location, the achieved privacy is bounded by the location with highest level of privacy $\max \mathcal{N}$. Thus, the maximum loss of utility for every sensitive location l^s between the two revealed insensitive locations is $\mathcal{U}_{max}(l^s) \leq (\max \mathcal{N}) + \mathcal{U}'$. □

6.5.3 Security Analysis

We analyze the efficacy of BlockPriv against different location privacy invading strategies by a malicious authority of the blockchain system list in subsection 6.3.2. We also briefly discuss the interesting impact of transaction dropping attack on location privacy.

Collusion Attack.

Definition 6.5.4 *A collusion with malicious mobile nodes is successful if the authority of the blockchain can find a new set of locations* $\mathcal{L}*$ *about a MU's sensitive location* l_i^s *such that*

$$|\mathcal{L} \cap \mathcal{L} * | < \mathcal{N}_i. \tag{6.13}$$

Lemma 6.5.5 *A combination of time reachability information and collusion with other malicious nodes will not leak privacy of a target mobile node.*

Proof. In BlockPriv, a mobile node remains silent in the spatial and temporal domains in order to preserve privacy against an untrusted authority of the blockchain. Thus, even if the authority colludes with some mobile nodes, it will not be able to construct a new set $\mathcal{L}*$ beyond $\mathcal{L}$ that would satisfy equation 6.13. In other words, its understanding about a targeted node's whereabouts will not be made any finer than $\mathcal{L}$ by colluding with other nodes. In fact, collusion with mobile nodes to track a target node is a costly approach. The target node changes its public keys frequently and to keep tracking it, the authority needs to update the colluding nodes at the same rate. The only way a colluding attack will be successful is if a malicious node physically tracks a target node. However, our work concentrates on providing security against software-based privacy invading techniques, not on physical observations. □

Map Matching Attack.

Definition 6.5.6 *For a sensitive location l^s, a map matching attack is considered to be successful if a attacker can find a set of locations $\mathcal{L}*$ from $\mathcal{L}$ such that, $(\mathcal{L}* \subset \mathcal{L}*)$, $(|\mathcal{L}*| > 0)$, and $Pr^{\infty}_{MU}(l_i) = 0; \forall l_i \in \mathcal{L}*$.*

Lemma 6.5.7 *BlockPriv is resilient against map matching attack.*

Proof. The mobile node calculates the time reachability information using a real-time map service provider and thus each location l, selected to form $\mathcal{L}$, is spatially reachable. That is, $\mathcal{L} = \{\forall l \in \mathcal{L} | Pr^{\infty}_{MU}(l) = 1\}$. Thus, $\mathcal{L}* = \emptyset$. □

Time Reachability-Based Path Reconstruction Attack.

Definition 6.5.8 *A time reachability-based path reconstruction attack on BlockPriv is said to be successful if, for a sensitive location l^s, the authority can find fewer than $\mathcal{N}$ number of paths between two revealed locations for a mobile node.*

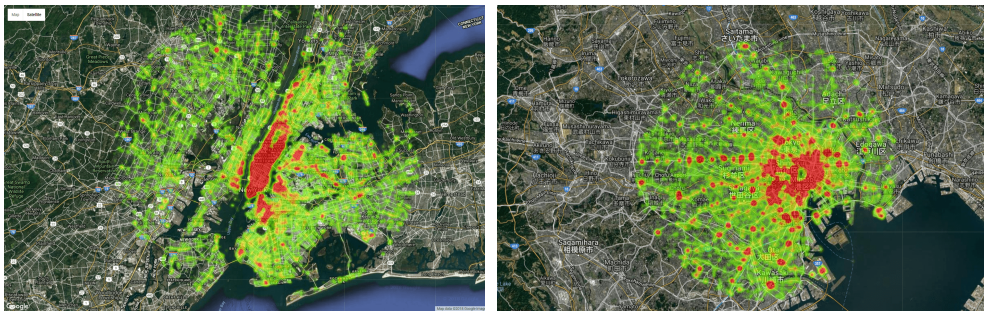
Lemma 6.5.9 *BlockPriv is resilient against time reachability-based path reconstruction attack.*

Proof. According to equation 6.11, every location $l_i \in \mathcal{L}$, including every sensitive location, is reachable from previously revealed location l_{prev} to l_{cur} in δt time. Thus, according to lemma 6.5.2, there are at least $\max \mathcal{N}$ number of 1-hop obfuscating paths from l_{prev} to l_{cur} for l_i.

We can now generalize the analysis for multi-hop paths. Let the actual path be: $l_{prev} \rightarrow l_1^s \rightarrow l_2^s \rightarrow l_{cur}$ and the temporal sequence of this path be: $t_{prev} \rightarrow t_1 \rightarrow t_2 \rightarrow t_{cur}$. Hence, $\delta t = \Phi(l_{prev}, l_1^s) + \Phi(l_1^s, l_2^s) + \Phi(l_2^s, l_{cur})$. Assume that, using BlockPriv, we got $\mathcal{L}$, where $\{l_1^s, l_2^s\} \in \mathcal{L}$. For the sake of argument, let us consider, for every location $l \in \mathcal{L}'$ ($\mathcal{L}' = \mathcal{L} \setminus \{l_1^s, l_2^s\}$), there exists no multi-hop path. In such a case, if somehow it is known that the node visited multiple locations between l_{prev} and l_{cur}, then the attacker can exclude all the single hop paths and is able to reconstruct the actual path: $l_{prev} \rightarrow l_1^s \rightarrow l_2^s \rightarrow l_{cur}$. However, in BlockPriv, the node remains silent in the network, such that every location in $\mathcal{L}$ exhibits similar probability of being the node's whereabouts under the time reachability condition. Also, such a special case can occur only when $Pr_{MU}^{\infty}(l) = 0; \quad \forall l \in \mathcal{L}'$. This case falls into the category of a map matching attack and lemma 6.5.7 proves that BlockPriv is resilient against such an attack. Hence, time reachability information cannot help a malicious authority to reconstruct the actual path. $\square$

Transaction Dropping Attack.

In this attack, a mobile node MU_i attempts to drop the transactions between itself and another node MU_j for a specific intention (e.g. preventing the other node from gaining reward out of ill intention or to protect its instance location privacy). There are two cases to consider here. First, MU_j passes the transaction information to



(a) New York City (NYC) (b) Tokyo (TKY)

Figure 6.3: Locations in (a) New York City (NYC) and (b) Tokyo (TKY) datasets. Green markers symbolize the locations. The red colors represent the high density regions.

the nearest blockchain node and thus MU_i's location information is revealed. In such a case, MU_i's attempt to protect location privacy will failed. Second, if MU_j also drop the transaction, then both the nodes' location information will remain undisclosed in the blockchain.

6.6 Experimental Evaluation

Table 6.2: Dataset Statistics

Dataset	#Transactions*	#Locations	#Types	#Nodes*
NYC	227428	38333	400	1083
TKY	573703	61858	385	2293

*Originally called "Checkins" and "Users". In this context, we renamed the variables "Transactions" and "Nodes", respectively.

In this section, we describe the details regarding the experimental evaluation of BlockPriv. To properly understand the efficiency and efficacy of our approach, we implemented two cases: locations with 1. similar privacy parameter and 2. different privacy parameters. These two versions will be referred to as **sim-BlockPriv** and **diff-BlockPriv**, respectively.

Table 6.3: Simulation Setup Parameters

Parameter	Value(s)
r	{500, 1000, 1500, 2000} meters
γ	{5,10,15,20}
v	30 miles per hour
α	{2, 4, 6, 8, 10}
n	100

6.6.1 Experimental Settings

Dataset Description

In this work, we consider the case of making frequent transactions in the network. Hence, we selected Foursquare's New York City (NYC) and Tokyo (TKY) datasets [YZZY15] to test the approach with factual data. These datasets contain the check-in information of nodes, in terms of location and time. The number of transactions, locations, location types, and nodes of the datasets are presented in Table 6.2 and a visualization of the locations in the datasets are depicted in Figure 6.3.

Simulation Setup

The datasets do not contain any mark on the privacy sensitive locations of the mobile nodes. Thus, we mark α% of the location types as sensitive locations for all the nodes. The different values of the parameters, including privacy level for a sensitive location r, used in the experiment, are shown in Table 6.3. For each combination of the parameters, we ran the simulation on both datasets for n number of nodes. As there is a correlation between the number of transactions and the impact of privacy on utility, we selected 100 nodes with the highest number of transactions. We justify this claim through comparing the result with 100 nodes with least number of transactions. Next, since the datasets do not contain continuous

Table 6.4: Pearson's Correlation Values

Dataset	Statistics	U-P	U-S
NYC	Minimum	0.75	0.44
	Average	0.94	0.92
	Maximum	1.00	0.99
TKY	Minimum	0.75	0.74
	Average	0.95	0.95
	Maximum	1.00	0.99

U-P: Loss of Utility vs. privacy level
U-S: Loss of Utility vs. sensitive location types

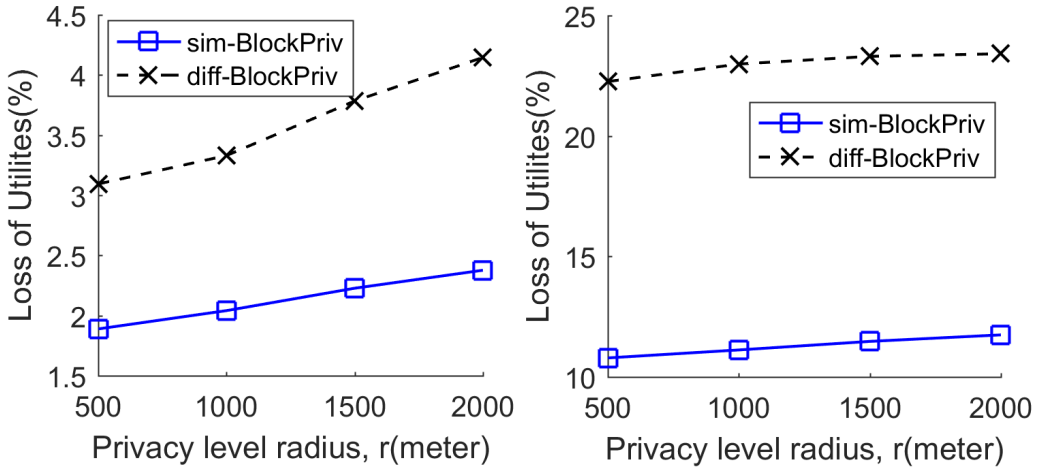
location information, we set a speed (v) for each node to simulate its reachability-based mobility. By nature of mobility, there are cases when a node cannot reach a new location, l_{new}, from a previous location, l_{prev} in a certain time, in the dataset with speed v. In these cases, we continue adding a small value to v (e.g. $v/5$) until it can reach l_{new}. In diff-BlockPriv, the difference in the privacy level for different sensitive locations is set by drawing a random number from the range $\{r - (r \times \gamma\%),\ r + (r \times \gamma\%)\}$.

6.6.2 Experiment Results

In the experiment, we examine the loss of utility of sim-BlockPriv and diff-BlockPriv. In particular, we examine the following two relationships, fundamental to the design of a privacy-preserving mechanism: 1. loss of utility versus privacy level, and 2. loss of utility versus number of sensitive locations.

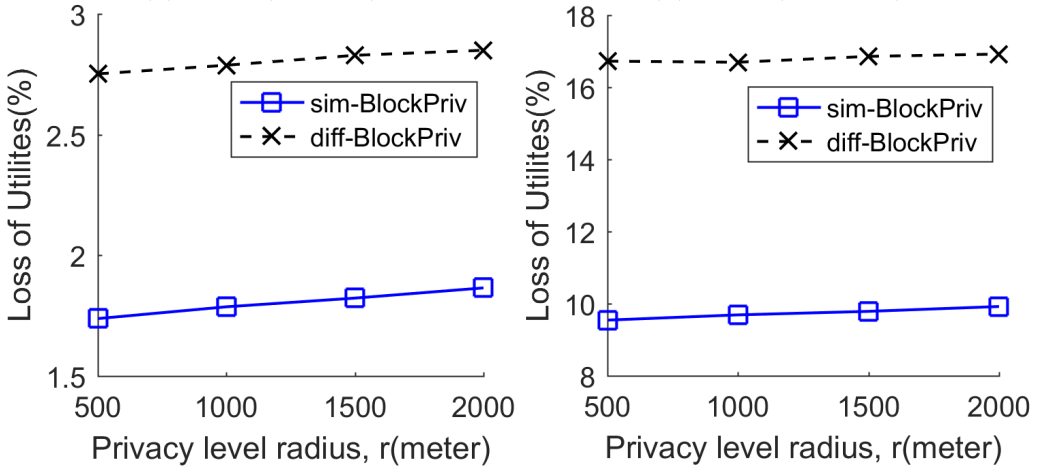
Utility versus Privacy Level

We first examine the relationship between the loss of utility and privacy (in term of radius r in meters). For example, Figures 6.4(a-d) visually show this relationship



(a) NYC ($\alpha = 2\%$)

(b) NYC ($\alpha = 10\%$)



(c) TKY ($\alpha = 2\%$)

(d) TKY ($\alpha = 10\%$)

Figure 6.4: Average loss of utilities versus privacy level in sim-BlockPriv and diff-BlockPriv.

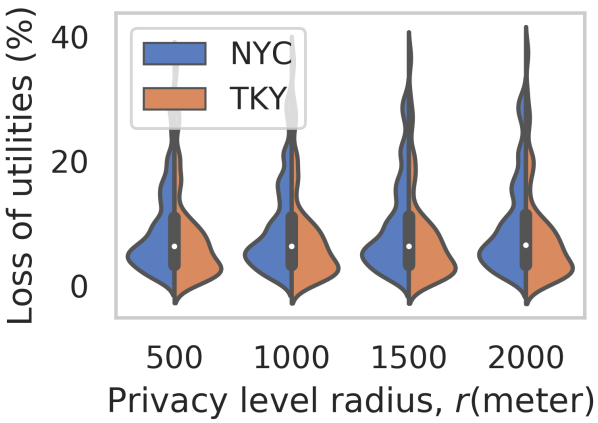
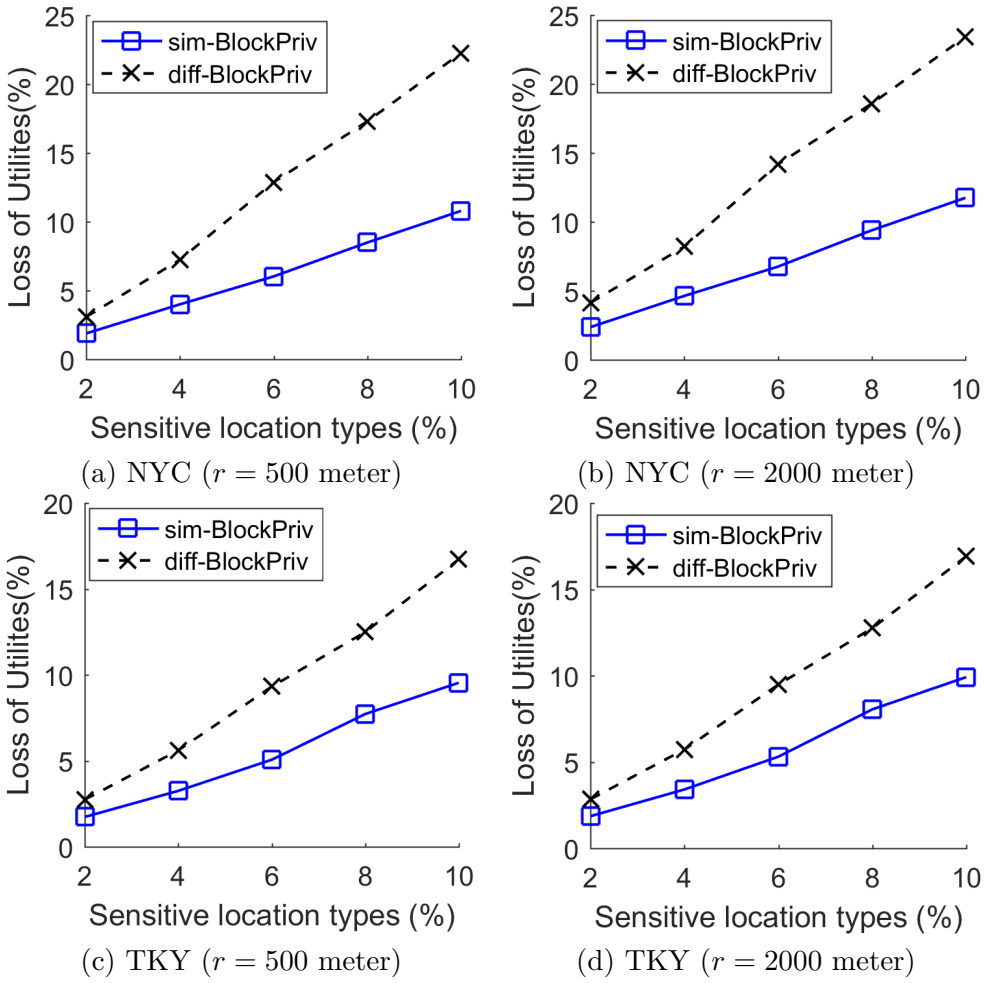


Figure 6.5: Distribution of loss of utilities in sim-BlockPriv, regarding different privacy levels.

for both sim-BlockPriv and diff-BlockPriv when there are a few number of sensitive locations($\alpha = 2\%$) and a significant number of sensitive locations($\alpha = 10\%$). Each data point in a figure refers to the average of the 100 users of a specific city. From these figures, we can make several important occlusions. First, we can draw a clear comparison between sim-BlockPriv and diff-BlockPriv, regarding the impact of privacy level r on the loss of utilities. From the city level view, for the same value of r, sim-BlockPriv imposes less utility loss than diff-BlockPriv due to the privacy level randomness associated in diff-BlockPriv.

Second, there is an almost linear correlation between the loss of utility and privacy level, regardless of the number of sensitive location types (α) in the dataset. We observe a similar upward trend of loss of utility against the increase in the privacy level for $\alpha = 2\%$ and $\alpha = 10\%$ in both of the datasets. The distribution of loss of utility in Figure 6.6.2 further improves the resolution of this linearity. If we look into the exact numeral values, presented in Table 6.4, the average Pearson's correlation values [BCHC09] are 0.94 and 0.95 for the NYC and TKY datasets, respectively. Such linear correlation and lower loss of utility give sim-BlockPriv an



(a) NYC ($r = 500$ meter) (b) NYC ($r = 2000$ meter)

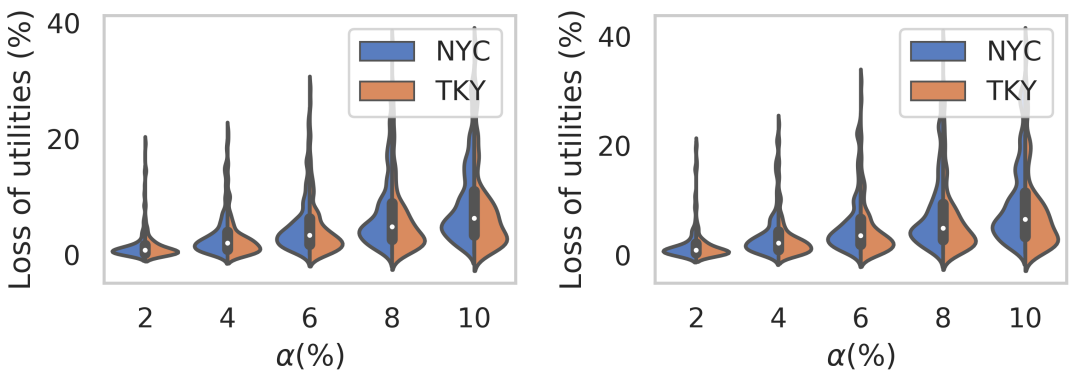
(c) TKY ($r = 500$ meter) (d) TKY ($r = 2000$ meter)

Figure 6.6: Average loss of utility versus number of sensitive location types (α) in sim-BlockPriv and diff-BlockPriv.

upper hand in designing a user-centric privacy scale, which we intend to explore in our extension of this work.

Utility versus Number of Sensitive Location Types

We then analyze the correlation between loss of utility and number of sensitive location types (α). While the analysis of the relationship between utility and privacy level show that the sim-BlockPriv charges less utility loss than diff-BlockPriv, the correlation between utility and number of sensitive location types further signifies



(a) (b)

Figure 6.7: sim-BlockPriv: comparison between the distribution of loss of utility for different numbers of sensitive location types (α) for r =(a) 500 meter, and (b) 2000 meter.

the superiority of sim-BlockPriv. Figures 6.6(a-d) present the average loss of utility for different values of α. We found that, regardless of the value of privacy level r, there is a linear correlation between utility and α. For the same value of r, the higher the value of the α, the higher the loss of utility. However, the increase of loss of utility is slightly sharper in diff-BlockPriv than in sim-BlockPriv. This sharpness is due to the effect of both the increase in the number of sensitive location types and the randomness in the privacy level. As we already know that sim-BlockPriv is better than diff-BlockPriv, we only present the distribution of loss of utility in sim-BlockPriv in Figure 6.7. For the same reason, we skipped the depiction of impact of different γ in diff-BlockPriv. Similar to the average values in Figure 6.6, the distributions of the loss of utility exhibit a linear correlation. More accurately, the average correlation is 0.92 and 0.95 in the NYC and TKY datasets, respectively. As we mentioned earlier, such a linear correlation can play important role to make BlockPriv usable for privacy-preserving applications.

User Level Correlation Analysis

Figure 6.6.2 depicts the correlation values for loss of utility versus privacy level (U-P) and loss of utility versus number of sensitive location types (U-S) for 100 users; Table 6.4 presents different statistics (min, average, and max) on these values. It is observed that in the NYC dataset, 75% of the nodes have 0.9 correlation for both U-P and U-S. In the case of the TKY dataset, these numbers are 82% and 84%, respectively. Note that, these statistics are generated by considering the 100 nodes with the greatest number of transactions in the datasets. We found that, when the number of transaction is fewer, the loss of utility is significantly less. For instance, in both datasets, the 100 nodes with fewest number of transactions achieved minimum 30% less loss of utility than the 100 nodes with highest number of transactions.

6.7 Conclusion

In this chapter, we introduce a user-centric obfuscation technique called BlockPriv, to preserve location privacy in permissioned blockchain-based IoT systems. We consider that a user cannot falsify its location and an untrusted authority can correlate locations by considering spatiotemporal constraints to predict unrevealed sensitive locations of a user. We quantify the relationship between the notion of privacy and utility of the system in BlockPriv. We analyze two variations of BlockPriv, sim-BlockPriv and diff-BlockPriv, where the first has the same privacy level for all the sensitive locations, and the second has a different privacy level for different sensitive locations. We show that there is a linear correlation between loss of utility and privacy level in sim-BlockPriv. Such linearity can be exploited to define a usable privacy scale.

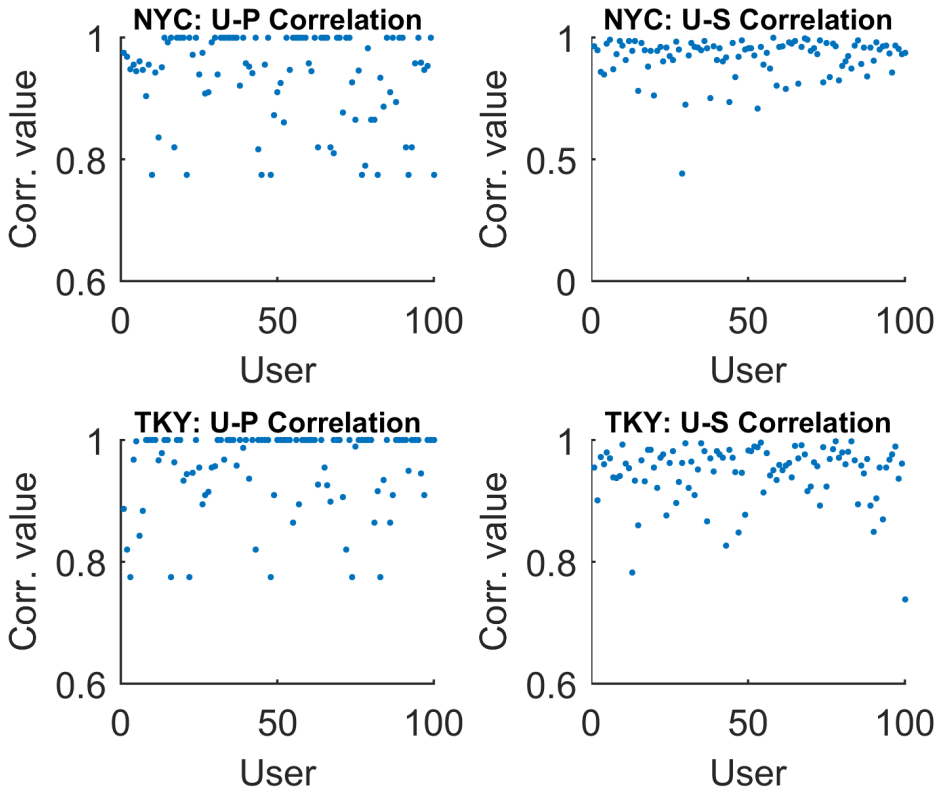


Figure 6.8: sim-BlockPriv: correlation values (Corr. value) of loss of utility versus privacy level (U-P) and loss of utility versus number of sensitive location types (U-S) for 100 users in NYC and TKY datasets.

CHAPTER 7

LIMITATIONS, FUTURE WORK, AND CONCLUSION

In this dissertation, we study the trajectory privacy preservation issue in both centralized and blockchain-based P2P decentralized network for location-based services (LBSs). In chapter 3, we propose a delay-aware long-term privacy preservation technique for frequently visited locations in query-based LBS. We then design a trajectory inference attack model by considering the spatiotemporal correlation of checkins, photos, and geo-tagged photos, test it on dummy-based obfuscation mechanisms and present its countermeasure in chapter 4. Afterward, in chapter 5, we introduce a framework for making blockchain lightweight for resource-constrained IoT devices towards achieving trajectory privacy in aggregation-based IoT systems, for instance, mobile crowdsensing applications. In chapter 6, we introduce a privacy-preserving obfuscation mechanism to protect against location inference attacks in permissioned-blockchain. In this chapter, we discuss the limitations of each of these works, future directions, and concluding remarks.

7.1 Limitations

7.1.1 Delay-Aware Long-Term Privacy Preservation for Frequently Visited Locations

We design a delay-aware long-term privacy-preserving obfuscation technique, coined as "KLAP" for frequently visited locations. KLAP considers a user's historical information and real-time traffic information (RTTI) over the road network such that an attacker cannot distinguish the locations in a concealing region (CR) from the real location using a variety of inference attacks including probabilistic and

movement boundary-based inferences. It also includes a CR pruning technique such that with a small compromise in privacy for infrequent sensitive locations, it is possible to ensure the privacy of frequently visited sensitive locations against long-term observation-based attack. However, there are several limitations to this work. First, it can incur high computation cost for computing the CR. Second, the spatial and temporal transition probabilities between locations are not considered for which an attacker with a model (e.g., Markov chain) based on such information can leak location privacy in the proposed scheme. Third, KLAP would leak privacy in case of group mobility. If an attacker gains information regarding group mobility, it can correlate the CRs of different users and compromise their privacy. For instance, in figure 3.1(b), if three different users generated the CRs in a short period, then an attacker can perform region intersection attack to get all of their location information at higher precision.

7.1.2 Location Inference Attacks on Geo-Tagged and Non geo-Tagged data and Their Countermeasures

We propose a location inference attack model to study the impact of a combination of spatiotemporally correlated geo-tagged and non-geo-tagged contents on existing trajectory obfuscation mechanisms. We design the model in the context of location-based social networks (LBSN) for checkins and photo-sharing and apply it on dummy-based obfuscation approach. In fact, to the best of our knowledge, this is the first work to investigate the impact of historical shared photos on location privacy. Our experiment with factual data reveals a negative impact of the attack model. To negate such a negative impact, we also propose an improved lightweight probabilistic countermeasure. Our contribution also includes a visualization ap-

proach to visualize privacy-preserving checkins and photo in LBSN. This work can be further improved by addressing the following limitations. Similar to our work on KLAP, the proposed approach does not consider the spatial and temporal transition probabilities between locations for both geo-tagged and non-geo-tagged data. Also, the proposed approach is only tested with historical data on checkins and photos. It can be improved further by considering different other types of information in LBSN, including, hashtags, tips, and video. Also, neither the attack model nor its countermeasure considers the group mobility of the users, which can be exploited by an attacker.

7.1.3 Spatiotemporal Blockchain Management for Resource-Constrained IoT devices to Achieve Decentralized Privacy

We introduce an innovative approach, coined as "Sensor-Chain", to make blockchain lightweight and privacy-friendly for sensor-based mobile IoT systems. We show that, by considering the spatial, temporal, and mobility information, it is possible to reduce the size of a blockchain while retaining important information about the system. The proposed approach ensures transaction privacy in a distributed and lightweight manner for the IoT sensor devices by adapting local differential privacy-preserving mechanism. We further demonstrate the efficiency and efficacy of Sensor-Chain by developing its prototype. The demonstration is still in its early stage, and it is not tested yet on IoT devices. Besides this, the proposed approach has several other drawbacks. The spatial subdivision of the plane in the proposed approach is static, which reduces the broad scope of the proposed approach. In addition to this,

the local differential privacy-preserving mechanism opens the door to distributed denial-of-service (DDoS) attack in the network. The malicious or compromised devices can collude and deploy DDoS attack on targeted devices by making a large number of transactions. Hence, the targeted devices will eventually run out of their privacy budget and the colluding devices, with their low privacy budget, can monopolize the network for some time. Another significant drawback is the lack of an approach to establish communication between the independent blockchains. As such, Sensor-Chain is not applicable in applications where different clusters of IoT nodes required inter-cluster communication.

7.1.4 Quantifying Location Privacy in Permissioned Blockchain-Based Internet of Things (IoT)

We propose a study of the privacy concerns with the location information in the scope of permissioned blockchain. The uniqueness of our study lies in consideration of short-term communication-based peer-to-peer location-based transactions for which an IoT node cannot falsify about its location. In this context, we design a location inference attack model and show that existing approaches, such as k-anonymity, dummies, or geo-indistinguishability, cannot be applied directly to preserve location information under such an attack model. We further propose a solution, an obfuscation mechanism based on the spatiotemporal mobility of the nodes, to nullify the attack model. The proposed approach is resilient against different types of attacks, including collusion, map matching, time reachability-based path reconstruction, and transaction attacks. However, it has several security limitations. It is susceptible against off-chain information-based attack. An attacker can combine off-chain information (e.g., information about the hours of operation of

a business) with the map matching attack to devise a better inference model. It is vulnerable against probabilistic inference attack from on-chain information. Here, an attacker can personalize the mobility of the node from the information available on the chain using machine learning algorithms and can able to improve the path reconstruction attack.

7.2 Future Work

Besides addressing the limitations mentioned above, we propose the following significant directions that can improve the current state-of-the-art and provide a better perspective of the future of trajectory privacy preservation techniques.

7.2.1 Secure and Privacy Preserving Inter- Blockchain Communication Scheme for IoT

In this dissertation, the research focuses on intra-blockchain communication. A significant direction to move forward with this dissertation is with the theory and development of efficient schemes for the different aspects of communication amongst multiple blockchains for mobility-centric IoT. The different research issues with interoperability among multiple blockchains have attracted some attention lately [FBS19, JDX18, ZJ18, BCD+14]. However, mobility-centric IoT is characterized by the communication latency, mobility, low resources, and short-range connectivity of the devices for which existing interoperability approaches for inter-blockchain communication cannot be applied directly in mobility-centric IoT applications. The goal of this research direction is to devise and implement efficient interoperability approaches for inter-blockchain communications. This research includes productive

asset searching across multiple blockchains, data interpretation, ensuring atomicity, and identifying and mitigating trajectory privacy and security issues in the process of interoperability.

Interoperability Method for Cross-Blockchain Data Search

The first step towards cross-blockchain communication is to ensure efficient data search under uncertainty. The efficacy and efficiency of a data search approach depend on its adaptability to different storage capabilities of the nodes, nodes unavailability, and link failure. In a blockchain, the IoT devices with shorter storage capacities would run out of storage space and cannot hold new blocks. One straightforward solution is to delete the old blocks from the storage [DOA16]. This raises the question of how to search an old block at a later time in the same network. One approach to address this problem is to flood the entire network [GHG+13], which is costly in terms of communication efficiency and latency. A better alternative is the LeapChain[RS18], which reduces the block traversal cost without compromising data integrity. It proposed to include one additional backlink such that it is possible to jump over many blocks without looking to the intermediate blocks. It assumes that there exists a prover device in the network which is capable of holding the whole blockchain. In a mobile scenario, such an assumption is weak as the prover devices can leave a network for many reasons, including becoming out of range, or due to power failure. All the prover devices can be out of a network where an intra-blockchain search would fail, and for which it is required to explore inter-blockchain data searching mechanisms. Existing approaches for cross-chain data searching, such as SideChain [BCD+14] or interoperable blockchain [JDX18], will also not work due to the mobility of the devices and uncertainty in the communication for which it is needed to address the problems with delay and drop in the communication.

Blockchain Switchover Under Uncertainty

To deal with the delay, one possible solution could be the blockchain switchover. Suppose, a node is trying to search specific data which can be found in multiple blockchains. However, it does not know with which blockchain to communicate. In such a case, querying the blockchain with higher throughput can yield better search performance [HLP18]. A throughput-based ranking of different blockchains [FBS19] can be helpful to design effective and efficient switchover technique. While blockchain switchover may also help with the search request drop, it cannot deal with the situation when there exists no blockchain to provide the desired data. Thus, it is necessary to integrate spatiotemporal constraints in the design of the switchover technique.

Security Issues in Inter-Blockchain Communication

The last and final step towards the design of inter-blockchain communication is to identify, quantify, and mitigate several security issues in inter-blockchain communications of mobility-centric IoTs. One crucial issue is the DDoS attack where an entire cluster of nodes acts maliciously and deploys DDoS attack on targeted blockchains. The malicious group does not use the security holes of a mobility-centric IoT but instead launches attacks on its availability. The impact of such group-based DDoS attack is far more severe than the DDoS attack performed on individuals as the volume of traffic generated by a malicious group can instantly paralyze the blockchains of victim mobility-centric IoTs. It is required to understand the nature of the DDoS attacks, which can be used for proactive detection and mitigation of the attack.

7.2.2 Developing Privacy-Preserving and Secure Scheme for Merging Multiple mobility-centric IoT Blockchains

Another untouched but very important issue to make blockchain a reality for the IoT is the enablement of real-time merging of multiple mobility-centric IoT blockchains into one. Consider multiple clusters of devices, each having its own blockchain with different consensus and lightweight mechanisms. At some point in time, for some reasons the clusters decide to merge into one. In such a scenario, the task is to merge the different blockchains into one in a secure and privacy-preserving way such that integrity of the individual blockchain remains intact. The task includes, reaching a consensus in the process of merging the blockchains, preventing data loss while merging with limited resource (e.g. limited storage capacity of the devices), and maintaining security and privacy in the process of merge.

Consensus on Consensus

Another untouched but crucial issue to make blockchain a reality for the IoT is the enablement of the real-time merging of multiple mobility-centric IoT blockchains into one. Consider multiple clusters of devices, each having its blockchain with different consensus and lightweight mechanisms. At some point in time, for some reasons, the clusters decide to merge into one. In such a scenario, the task is to merge the different blockchains into one in a secure and privacy-preserving way such that integrity of the individual blockchain remains intact. The task includes reaching a consensus in the process of merging the blockchains, preventing data loss while merging with a limited resource (e.g., the limited storage capacity of the devices), and maintaining security and privacy in the process of the merge.

Efficient Approach for Merging

Merging multiple blockchains into one is constrained by many factors. The first factor is the ordering of the merged blockchain and how to achieve it. In a blockchain, the blocks are linked with each other in a timely fashion using their hashes. If we want to maintain the time sequence, then the security by hashes would be violated, and a new hash would be required to generate. For large blockchains, such an approach can incur high delay and communication cost. Thus, thorough research is needed to device an efficient merging approach. Another vital factor associated with merging is the limited resource of the devices. Merging of multiple blockchains can yield an extensive blockchain which may not be within the storage capacity of a device. Thus, a merging approach must be made practical to deal with such a factor. There are much severe security and privacy factors that must be taken into account while designing merging an approach. For instance, colluding groups can utilize their majority benefit to take over the merged blockchain. Also, the merging can cause inter-blockchain privacy tension as the private data from one blockchain can be leaked due to the merging.

7.2.3 Distributed Spatiotemporal Federated Learning using Blockchain and Smart Contract

Traditional machine learning approaches require a centralized entity to collect data from all the users and learn a model from it. Such an approach leaks an individual's privacy, incurs high communication cost, and is susceptible to a variety of security vulnerabilities, including a single point of failure. Federate learning[KMY+16] was proposed to improve these problems by allowing to train a high-quality centralized model while training data remains distributed over the users. In this approach,

a preliminary model is shared by the centralized entity with each user; each user updates the model based on his/her local data, and send it back to the centralized entity. While federated learning promises better privacy and security than orthodox machine learning, it is still in an inchoate state and spatiotemporal aspect of this not fully understood. Besides, individual's data, in terms of the model, are still at the hand of the centralized entity for which the degree of trust and privacy can are questionable. One way to redesign the federated learning approach for spatiotemporal applications (e.g., mobile crowdsensing) with improved trust and privacy is by using blockchain and smart contract. The blockchain and smart contract would allow users to train machine learning models for a reward in a trustless manner without relying on a centralized entity. The smart contract will facilitate automatic validation of a shared model on the blockchain, and reward and penalize the users based on the output of the validation process. Such an approach has great potential in designing more secure and privacy-preserving location-based recommendation systems for IoT systems. Another potential is the creation of an AI market where parties good at solving machine learning problems can directly monetize their skillset, and others can solicit machine learning solutions.

7.3 Conclusion

In this dissertation, we present methods to identify and mitigate different privacy issues associated with the location information shared in an IoT system. The scope of this research covers both centralized and decentralized P2P architectures for IoT. In our first study, we propose an approach to define a balance between privacy and quality of service in real-time IoT systems such that a user's location information remains protected in the long-term of the use of the system. We further investi-

gate the issue with the consideration of system usages with both geo-tagged and non-geo-tagged data. We show that current existing approaches, designed with the focus of geo-tagged-data, leak privacy in the presence of heterogeneous data in the system. We propose a privacy-preserving framework for heterogeneous data sharing in the context of location-based social network (LBSN). We study the problem in blockchain-based decentralized IoT systems as well. Blockchain, despite its variety of benefits, is not directly applicable in resource-constrained IoT systems as it requires a large amount of resources on the IoT devices. Thus, before analyzing and addressing its privacy issues, we first propose an approach to reduce its storage requirement by considering the spatiotemporal mobility of IoT devices. Afterward, we study the trajectory privacy issue in a permissioned blockchain where the short-range communication between the devices forms proof-of-locations. We propose an obfuscation technique that quantifies, both theoretically and experimentally, the relationship between privacy and utility to dynamically protect the privacy of sensitive locations in the permissioned blockchain.

BIBLIOGRAPHY

[ABCP13] Miguel E Andrés, Nicolás E Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 901–914. ACM, 2013.

[ABMZ18] M. Amoretti, G. Brambilla, F. Medioli, and F. Zanichelli. Blockchain-based proof of location. In *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 146–153, July 2018.

[ACdVS11] Claudio A Ardagna, Marco Cremonini, Sabrina De Capitani di Vimercati, and Pierangela Samarati. An obfuscation-based approach for protecting location privacy. *IEEE Transactions on Dependable and Secure Computing*, 8(1):13–27, 2011.

[AHHH16] Berker Ağır, Kévin Huguenin, Urs Hengartner, and Jean-Pierre Hubaux. On the privacy implications of location semantics. *Proceedings on Privacy Enhancing Technologies*, 2016(4):165–183, 2016.

[AIM10] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[AIRX08] Waleed Alsalih, Kamrul Islam, Yurai Núñez Rodríguez, and Henry Xiao. Distributed voronoi diagram computation in wireless sensor networks. In *SPAA*, page 364, 2008.

[AKKH18] G. Ayoade, V. Karande, L. Khan, and K. Hamlen. Decentralized iot data management using blockchain and trusted execution environment. In *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 15–22, July 2018.

[ama] Amazon qldb. `https://aws.amazon.com/qldb/`.

[AMM+18] Pelin Angin, Melih Burak Mert, Okan Mete, Azer Ramazanli, Kaan Sarica, and Bora Gungoren. A blockchain-based decentralized security architecture for iot. In *Internet of Things – ICIOT 2018*, pages 3–18, Cham, 2018. Springer International Publishing.

[ARNK19] Abdulhamid Adebayo, Danda B Rawat, Laurent Njilla, and Charles A Kamhoua. Blockchain-enabled information sharing framework for cybersecurity. *Blockchain for Distributed Systems Security*, pages 143–158, 2019.

[ASN+19] Ashar Ahmad, Muhammad Saad, Laurent Njilla, Charles Kamhoua, Mostafa Bassiouni, and Aziz Mohaisen. Blocktrail: A scalable multichain solution for blockchain-based audit trails. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2019.

[Axo15] Louise Axon. Privacy-awareness in blockchain-based pki. 2015.

[Azu] Microsoft Azure. Blockchain. `https://azure.microsoft.com/en-us/solutions/blockchain/`.

[BAZ16] Giacomo Brambilla, Michele Amoretti, and Francesco Zanichelli. Using blockchain for peer-to-peer proof-of-location. *arXiv preprint arXiv:1607.00174*, 2016.

[BCD+14] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. Enabling blockchain innovations with pegged sidechains. 2014.

[BCHC09] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[BCP14] Nicolás E Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Optimal geo-indistinguishable mechanisms for location privacy. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 251–262. ACM, 2014.

[BKP14] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. Deanonymisation of clients in bitcoin p2p network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 15–29. ACM, 2014.

[BNM+14] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A Kroll, and Edward W Felten. Mixcoin: Anonymity for bitcoin with accountable mixes. In *International Conference on Financial Cryptography and Data Security*, pages 486–504. Springer, 2014.

[BS16] V. Bindschaedler and R. Shokri. Synthesizing plausible privacy-preserving location traces. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 546–563, May 2016.

[BSM16] Abdelwahab Boualouache, Sidi-Mohammed Senouci, and Samira Moussaoui. Towards an efficient pseudonym management and changing scheme for vehicular ad-hoc networks. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE, 2016.

[CCLS11] Zhiyuan Cheng, James Caverlee, Kyumin Lee, and Daniel Z Sui. Exploring millions of footprints in location sharing services. In *Fifth International AAAI Conference on Weblogs and Social Media*, 2011.

[CPNX19] Huashan Chen, Marcus Pendleton, Laurent Njilla, and Shouhuai Xu. A survey on ethereum systems security: Vulnerabilities, attacks and defenses. *arXiv preprint arXiv:1908.04507*, 2019.

[CS16] S. Chen and H. Shen. Semantic-aware dummy selection for location privacy preservation. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 752–759, Aug 2016.

[CVPC19] Roberto Casado-Vara, Javier Prieto, and Juan M. Corchado. How blockchain could improve fraud detection in power distribution grid. In *International Joint Conference SOCO'18-CISIS'18-ICEUTE'18*, pages 67–76, Cham, 2019. Springer International Publishing.

[CYXX17] Y. Cao, M. Yoshikawa, Y. Xiao, and L. Xiong. Quantifying differential privacy under temporal correlations. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 821–832, April 2017.

[DBS10] Maria Luisa Damiani, Elisa Bertino, and Claudio Silvestri. The probe framework for the personalized cloaking of private locations. *Transactions on Data Privacy*, 2010.

[DGP17] Arnaud Durand, Pascal Gremaud, and Jacques Pasquier. Decentralized web of trust and authentication for the internet of things. In *Proceedings of the Seventh International Conference on the Internet of Things*, IoT '17, pages 27:1–27:2, New York, NY, USA, 2017. ACM.

[DIIP19] Kostas Drakonakis, Panagiotis Ilia, Sotiris Ioannidis, and Jason Polakis. Please forget where i was last summer: The privacy risks of public location (meta) data. *arXiv preprint arXiv:1901.00897*, 2019.

[DKJG17] Ali Dorri, Salil S Kanhere, Raja Jurdak, and Praveen Gauravaram. Blockchain for iot security and privacy: The case study of a smart home. In *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on*, pages 618–623. IEEE, 2017.

[DMHVB13] Yves-Alexandre De Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3:1376, 2013.

[DOA16] Richard Dennis, Gareth Owenson, and Benjamin Aziz. A temporal blockchain: a formal analysis. In *Collaboration Technologies and Systems (CTS), 2016 International Conference on*, pages 430–437. IEEE, 2016.

[DSKJ17] Ali Dorri, Marco Steger, Salil S Kanhere, and Raja Jurdak. Blockchain: A distributed solution to automotive security and privacy. *IEEE Communications Magazine*, 55(12):119–125, 2017.

[Dwo11] Cynthia Dwork. Differential privacy. *Encyclopedia of Cryptography and Security*, pages 338–340, 2011.

[FBS19] Philipp Frauenthaler, Michael Borkowski, and Stefan Schulte. A framework for blockchain interoperability and runtime selection. *arXiv preprint arXiv:1905.07014*, 2019.

[FKP19] Sean Foley, Jonathan R Karlsen, and Tālis J Putniņš. Sex, drugs, and bitcoin: How much illegal activity is financed through cryptocurrencies? *The Review of Financial Studies*, 32(5):1798–1853, 2019.

[fou] Foursquare. `https://foursquare.com/about`.

[Fou18] Iota Foundation. Iota, 2018.

[Fra14] Pedro Franco. *Understanding Bitcoin: Cryptography, engineering and economics*. John Wiley & Sons, 2014.

[GDSB16] Gabriel Ghinita, Maria Luisa Damiani, Claudio Silvestri, and Elisa Bertino. Protecting against velocity-based, proximity-based, and external event attacks in location-centric social networks. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, 2(2):8, 2016.

[GHG+13] Shuo Guo, Liang He, Yu Gu, Bo Jiang, and Tian He. Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links. *IEEE Transactions on Computers*, 63(11):2787–2802, 2013.

[GL08] Buğra Gedik and Ling Liu. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *IEEE Transactions on Mobile Computing*, 7(1):1–18, January 2008.

[GMG18] N. Guo, L. Ma, and T. Gao. Independent mix zone for location privacy in vehicular networks. *IEEE Access*, 6:16842–16850, 2018.

[GMS+13] Sheng Gao, Jianfeng Ma, Weisong Shi, Guoxing Zhan, and Cong Sun. Trpf: A trajectory privacy-preserving framework for participatory sensing. *IEEE Transactions on Information Forensics and Security*, 8(6):874–887, 2013.

[goo] Google images. `https://images.google.com/`.

[GPI15] Mingming Guo, Niki Pissinou, and S Sitharama Iyengar. Pseudonym-based anonymity zone generation for mobile service with strong adversary model. In *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pages 335–340. IEEE, 2015.

[GPI16] Mingming Guo, Niki Pissinou, and S. S. Iyengar. Privacy-aware mobile sensing in vehicular networks. In *2016 International Conference on Computing, Networking and Communications (ICNC)*, pages 1–5, Feb 2016.

[Gra18] Kevin Granville. Facebook and cambridge analytica: What you need to know as fallout widens. `https://www.nytimes.com/2018/03/19/technology/facebook-cambridge-analytica-explained.html`, journal=The New York Times, Mar 2018.

[HBM+18] Kévin Huguenin, Igor Bilogrevic, Joana Soares Machado, Stefan Mihaila, Reza Shokri, Italo Dacosta, and Jean-Pierre Hubaux. A predictive model for user motivation and utility implications of privacy-protection mechanisms in location check-ins. *IEEE Transactions on Mobile Computing*, 17(4):760–774, 2018.

[HLB+15] Jianping He, Bin Liu, Xuan Bao, Hongxia Jin, and George Kesidis. On privacy preserving partial image sharing. In *2015 IEEE 35th Interna-*

tional Conference on Distributed Computing Systems, pages 758–759. IEEE, 2015.

[HLP18] Thomas Hardjono, Alexander Lipton, and Alex Pentland. Towards a design philosophy for interoperable blockchain systems. *arXiv preprint arXiv:1805.05934*, 2018.

[HSI+16] T. Hara, A. Suzuki, M. Iwata, Y. Arase, and X. Xie. Dummy-based user location anonymization under real-world constraints. *IEEE Access*, 4:673–687, 2016.

[HV07] Esa Hyytiä and Jorma Virtamo. Random waypoint mobility model in cellular networks. *Wireless Networks*, 13(2):177–188, 2007.

[IoT15] IoT.Business.News. Location-based service revenues will grow to €34.8 billion in 2020. `https://iotbusinessnews.com/2015/08/31/`, 2015. Accessed: 2019-03-19.

[JDX18] H. Jin, X. Dai, and J. Xiao. Towards a novel architecture for enabling interoperability amongst multiple blockchains. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1203–1211, July 2018.

[KAG18] Amira Kchaou, Ryma Abassi, and Sihem Guemara. Toward a distributed trust management scheme for vanet. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, page 53. ACM, 2018.

[KKM14] Philip Koshy, Diana Koshy, and Patrick McDaniel. An analysis of anonymity in bitcoin using p2p network traffic. In *International Conference on Financial Cryptography and Data Security*, pages 469–485. Springer, 2014.

[KMY+16] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[Kru07] John Krumm. Inference attacks on location tracks. In *International Conference on Pervasive Computing*, pages 127–143. Springer, 2007.

[KXN+18] Jiawen Kang, Zehui Xiong, Dusit Niyato, Dongdong Ye, Dong In Kim, and Jun Zhao. Towards secure blockchain-enabled internet of vehicles: Optimizing consensus management using reputation and contract theory. *arXiv preprint arXiv:1809.08387*, 2018.

[LARC16] Ilaria Liccardi, Alfie Abdul-Rahman, and Min Chen. I know where you live: Inferring details of people's lives by visualizing publicly shared location data. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 1–12. ACM, 2016.

[LBTC+15a] Cédric Le Barz, Nicolas Thome, Matthieu Cord, Stéphane Herbin, and Martial Sanfourche. Absolute geo-localization thanks to hidden markov model and exemplar-based metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 9–17, 2015.

[LBTC+15b] Cédric Le Barz, Nicolas Thome, Matthieu Cord, Stéphane Herbin, and Martial Sanfourche. Exemplar based metric learning for robust visual localization. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 4342–4346. IEEE, 2015.

[LCTZ13] J. Z. Liang, N. Corso, E. Turner, and A. Zakhor. Image based localization in indoor environments. In *2013 Fourth International Conference on Computing for Geospatial Research and Application*, pages 70–75, July 2013.

[LHA+16] D. Liao, X. Huang, V. Anand, G. Sun, and H. Yu. k-dlca: An efficient approach for location privacy preservation in location-based services. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6, May 2016.

[LJY08] Hua Lu, Christian S. Jensen, and Man Lung Yiu. Pad: Privacy-area aware, dummy-based location privacy in mobile services. In *Proceedings of the Seventh ACM International Workshop on Data Engineering for Wireless and Mobile Access*, MobiDE '08, pages 16–23, New York, NY, USA, 2008. ACM.

[LLC+18] L. Li, J. Liu, L. Cheng, S. Qiu, W. Wang, X. Zhang, and Z. Zhang. Creditcoin: A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 19(7):2204–2220, July 2018.

[LLL+11] Rongxing Lu, Xiaodong Lin, Tom H Luan, Xiaohui Liang, and Xuemin Shen. Pseudonym changing at social spots: An effective strategy for location privacy in vanets. *IEEE transactions on vehicular technology*, 61(1):86–96, 2011.

[LLL+17] H. Liu, X. Li, H. Li, J. Ma, and X. Ma. Spatiotemporal correlation-aware dummy-based privacy protection scheme for location-based services. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9, May 2017.

[LMNZ18] Roben Castagna Lunardi, Regio Antonio Michelin, Charles Varlei Neu, and Avelino Francisco Zorzo. Distributed access control on iot ledger-based architecture. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–7. IEEE, 2018.

[LSL+19] Fenghua Li, Zhe Sun, Ang Li, Ben Niu, Hui Li, and Guohong Cao. Hideme: Privacy-preserving photo sharing on social networks. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 154–162. IEEE, 2019.

[LSTL13] Ming Li, S. Salinas, A. Thapa, and Pan Li. n-cd: A geometric approach to preserving location privacy in location-based services. In *INFOCOM, 2013 Proceedings IEEE*, pages 3012–3020, April 2013.

[McD09] John H McDonald. *Handbook of biological statistics*, volume 2. 2009.

[MDB17] Axel Moinet, Benoît Darties, and Jean-Luc Baril. Blockchain based trust & authentication for decentralized sensor networks. *arXiv preprint arXiv:1706.01730*, 2017.

[MDFFF17] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[MDL+18] Regio A Michelin, Ali Dorri, Roben C Lunardi, Marco Steger, Salil S Kanhere, Raja Jurdak, and Avelino F Zorzo. Speedychain: A framework for decoupling data from blockchain for smart cities. *arXiv preprint arXiv:1807.01980*, 2018.

[MDS+18] Regio A. Michelin, Ali Dorri, Marco Steger, Roben C. Lunardi, Salil S. Kanhere, Raja Jurdak, and Avelino F. Zorzo. Speedychain: A frame-

work for decoupling data from blockchain for smart cities. In *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MobiQuitous '18, pages 145–154, New York, NY, USA, 2018. ACM.

[MF17] Jethro Mullen and Seth Fiegerman. Yahoo tops the list of largest ever data breaches. `https://money.cnn.com/2017/10/04/technology/yahoo-biggest-data-breaches-ever/index.html`, 2017.

[MKGV07] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3, 2007.

[Mor19] J.P Morgan. Quorum. `https://www.goquorum.com/`, 2019.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *bitcoin.org*, 2008.

[NLZ+14] Ben Niu, Qinghua Li, Xiaoyan Zhu, Guohong Cao, and Hui Li. Achieving k-anonymity in privacy-aware location-based services. In *INFOCOM, 2014 Proceedings IEEE*, pages 754–762. IEEE, 2014.

[NLZ+15] B. Niu, Q. Li, X. Zhu, G. Cao, and H. Li. Enhancing privacy through caching in location-based services. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 1017–1025, April 2015.

[Nov18] O. Novo. Blockchain meets iot: An architecture for scalable access management in iot. *IEEE Internet of Things Journal*, 5(2):1184–1195, April 2018.

[NXN+07] SY Nikouei, R Xu, D Nagothu, Y Chen, A Aved, and E Blasch. Real-time index authentication for event-oriented surveillance video query using blockchain. arxiv 2018. *arXiv preprint arXiv:1807.06179*, 1807.

[NXN+18] Seyed Yahya Nikouei, Ronghua Xu, Deeraj Nagothu, Yu Chen, Alexander Aved, and Erik Blasch. Real-time index authentication for event-oriented surveillance video query using blockchain. In *2018 IEEE International Smart Cities Conference (ISC2)*, pages 1–8. IEEE, 2018.

[NZLL14] B. Niu, Z. Zhang, X. Li, and H. Li. Privacy-area aware dummy generation algorithms for location-based services. In *2014 IEEE International Conference on Communications (ICC)*, pages 957–962, June 2014.

[OBS18] Abdallah Zoubir Ourad, Boutheyna Belgacem, and Khaled Salah. Using blockchain for iot access control and authentication management. In *Internet of Things – ICIOT 2018*, pages 150–164, Cham, 2018. Springer International Publishing.

[OFS17] S. J. Oh, M. Fritz, and B. Schiele. Adversarial image perturbation for privacy protection a game theory perspective. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1491–1500, Oct 2017.

[OHS+17] Alexandra-Mihaela Olteanu, Kévin Huguenin, Reza Shokri, Mathias Humbert, and Jean-Pierre Hubaux. Quantifying interdependent privacy risks with location data. *IEEE Transactions on Mobile Computing*, 16(3):829–842, 2017.

[PL15] Balaji Palanisamy and Ling Liu. Attack-resilient mix-zones over road networks: Architecture and algorithms. *IEEE Transactions on Mobile Computing*, (3):495–508, 2015.

[PLMW17] Tao Peng, Qin Liu, Dacheng Meng, and Guojun Wang. Collaborative trajectory privacy preserving scheme in location-based services. *Information Sciences*, 387:165–179, 2017.

[PPJP12] Sitthapon Pumpichet, Niki Pissinou, Xinyu Jin, and Deng Pan. Belief-based cleaning in trajectory sensor streams. In *Communications (ICC), 2012 IEEE International Conference on*, pages 208–212. IEEE, 2012.

[PSFK14] Jonathan Petit, Florian Schaub, Michael Feiri, and Frank Kargl. Pseudonym schemes in vehicular networks: A survey. *IEEE communications surveys & tutorials*, 17(1):228–255, 2014.

[PWH+18] Jianli Pan, Jianyu Wang, Austin Hester, Ismail Alqerm, Yuanni Liu, and Ying Zhao. Edgechain: An edge-iot framework and prototype based on blockchain and smart contracts. *arXiv preprint arXiv:1806.06185*, 2018.

[RDJK18] Clemence Roulin, Ali Dorri, Raja Jurdak, and Salil Kanhere. On the activity privacy of blockchain for iot. *arXiv preprint arXiv:1812.08970*, 2018.

[RH13] Oliver Roick and Susanne Heuser. L ocation b ased s ocial n etworks–definition, current state of the art and research agenda. *Transactions in GIS*, 17(5):763–784, 2013.

[RKH+19] Jonathan Rusert, Osama Khalid, Dat Hong, Zubair Shafiq, and Padmini Srinivasan. No place to hide: Inadvertent location privacy leaks on twitter. *Proceedings on Privacy Enhancing Technologies*, 2019(4):172–189, 2019.

[RNKK18] Danda B Rawat, Laurent Njilla, Kevin Kwiat, and Charles Kamhoua. ishare: Blockchain-based privacy-aware multi-agent information sharing games for cybersecurity. In *2018 International Conference on Computing, Networking and Communications (ICNC)*, pages 425–431. IEEE, 2018.

[RS18] Emanuel Regnath and Sebastian Steinhorst. Leapchain: efficient blockchain verification for embedded iot. In *Proceedings of the International Conference on Computer-Aided Design*, page 74. ACM, 2018.

[SC18] Rohit Sharma and Suchetana Chakraborty. Blockapp: Using blockchain for authentication and privacy preservation in iov. In *2018 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6. IEEE, 2018.

[SJZ+17] A. R. Shahid, L. Jeukeng, Wei Zeng, N. Pissinou, S. S. Iyengar, S. Sahni, and M. Varela-Conover. Ppvc: Privacy preserving voronoi cell for location-based services. In *2017 International Conference on Computing, Networking and Communications (ICNC)*, pages 351–355, Jan 2017.

[SK17] Madhusudan Singh and Shiho Kim. Blockchain based intelligent vehicle data sharing framework. *arXiv preprint arXiv:1708.09721*, 2017.

[SMP17] Pradip Kumar Sharma, Seo Yeon Moon, and Jong Hyuk Park. Block-vn: A distributed blockchain based vehicular network architecture in smart city. *Journal of Information Processing Systems*, 13(1):84, 2017.

[SNKM19] Muhammad Saad, Laurent Njilla, Charles Kamhoua, and Aziz Mohaisen. Countering selfish mining in blockchains. In *2019 International Conference on Computing, Networking and Communications (ICNC)*, pages 360–364. IEEE, 2019.

[SPI+18] A. R. Shahid, N. Pissinou, S. S. Iyengar, J. Miller, Z. Ding, and T. Lemus. Klap for real-world protection of location privacy. In *2018 IEEE World Congress on Services (SERVICES)*, pages 17–18, July 2018.

[SPI+19] A. R. Shahid, N. Pissinou, S. S. Iyengar, K. Makki, and M. Klass. Delay-aware privacy-preserving location-based services under spatiotemporal constraints. *Journal of Communications*, pages 1–10, 2019.

[SPIM18] A. R. Shahid, N. Pissinou, S. S. Iyengar, and K. Makki. Check-ins and photos: Spatiotemporal correlation-based location inference attack and defense in location-based social networks. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 1852–1857, Aug 2018.

[SPN+19a] A. R. Shahid, Niki Pissinou, Laurent Njilla, Edwin Aguilar, and Eric Perez. Demo:towards the development of a differentially private lightweight and scalable blockchain for iot. In *2019 16th IEEE International Conference on Mobile Ad-Hoc and Smart Systems (MASS-2019)*, Monterey, CA, USA, November 2019.

[SPN+19b] A. R. Shahid, Niki Pissinou, Laurent Njilla, Sheila Alemany, Ahmed Imteaj, and Kia Makki. Quantifying location privacy in permissioned blockchain-based internet of things (iot). In *The 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous 2019)*, Houston, USA, November 2019.

[SPSK19] A. R. Shahid, Niki Pissinou, Corey Staier, and Rain Kwan. SensorChain: a lightweight scalable blockchain framework for internet of things. In *The 2019 IEEE International Conference on Internet of Things (iThings-2019)*, Atlanta, USA, July 2019.

[SRK+17] Sachin Shetty, Val Red, Charles Kamhoua, Kevin Kwiat, and Laurent Njilla. Data provenance assurance in the cloud using blockchain. In *Disruptive Technologies in Sensors and Sensor Systems*, volume 10206, page 102060I. International Society for Optics and Photonics, 2017.

[SSN+19] Muhammad Saad, Jeffrey Spaulding, Laurent Njilla, C Kamhoua, DaeHun Nyang, and Aziz Mohaisen. Overview of attack surfaces in blockchain. *Blockchain for Distributed System Security*, pages 51–66, 2019.

[Sta17] Statista. U.s.location-based service users 2013-2018. `https://www.statista.com/statistics/436071/location-based-service-users-usa/`, 2017.

[Sta19] Statista. Number of smartphone users worldwide from 2014 to 2020 (in billions). `https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/`, 2019. Accessed: 2019-04-02.

[STP+14] R. Shokri, G. Theodorakopoulos, P. Papadimitratos, E. Kazemi, and J. Hubaux. Hiding in the mobile crowd: Locationprivacy through collaboration. *IEEE Transactions on Dependable and Secure Computing*, 11(3):266–279, May 2014.

[TPFF+15] Kevin Tang, Manohar Paluri, Li Fei-Fei, Rob Fergus, and Lubomir Bourdev. Improving image classification with location context. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[TPI17] Samia Tasnim, Niki Pissinou, and SS Iyengar. A novel cleaning approach of environmental sensing data streams. In *Consumer Communications & Networking Conference (CCNC), 2017 14th IEEE Annual*, pages 632–633. IEEE, 2017.

[TSL+19a] Deepak Tosh, Sachin Shetty, Xueping Liang, Charles Kamhoua, and Laurent L Njilla. Data provenance in the cloud: A blockchain-based approach. *IEEE Consumer Electronics Magazine*, 8(4):38–44, 2019.

[TSL+19b] Deepak Tosh, Sachin S Shetty, Xueping Liang, Laurent Njilla, Charles A Kamhoua, and Kevin Kwiat. Blockcloud security analysis. *Blockchain for Distributed Systems Security*, pages 159–192, 2019.

[VMG+01] Kirsi Virrantaus, Jouni Markkula, Artem Garmash, Vagan Terziyan, Jari Veijalainen, Artem Katanosov, and Henry Tirri. Developing gis-supported location-based services. In *Proceedings of the Second International Conference on Web Information Systems Engineering*, volume 2, pages 66–75. IEEE, 2001.

[why19] whyrusleeping. go-libp2p-pubsub. `https://github.com/libp2p/go-libp2p-pubsub`, 2019.

[WKP16] Tobias Weyand, Ilya Kostrikov, and James Philbin. Planet-photo geolocation with convolutional neural networks. In *European Conference on Computer Vision*, pages 37–55. Springer, 2016.

[WLYD17] Xiaotong Wu, Shu Li, Jun Yang, and Wanchun Dou. A cost sharing mechanism for location privacy preservation in big trajectory data. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2017.

[WvB14] Dominic Wörner and Thomas von Bomhard. When your sensor earns money: exchanging data for cash with bitcoin. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, pages 295–298. ACM, 2014.

[WY19] Shibin Wang and Nianmin Yao. A rsu-aided distributed trust framework for pseudonym-enabled privacy preservation in vanets. *Wireless Networks*, 25(3):1099–1115, 2019.

[WYGG18] Shibin Wang, Nianmin Yao, Ning Gong, and Zhenguo Gao. A trigger-based pseudonym exchange scheme for location privacy preserving in vanets. *Peer-to-Peer Networking and Applications*, 11(3):548–560, 2018.

[XCBC18] Ronghua Xu, Yu Chen, Erik Blasch, and Genshe Chen. Blendcac: A blockchain-enabled decentralized capability-based access control for iots. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1027–1034. IEEE, 2018.

[XCBC19] Ronghua Xu, Yu Chen, Erik Blasch, and Genshe Chen. Exploration of blockchain-enabled decentralized capability-based access control strat-

egy for space situation awareness. *Optical Engineering*, 58(4):041609, 2019.

[XFN+17] Zehui Xiong, Shaohan Feng, Dusit Niyato, Ping Wang, and Zhu Han. Edge computing resource management and pricing for mobile blockchain. *CoRR*, abs/1710.01567, 2017.

[XFW+18] Z. Xiong, S. Feng, W. Wang, D. Niyato, P. Wang, and Z. Han. Cloud/fog computing resource management and pricing for blockchain networks. *IEEE Internet of Things Journal*, pages 1–1, 2018.

[XNC+19] Ronghua Xu, Seyed Yahya Nikouei, Yu Chen, Erik Blasch, and Alex Aved. Blendmas: A blockchain-enabled decentralized microservices architecture for smart public safety. *arXiv preprint arXiv:1902.10567*, 2019.

[XX15] Yonghui Xiao and Li Xiong. Protecting locations with differential privacy under temporal correlations. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1298–1309. ACM, 2015.

[XZN+17] Zehui Xiong, Yang Zhang, Dusit Niyato, Ping Wang, and Zhu Han. When mobile blockchain meets edge computing: challenges and applications. *arXiv preprint arXiv:1711.05938*, 2017.

[YFX13] Dejun Yang, Xi Fang, and Guoliang Xue. Truthful incentive mechanisms for k-anonymity location privacy. In *2013 Proceedings IEEE INFOCOM*, pages 2994–3002. IEEE, 2013.

[YKH+16] Rong Yu, Jiawen Kang, Xumin Huang, Shengli Xie, Yan Zhang, and Stein Gjessing. Mixgroup: Accumulative pseudonym exchanging for location privacy enhancement in vehicular social networks. *IEEE Transactions on Dependable and Secure Computing*, (1):93–105, 2016.

[YLP17] Lei Yu, Ling Liu, and Calton Pu. Dynamic differential location privacy with personalized error bounds. In *NDSS*, 2017.

[YLX+17] Ayong Ye, Yacheng Li, Li Xu, Qing Li, and Hui Lin. A trajectory privacy-preserving algorithm based on road networks in continuous location-based services. In *Trustcom/BigDataSE/ICESS, 2017 IEEE*, pages 510–516. IEEE, 2017.

[YMH15] Bidi Ying, Dimitrios Makrakis, and Zhengzhou Hou. Motivation for protecting selfish vehicles' location privacy in vehicular networks. *IEEE Transactions on Vehicular Technology*, 64(12):5631–5641, 2015.

[YWN+18] B. Yu, J. Wright, S. Nepal, L. Zhu, J. Liu, and R. Ranjan. Iotchain: Establishing trust in the internet of things ecosystem using blockchain. *IEEE Cloud Computing*, 5(4):12–23, Jul./Aug. 2018.

[YYL+18] Zhe Yang, Kan Yang, Lei Lei, Kan Zheng, and Victor CM Leung. Blockchain-based decentralized trust management in vehicular networks. *IEEE Internet of Things Journal*, 2018.

[YZL+19] Mengmeng Yang, Tianqing Zhu, Kaitai Liang, Wanlei Zhou, and Robert H Deng. A blockchain-based location privacy-preserving crowdsensing system. *Future Generation Computer Systems*, 94:408–418, 2019.

[YZZY15] Dingqi Yang, Daqing Zhang, Vincent W Zheng, and Zhiyong Yu. Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(1):129–142, 2015.

[Zha19] Yang Zhang. Language in our time: An empirical analysis of hashtags. In *The World Wide Web Conference*, WWW '19, pages 2378–2389, New York, NY, USA, 2019. ACM.

[ZHR+18] Yang Zhang, Mathias Humbert, Tahleen Rahman, Cheng-Te Li, Jun Pang, and Michael Backes. Tagvisor: A privacy advisor for sharing hashtags. In *Proceedings of the 2018 World Wide Web Conference*, pages 287–296. International World Wide Web Conferences Steering Committee, 2018.

[ZJ18] K. Zhang and H. Jacobsen. Towards dependable, scalable, and pervasive distributed ledgers with blockchains. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1337–1346, July 2018.

[ZLS+16] Honggang Zhang, Benyuan Liu, Hengky Susanto, Guoliang Xue, and Tong Sun. Incentive mechanism for proximity-based mobile crowd service systems. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.

[ZLZ+18] P. Zhao, J. Li, F. Zeng, F. Xiao, C. Wang, and H. Jiang. Illia: Enabling$k$-anonymity-based privacy preserving against location injection attacks in continuous lbs queries. *IEEE Internet of Things Journal*, 5(2):1033–1042, April 2018.

[ZN+15] Guy Zyskind, Oz Nathan, et al. Decentralizing privacy: Using blockchain to protect personal data. In *Security and Privacy Workshops (SPW), 2015 IEEE*, pages 180–184. IEEE, 2015.

[ZTZ16] Yuan Zhang, Wei Tong, and Sheng Zhong. On designing satisfaction-ratio-aware truthful incentive mechanisms for k-anonymity location privacy. *IEEE Transactions on Information Forensics and Security*, 11(11):2528–2541, 2016.

VITA

ABDUR RAHMAN BIN SHAHID

	Born, Chittagong, Bangladesh
2011	B.Sc., Computer Science and Engineering Chittagong University of Engineering and Technology Chittagong, Bangladesh
2018	M.S., Computer Science Florida International University Miami, Florida
2019	Doctoral Candidate, Computer Science Florida International University Miami, Florida

PUBLICATIONS AND PRESENTATIONS

1. Shahid, A.R., Pissinou, N., Iyengar, S. S., Miller, J., Ding, Z., & Lemus, T. (2018, July). KLAP for Real-World Protection of Location Privacy. In *2018 IEEE World Congress on Services (SERVICES)* (pp. 17-18). IEEE.
2. Shahid, A.R., Pissinou, N., Iyengar, S. S., & Makki, K. (2018, August). Check-ins and Photos: Spatiotemporal Correlation-Based Location Inference Attack and Defense in Location-Based Social Networks. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)* (pp. 1852-1857). IEEE.
3. Shahid, A.R., Pissinou, N., Staier, C., & Kwan, R. (2019, July). Sensor-Chain: A Lightweight Scalable Blockchain Framework for Internet of Things. In *2019 IEEE international conference on Internet of Things (iThings).* IEEE.
4. Shahid, A.R., Pissinou, N., Njilla, L., Alemany, S., Imteaj, A., Makki, K., & Aguilar, E. (2019, November). Quantifying Location Privacy in Permissioned Blockchain-Based Internet of Things (IoT). In *2019 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous).* EAI.
5. Shahid, A.R., Pissinou, N., Njilla, L., Aguilar. E., & Perez. E. (2019, November). Demo: Towards the Development of a Differentially Private Lightweight and Scalable Blockchain for IoT. In *16th IEEE International Conference on Mobile Ad-Hoc and Smart Systems (MASS),* IEEE.

6. Shahid, A.R., Pissinou, N., Iyengar, S.S., & Makki,K. (2019). Delay-Aware Privacy-Preserving Location-Based Services under Spatiotemporal Constraints. In *Journal of Communications* (under submission).

7. Tang,Y., Tasnim, S., Pissinou, N., Iyengar, S.S., & Shahid, A.R. (2018) Reputation-Aware Data Fusion and Malicious Participant Detection in Mobile Crowdsensing. In *2018 IEEE International Conference on Big Data (Big Data)* (pp. 4820-4828). IEEE.