UNIVERSITY OF MISSOURI - COLUMBIA

Privacy Preservation in Mobile Social Networks

by

Douglas Steiert

A dissertation submitted in partial fulfillment for the degree of Doctor of Philosophy

in the
Department of Electrical Engineering and Computer Science
Advisor: Dan Lin

May 2019

The undersigned, appointed by the dean of the Graduate School, have examined the dissertation entitled

PRIVACY PRESERVATION IN
MOBILE SOCIAL NETWORKS

presented by Douglas Steiert,
a candidate for the degree of doctor of philosophy, and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Dan Lin

Dr. Wei Jiang

Dr. Prasad Calyam

Dr. Ye Duan

Dr. Hong He

Acknowledgements

I would first like to thank the Scholarship for Service (SFS) program for enabling my dream to pursue a higher education degree possible. Without receiving this scholarship, it is uncertain when or if I would have ever extended my knowledge in this realm of research and achieved a higher education. The program also allowed time for fellow scholars, Dr. Dan Lin, and Dr. Wei Jiang to get together to discuss each other's work, current cybersecurity trends or threats, and various cybersecurity events happening which helped keep me learning in other areas aside from my research. Thanks are also given to Dr. Dan Lin for taking me under her guidance in being my advisor and helping me push through the program. She provided immense advice in research direction, as well as support when times were rough and I struggled. Without Dr. Dan Lin, I definitely would not be able to discuss the research I have done and advocate for further research efforts, and for that, I am forever grateful to her.

I would also like to extend gratitude to my committee members Dr. Wei Jiang, Dr. Ye Duan, Dr. Prasad Calyam, and Dr. Hong He. Having transferred to the University of Missouri - Columbia without knowing many faculty, they accepted to be apart of my doctoral committee and have helped give excellent feedback and advice to me this past year.

Lastly, I would like to give thanks to my friends and family that have been with me throughout these years in graduate school. My friends have kept me social and create strong friendships that will last years that I may not have made without them there at school with me. For my family, your undying support for me to keep pushing through and words of encouragement have helped me stay on track.

Contents

List of Figures

Abbreviations

LBS	**L**ocation **B**ased **S**ervice
API	**A**pplication **P**rogramming **I**nterface
PIR	**P**rivate **I**nformation **R**etrieval
OT	**O**blivious **T**ransfer
IOT	**I**nternet-**o**f-**T**hings
AOI	**A**rea **O**f **I**nterest
POI	**P**oint **O**f **I**nterest
MITM	**M**an **I**n **T**he **M**iddle
VPN	**V**irtual **P**rivate **N**etwork
TOR	**T**he **O**nion **R**outer
REMIND	**R**isk **E**stimation **M**echanism for **I**mages in **N**etwork **D**istribution

Abstract

In this day and age with the prevalence of smartphones, networking has evolved in an intricate and complex way. With the help of a technology-driven society, the term "social networking" was created and came to mean using media platforms such as Myspace, Facebook, and Twitter to connect and interact with friends, family, or even complete strangers. Websites are created and put online each day, with many of them possessing hidden threats that the average person does not think about. A key feature that was created for vast amount of utility was the use of location-based services, where many websites inform their users that the website will be using the users' locations to enhance the functionality. However, still far too many websites do not inform their users that they may be tracked, or to what degree. In a similar juxtaposed scenario, the evolution of these social networks has allowed countless people to share photos with others online. While this seems harmless at face-value, there may be times in which people share photos of friends or other non-consenting individuals who do not want that picture viewable to anyone at the photo owner's control. There exists a lack of privacy controls for users to precisely define how they wish websites to use their location information, and for how others may share images of them online. This dissertation introduces two models that help mitigate these privacy concerns for social network users. MoveWithMe is an Android and iOS application which creates decoys that move locations along with the user in a consistent and semantically secure way. REMIND is the second model that performs rich probability calculations to determine which friends in a social network may pose a risk for privacy breaches when sharing images. Both models have undergone extensive testing to demonstrate their effectiveness and efficiency.

Chapter 1

Introduction

Smartphones have become a strong driving force in what we do every day. From scheduling out the entire day to playing games in-between classes to pass time, it is hard to imagine what they cannot do that is useful for us. Since the introduction of the smartphone, there has been an explosion in the creation of mobile applications and websites, and it shows no signs of slowing down anytime soon. These applications, or apps, and mobile websites have allowed users to connect with each other in a way that was hard to foresee before the time. Apps such as Facebook and Instagram allow users to ”friend” or ”follow” each other, which allows them to stay up-to-date with the other with what they post online, including any images they share or places that they check-in at. As we continue to be driven into a more technologically advanced society, we can easily imagine a world depicted in 1.1, in which people are connected to many things around them - constantly uploading and downloading images, menus, etc. from the cloud.

1.1 Location-Based Services

One powerful feature developed years after the introduction of the smartphone was Location Based Services (LBSs). These services allow mobile apps or websites - hereon generalized to ”mobile apps” - to tap into Application Programming Interfaces (APIs) which allow them to retrieve semantic locations, GPS coordinates, or even the current speed of users using the app or website. At first glance, this feature seems incredible - and it undoubtedly is - but it does not come without threats that many people may not

Figure 1.1: A Smart City

foresee. First of all, there are numerous mobile apps that do not inform the user that his or her location is being used, even when he/she may not want the location being used. Without this knowledge, users are left even more in the dark about privacy risks that exist when using LBSs. While some researchers and developers have tried to enhance the privacy controls users can utilize, or even create more settings for better control, the current privacy controls are too loose for mobile apps and many threats are still prevalent. For example, an adversary that learns a few locations from an unsuspecting user may learn the user's daily movement pattern, hobbies, political affiliations, and medical problems - and that is only from a simple profiling threat that can be performed just by observing a semi-coarse area that the user travels to.

1.2 Image Sharing

In a similar, but different, scenario, these mobile apps have further enabled millions of users to connect with each other by sharing images online. This online sharing allows friends of users to live vicariously through one another or help them imagine being ”in the moment.” While most of this user-generated content may be harmless and used primarily for self-recognition and gratification, there are some risks associated to them. The photos may very well depict the social circles of friends, which in itself could be a privacy breach, but also the images may explicitly disclose privacy breaches such as if an image is of a child or an easily identifiable house or location. Furthermore, friends of users that share the photos may be granted the permission to share images and greatly increase the privacy risk of the friends in the photo. Even though many of the popular mobile apps allow users to choose whom they would like to share the photos with, the re-sharing via friends to friends allow the photos to reach a wider audience than anticipated.

Figure 1.2 illustrates such a simple breach that may happen by this image re-sharing. In the example, Alice wanted to share an image with her friends, but not with Mary. But as one can see from the illustration, due to Alice's friends re-sharing her image to their friends, we observe that the image eventually reaches Mary from a couple of sources.

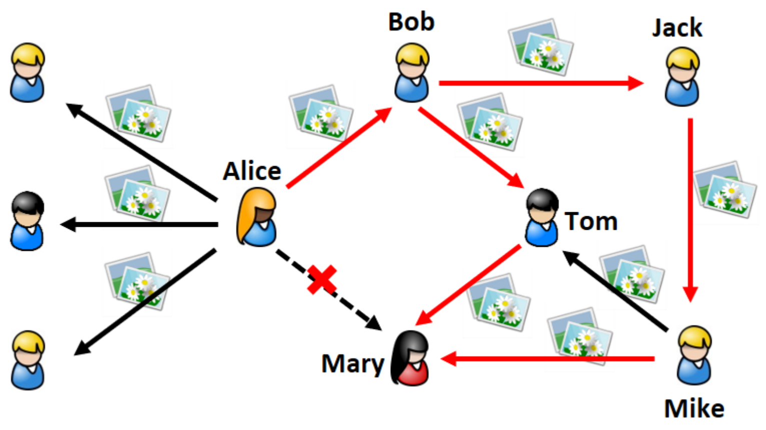


FIGURE 1.2: An Example of Privacy Breach Due to Image Propagation

Such privacy risks caused by sharing from friends to friends have been aware by many [1–4]. Some propose monitoring approaches to check the privacy violation during each sharing event [1, 2]. Most employ [5–9] clustering techniques to classify users based on their privacy preferences, profile similarities, social network topology, image content and metadata, in order to identify risky users and recommend better privacy policies. However, to the best of our knowledge, none of the existing works leverages the image sharing history and develops probability models to provide a straightforward view of the sharing consequence.

1.3 Our Proposals

Privacy breaches from image sharing share a common connection with privacy risks from mobile app LBSs in that they can be caused from the lack thereof fine-tuned privacy controls. The latter forms privacy risks from the non-existence of privacy controls that allow users to carefully define how they want their location used. The former, however, comes from non-existence of deconfliction algorithms within image sharing platforms. The privacy preferences of friends, family members, and strangers can differ - even when they are depicted in the same photo. People have a variety of personalities and preferences when it comes to sharing a photo depending on what it means to the person; having a co-owned photo of multiple people can certainly cause problems when the digital owner decides how to share that image. While in many cases it may be a courtesy that the photo owner discuss with the other parties whether to post a photo or not, but that takes more effort and time, making this route less traveled.

In this dissertation, we will discuss how we tackle such difficult problems, such as privacy risk estimation in image sharing networks, by digging into big data regarding sharing history. By analyzing and modeling the rich information of image sharing history, we build a sophisticated probability model that aggregates image disclosure probabilities along different possible image propagation chains and loops. We then present users with the direct evidence of the potential scope of sharing risk, which allows the users to make informed decisions when setting privacy preferences. This proposed system is named REMIND (Risk Estimation Mechanism for Images in Network Distribution), and would be a great add-on that could be adopted by social networking providers such as Facebook.

Many researchers have proposed several strategies to mitigate the risk of users' location privacy when visiting location-based services [10–16]. These strategies range from using access control mechanisms to control location disclosure to service providers, employing spatial cloaking or k-anonymity techniques, to generating dummy trajectories. While many of these strategies perform well under certain scenarios and conditions, they are not satisfiable for continuous location disclosure or queries from the user; that is, the user is continuously requesting location services throughout the day, not intermittently a few times a day. Even the works that generate dummy trajectories fall short due to randomly generating the dummies that can be defeated through the use of data mining techniques such as sequential pattern mining, as reported in [13].

To overcome the limitations of previous works, this dissertation will also introduce a location-privacy preservation mobile app, called MoveWithMe. This iOS and Android application generates a number of decoys that move with the user like real humans and serve as a distraction to the service providers. These decoys, however, differ from prior decoy generation algorithms in that they have their own moving patterns which include favorite places, daily schedules, social behaviors, etc. Unlike previous dummy-based approaches which only generate dummies in the nearby region and the same city where the real user is located, our decoys may be in the same city as the user, or in different cities of different countries in order to further confuse the attackers about the locations of the real user. Factors such as GPS error and changing of speed are also taken in consideration when generating the decoys. Even if adversaries were to utilize data mining tools on the decoys, they would not be able to easily identify which of the trajectories are decoys and which is the user.

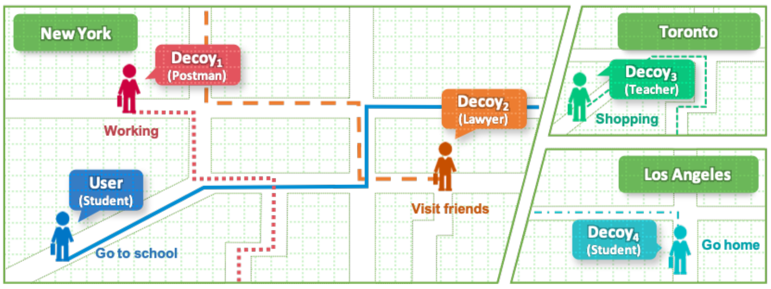


FIGURE 1.3: An Example Scenario in the MoveWithMe System

Figure 1.3 helps illustrate how MoveWithMe operates when there is a real user (say Bob) and four decoys deployed. With the activated MoveWithMe system running in the background, the decoys continuously move throughout the day as real humans. In the example, $Decoy_1$ will follow a postman's daily routine and may visit many houses, $Decoy_2$ is a lawyer who may visit clients during the day, and $Decoy_3$ and $Decoy_4$ will move based on their profiles as a teacher and a student, respectively. As time passes, when Bob has been to his research lab and a fast food restaurant, his decoys may have visited residential areas, other schools, hotels, pizza places, parks, etc. Whenever Bob visits a location-based service website such as Yelp, the MoveWithMe app will intercept Bob's request before it goes out to the location-based service, mix Bob's request with other simulated requests from the four decoys, and then send five requests altogether to Yelp. Even if Bob continuously accesses the same location-based service, the service provider will still have a hard time to discover Bob's locations out of five trajectories that demonstrate different moving patterns, jobs, social behaviors, etc.

The rest of this dissertation is organized as follows. Chapter 2 reviews background research related to LBS work, as well as image privacy work. Chapter 3 discusses our first work, MoveWithMe, including the algorithm and testing results. Chapter 4 outlines the second work, REMIND, and its' associated algorithm and testing results. Lastly, Chapter 5 concludes the dissertation and provides an overview of future work.

Chapter 2

Background Overview

Since the introduction of smartphones and boom in created mobile applications, there have been thousands of risks and flaws with the apps that went undiscovered for awhile. Upon the first instances of discovery of these risks to mobile users, researchers from all over have begun arduous work on determining what conceived the flaws and risks, as well as diving in to remedy the situations. Over the last decade, a great bound of work has been performed on Location-based Service applications and image-sharing scenarios with the intention to protect mobile users from risks unknown to them. Below discusses some of the novel work done in the respective fields, but also highlights the importance of MoveWithMe and REMIND to the social networking realm. While the works presented helped to increase knowledge in these areas and protect users from various situations, we will discuss why the works fall short of protecting users from point-of-views that we draw attention to.

2.1 Location-Based Service Related Research

Various approaches have been proposed to preserve location privacy, which can be classified into four main categories: (i) spatial-temporal cloaking based approaches; (ii) dummy-based approaches; (iii) differential privacy based approaches; and (iv) encryption-based approaches.

2.1.1 Spatial-Temporal Cloaking

The key idea of spatial-temporal cloaking is to generate a cloaking region that contains the user's real location and $k-1$ other users. In this way, the service provider would not be able to distinguish the k users in the same region and hence users achieve k-anonymity. The idea was first introduced by Gruteser et al. [17] and later has been extended by many [18–23] with different ways of generating the cloaking regions. Although these kind of approaches can hide the user's exact location, the coarse location information of the user such as the user's moving trend is still not well-protected. For example, even though the attackers cannot know the exact location of the user's home, it is still possible for them to know which city the user lives, and the approximate trajectory of the user by connecting the cloaking regions. Lin et al. [24] propose a remedy solution that transforms all the real locations to a new domain, which fully prevents the leak of the exact and continuous locations but can only support limited types of queries such as queries on the friends' locations. The main limitation of the approach is that it only supports location-based services that query on moving objects but not any static objects like restaurants. More recently, Zang and Bolot [25] propose to publish shorter trajectories at a coarse granularity to prevent attackers from correlating information obtained from call detail records with the users' true locations. However, such published trajectories will have little data utility.

2.1.2 Dummy-based Approaches

Dummy-based approaches generate dummies and send fake locations along with user's real location to the service provider so as to protect user's location privacy. For example, Niu et. al [12] propose dummy swapping and dummy selection strategies. Xue et al. [26] propose to place multiple virtual probes to pinpoint user location from fake GPS locations. Zhang et al. in [27] propose two dummy-POI selection algorithms so as to support the queries of top-k POIs. Fei et al. in [28] propose to divide users into groups, select dummies based on groups, and then share the returned results from the service provider. However, these dummies do not have continuous movement patterns which can be easily discovered by attackers who analyze dummies collected at different time stamps. As a step further, Lei et al. [13] propose two schemes to generate dummies that exhibit long-term movement patterns. Wang et al. in [15] propose a fog structure to store partial information and generate dummy trajectories. However, they did not consider the geographical constraints. As a result, the generated dummy trajectory may be off road or at places that are not accessible by real humans. Later, Hara et al. [14] added the consideration of geographical constraints during the dummy generation. Liu et al. [29] based on existing dummy generation schemes, filters out the dummies that can be identified by taking into account of the spatiotemporal correlation. Hayashida et al in [16] propose a dummy generation method which can estimate user-movement based on the visiting points inputted by the user. However, these approaches still lack the consideration of dummies' behavior rationale. Their generated dummies do not have daily routines. Such random behavior of dummies can be easily distinguished from real human trajectories by existing data mining techniques.

2.1.3 Differential Privacy

The differential privacy based approaches add noise to the users' real data so that the service providers would not know the true user locations. Andrés et al. apply Laplacian noise to location data in a discrete Cartesian plane in [30]. Users are able to adjust the level of desired privacy, which in turns increases the number of noises added to the location data. Xiao et al. [31] propose to adjust the privacy protection levels based on users' location profile and mobility history. Differential privacy is also used in [32], in which Ngo and Kim reduce the average size of cloaking regions generated by the Hilbert curve. Chen et al. propose LISA in [33], which does not rely on a trusted third party for anonymization. LISA's core algorithm is based on unobservability along with a Kalman filter to adjust noise to location data. Although these differential based approaches can obfuscate the user's locations, the noises that are added to the location data still need to be limited to ensure the service quality. That means the adversaries will still be able to know the city where the user lives, the approximate user trajectories, the time pattern of the user's daily routine, and hence be able to profile the user. Moreover, by observing non-sensitive contexts, the adversary may also be able to infer the user's sensitive information as pointed out in [34].

2.1.4 Encryption-based Approaches

The encryption-based approaches aim to fully preserve the location privacy by encrypting the location data and conducting queries directly on the encrypted data. One representative work is by Ghinita et al. [35] who propose a framework to support private nearest neighbor queries based on Private Information Retrieval (PIR). In [36], Li and Jung devise a privacy-preserving location query protocol (PLQP) in which the locations of users are shared based on a condition-matching system. Location data is encrypted using Paillier encryption to ensure that adversaries cannot intercept transmitted data. Guha et al. [37] introduce a privacy-preserving framework which provides a cloud-based matching service to return attributes and their values in an encrypted fashion. Puttaswamy et al. [38] propose to encrypt location coordinates before sharing which ensures that only designated users can decrypt the location information. Huang et al. [39] use smartphones to perform secure multi-party computation over users' location data. Wei et al. in [40] propose a system named MobiShare to support the location sharing among trusted friends and untrusted strangers while preserving user's location privacy. Combining oblivious transfer (OT) and private information retrieval, Paulet et al. [41] aim to enable efficient processing of location-dependent queries. Based on the improved homomorphic encryption, Zhu et al. [42] present a query framework in which users can query LBS results in a polygon range without leaking the information of the query polygon. These encryption-based approaches can provide a strong privacy guarantee of user's location information. However, to support the encryption-based features, the current architecture of the LBS server and client have to be significantly changed, which may not be easily deployed in the near future due to the capital cost involved.

2.1.5 Miscellaneous

Another thread of work which is related but orthogonal to our work is the query privacy preservation during continuous location-based services [43]. For example, instead of preserving location privacy, Pingley et al. propose a user-centric approach to preserve the privacy of the location-based queries. The main idea is to generate fake queries with different service attributes so that an adversary cannot associate a query with a user's ID. One limitation as reported is that it was difficult to maintain the diversity of queries with longer trajectories.

Although there have been extensive studies on location privacy theories, very few efforts have been devoted to developing real mobile apps for users to actually control their locations. Existing applications are mostly preliminary. For example, in [44], Hornyack et al. develop a system which returns a fixed location and phone number at all times. While this can ensure good privacy for the user, the user will never be able to enjoy most utilities of the location-based services. Shokri et al. [45] devise an interesting collaborative approach that allows peer users to form MobiCrowd. When a user needs to contact a location-based service, his/her request will not be directly sent to the server but be routed through the MobiCrowd. In this way, the location-based service provider will not know who sent the query. However, such strategy falls short when there are not enough users nearby. Most recently, Fawaz et al. [11] conducted a detailed risk analysis of the use of mobile apps in terms of location privacy leak. They propose an app called LP-Doctor which allows users to adjust the amount of location information to be disclosed to various apps. However, the service providers which have been granted the permission to access the locations can still track the users.

2.1.6 Summary

Compared to existing works on location-privacy-preserving mobile apps, our proposed MoveWithMe is unique in the following aspects. First, it is not constrained by the people density and can be used at any time and any place. Second, it guarantees the user experience and service quality in that the user is able to obtain the same query result without performing extra steps. Third, it introduces very little overhead as evaluated in our experiments.

It is worth noting that the initial idea of having a MoveWithMe system was first presented in our prior poster [46] which, however, has a very simple decoy generation algorithm and a simple app implementation that mainly relies on the Android platform's location mocking. In this dissertation, we have made the following significant improvement. We designed a much more sophisticated decoy generation algorithm. We developed a new app framework that is able to automatically capture and modify the data packets between users and service providers in the back end so as to automate the location mocking process which had to be done manually in our prior work. Our app can now be deployed both on Android and iOS platforms. Moreover, we conducted a whole new set of experiments including the evaluation of the use of advanced data mining techniques as an attack to our proposed system.

2.2 Image Privacy Related Research

Moving on from reviewing location-based service research, we share a view of the work done in the realm of image privacy. The work of REMIND shares similar goals of privacy protection with existing works on privacy policy recommendation systems, privacy risk estimation, and privacy violation detection in social networks. However, our proposed probability-based approach is unique in that it has not been explored in the past. More details are elaborated in the following.

2.2.1 Privacy Policy Recommendation Systems

There have been many privacy policy recommendation systems [5–8, 47, 48]. They typically utilize certain types of machine-learning algorithms to analyze users' profiles, historical privacy preferences, image content and metadata, and/or social circles, in order to predict privacy policies. Instead of relying on social circles and clustering social contexts, another thread of work looks into the image content and metadata directly [9, 49–51]. In order to even better capture the users' privacy preferences, there is a new trend of hybrid approaches which combine knowledge learned from both social contexts and the image content [52, 53]. For example, Squicciarini et al. [52] propose to utilize community practices for the cold start problem in new users and image classification based approaches for users with long privacy configuration history. Yu et al. [53] consider both content sensitiveness of the images being shared and trustworthiness of the users being granted to see the images during the fine-grained privacy settings for social image sharing.

2.2.2 Privacy Breaches from Re-sharing

Since the work of REMIND considers the privacy breach caused by friend-to-friend sharing, we review works that also examine this aspect. Li et al. [54] present a general discussion of privacy exploits, such as leakage of employment information, through friend-to-friend sharing. Akcora et al. [1] propose a risk model that estimates the risk of adding a stranger as a new friend. They cluster users based on their profile features, privacy settings and mutual friends. Our approach is different from theirs in terms of both goals and approaches. We aim to estimate the risk that an image may be seen by an unwanted person, while they aim to estimate whether a stranger could be added as a new friend. We define probability models while they use clustering techniques. Another work on malicious user identification is by Laleh et al. [2] who analyze social graphs using the assumption that malicious users show some common features on the topology of their social graphs. This work is also different from ours regarding goals and approaches. More related to our work, Kafali et al. [3] propose a privacy violation detection system called PROTOSS, which checks and predicts if the users' privacy agreements may be violated due to the friends of friends sharing. Their approach is based on semantic checking and rule reasoning. The potential limitation is that the privacy violation prediction is likely to report lots of false positives in a well-connected social network since the system presumes that the sharing would happen as long as the two users are connected in the social network. In our work, our proposed probability model not only models social network topology but also the image sharing statistics to provide more refined and accurate predictions. Later, Kökciyan et al. [4] also propose a monitoring approach which utilizes agents to keep checking whether the current sharing activity (e.g., by a friend of the owner) violates the privacy requirements of the content owner. Unlike this approach that relies on agents to continuously monitor the sharing events, our approach aims to prevent the potential privacy breach at the beginning of the sharing.

2.2.3 Privacy Scores

Similar to REMIND which provides an image disclosure probability for users, there have also been some other types of privacy scores being proposed to help enhance users' privacy awareness. Many of these privacy scores [55–57] are defined based on image sensitivities, privacy settings and users' positions in the social network. Unlike these existing privacy scores, our work calculate the privacy risk from a different angle – the image sharing history.

2.2.4 Information Diffusion

Another related area of research is information diffusion. The works on information diffusion [58–60] study how information may be propagated in the social network, finding the most influential nodes (i.e., the nodes that can distribute information to a large number of nodes) or possible reactions to information sharing. Compared to these information diffusion models which have the information propagation graph as their output, our work takes the historical information propagation as the input and then calculates the privacy disclosure risk based on that.

2.2.5 Image Policy Conflicts

Lastly, there have been works on resolving image policy conflicts among multiple users. Hu et al. [61] formulate an access control model to capture the essence of multiparty authorization requirement and employ a voting scheme for decision making when sharing photos. Such and Criado [62] propose a set of concession rules that model how users would actually negotiate to reach the common ground. Kökciyan et al. [63] propose PriArg (Privacy for Argumentation) where agents help reach sharing consensus by negotiation. Similar to PriArg, Kekulluoglu et al. [64] propose PriNego that can follow two different negotiation strategies to help agents agree to share faster. In addition, there have also been general approaches for integrating access control policies of collaborating parties [65] which, however, requires the users to clearly specify how these policies should be combined.

2.2.6 Summary

Compared to all the existing works on image privacy preservation, our work distinguishes itself in two main aspects. First, to the best of our knowledge, it is the first time that the large volume of image sharing statistic data is being considered and sophisticated probability models being built for privacy risk estimation. Second, compared to existing approaches which usually recommend policies based on relatively fuzzy logics, REMIND offers a direct and quantitative view of the risk of sharing so that the users could make more informed decisions regarding the image sharing. That is, we calculate, within a social network of varying size, the probability that different users will be able to view an image if one particular user decides to share it. In this way, it gives users an insight into just how vulnerable their photo is, and the probability someone they do not desire to view their photo will end up seeing it.

Chapter 3

Location Privacy: MoveWithMe

The goal of the MoveWithMe system is to prevent the service provider from profiling a user who is using the location-based services. The main idea is to conceal the real user's trajectory among a group of carefully generated decoys' trajectories. In what follows, we first present our threat model and then give an overview of the proposed system, followed by the detailed algorithms of each component in the system.

3.1 Threat Model

In our work, there are two main parties: (i) Location-based service providers; (ii) Smartphone users who request for location-based services.

We assume that the smartphone users connect to the Internet via certain VPN (Virtual Private Network) or anonymity network TOR (The Onion Router) so that the location-based service providers cannot use the IP address attached to the service request to pinpoint a user's location.

We consider two types of location-based service providers:

- *Precise location collectors*: Some location-based services collect users' precise location information such as the GPS coordinates or other forms of data which could be used to reveal the user's exact locations (e.g. embedded accelerometer, gyroscope, etc. [66]). For example, navigation apps need the user's exact locations to calculate the correct routes; the IoT (Internet-of-Things) device management platforms may need to know the user's precise locations to trigger certain location-based functions.

- *Coarse location collectors*: Some location-based services only need coarse location information such as the zip code. For example, weather forecasting services just need to know which city a user is located.

These adversaries may attempt to seek users' private information in the following ways:

- The adversary profiles the users' daily routines and preferences by analyzing the users' accurate locations or coarse locations collected from the users' service requests. Specifically, if the user accesses the location-based services intermittently, the adversary will obtain the user locations as disconnected spatial points on the map. If the user uses the service continuously, the adversary will obtain the user's trajectories or moving trends. In either case, the adversary can learn the time patterns of the users' movement by analyzing the timestamps associated with the location information.
- The adversary may try to link accounts of the same user in different location-based services, combine the collected location information from different accounts, and obtain a more complete trajectory information of the user.
- The adversary may exploit many tools such as advanced data mining tools and statistical tools to try to filter out the fake locations/trajectories that the users intend to use to obfuscate their true locations.

We assume that the adversaries can only passively receive location information provided by users. That means the adversaries are not able to control the user's mobile device or directly pull the user's location information without users' permissions. Our proposed approach will be robust against these attacks.

3.2 System Overview

The MoveWithMe system consists of five main components:

- **Decoy Simulator:** The decoy simulator component takes movement patterns and social profiles as inputs to generate real-time trajectories of the decoys. The decoys' trajectories also consider moving speed and possible stay time as well as GPS errors in order to mimic real human behavior as much as possible. In order to ensure the consistency of the decoys' movement and better protect user's location privacy, this component is constantly running in the background even when the real user is not using a location-based service or is not moving.

- **Request Interceptor:** When the user accesses a location-based service, the request interceptor component will analyze the request based on the pre-defined intercepting rules. Specifically, this component will first check if the request contains location information, and what type it is. Then, it will take the decoys' locations from the decoy simulator component, generate several requests for decoys, mix the simulated requests with the user's real request, and send them to the service provider altogether. Upon receiving the response from the service provider, this component will filter out the response to the decoys' requests and display only the response to the user's request. By intercepting the communication between the user's mobile phone and the service provider, this component is able to prevent the service provider from identifying the real user request.

- **Service Monitor:** When the user is accessing a location-based service, the request interceptor component will hand over the request record to the service monitor component. The service monitor component will record each location request from the service provider and notify the user about his/her location usage.

- **Location Recorder:** This component is in charge of storing both the real and fake location information in a historical trace database in order to ensure the consistency during the decoy generation and adjust the decoy profile generation parameters if needed. By analyzing the historical trajectories and with the help of Google Places API, we can find out the user's moving pattern, daily schedule, social behaviors, favorite places, etc., which are useful for generating new patterns and profiles for decoys to better meet the user's needs.

- **Trajectory Display:** This function is for the user to visualize his/her real trajectories and the decoys' trajectories so that he/she may adjust the privacy settings if needed.

Figure 4.1 gives an overview of how the components in the MoveWithMe system are cooperating with each other and interacting with location-based services. In particular, to obtain the protection from MoveWithMe, the smartphone user just needs to open the MoveWithMe app before visiting any location-based service websites. If the service monitor detects that a location-based service requires the user's phone to upload the user's location information, the MoveWithMe app will automatically send a mixed group of the real user request and the fake requests based on the decoys' locations to confuse the service provider.

The MoveWithMe app needs two permissions from the user, which are the permission to access the Internet for accessing Google Maps API, and the permission to access GPS location. Note that compared to many apps in the Google Play Store and Apple App Store, the number of permissions requested by our app is relatively minimal.

3.2.1 Decoy Pattern and Profile

In the MoveWithMe system, we model the decoys' social and travel behavior patterns and personalized profiles as follows:

Definition 1. *A decoy's social and travel behavior pattern is in the form of* $\langle PID, \mathcal{T}, \mathcal{M}, \mathcal{P}\rangle$, *which describes when and where a decoy may be and in what travel mode:*

- ***PID*** *is the unique ID of the pattern.*
- $\mathcal{T}$ *contains the types of places a decoy may visit. It is defined as a matrix* $[\langle Type_i, Rand_i, Mean_i, Dev_i\rangle]^n$, *where* $Type$ *denotes the type of a place such as*

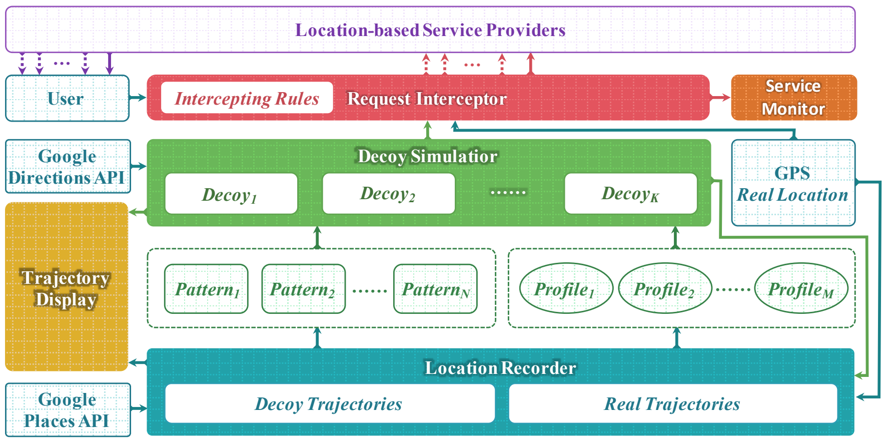


FIGURE 3.1: The Framework of the MoveWithMe System

"home", "friend's home", "university" and "restaurant", Rand indicates whether this is a fixed type (F) (e.g., home) or a randomly selected type (R) (e.g., restaurant), $Mean$ and Dev are the mean and deviation of the length of time that a decoy may stay at this type of place, and n $(n > 0)$ is the total number of place types that a decoy may visit.

- *$\mathcal{M}$ depicts the travel modes that a decoy may take under different situations. Specifically, $\mathcal{M}$ is a matrix in the form of:* $[\langle Dis\_min_i, Dis\_max_i, Pd_i, Pt_i, Pb_i, Pw_i\rangle]^m$, *where Pd, Pt, Pb, and Pw are respectively the probabilities of four travel modes (**d**riving, public **t**ransit system, **b**icycling, and **w**alking) that a decoy may take when the estimated travel distance is in the range of $[Dis\_min, Dis\_max)$, and m $(m > 0)$ denotes the total number of travel modes in this pattern.*

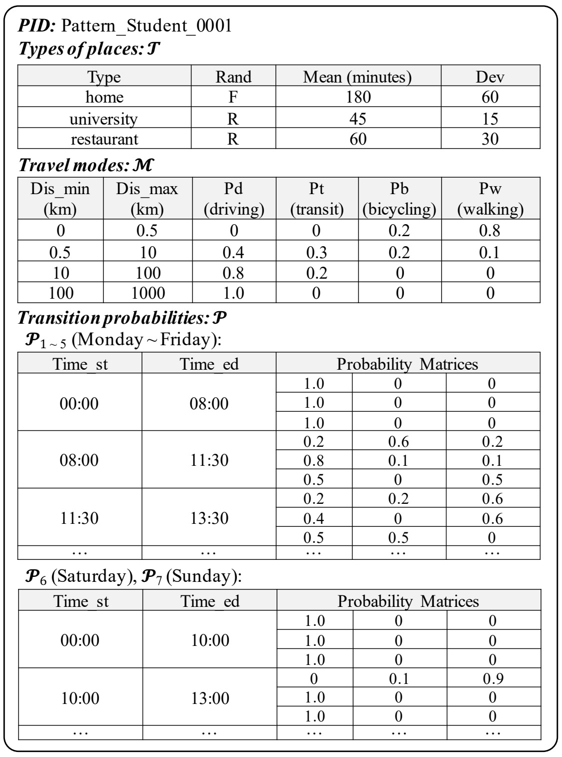
- *$\mathcal{P}$ defines the transition probabilities between different types of places in a week. For the w^{th} day in a week (let 1 to 7 denote Monday to Sunday respectively), $\mathcal{P}_w$ is a set of probability matrices in the form of $[\langle Time\_st_i, Time\_ed_i, [P_{Type_j,Type_k}]^{n\times n}\rangle]^q$, where $P_{Type_j,Type_k}$ indicates the probability that a decoy may transit to $Type_k$ when it leaves a place of $Type_j$ during the time period $[Time\_st, Time\_ed)$, n is the total number of place types, and q $(q > 0)$ is the total number of transition probability matrices.*

Definition 2. *A decoy's personalized profile is in the form of $\langle FID, \mathcal{SP}, \mathcal{MB}, \mathcal{G}\rangle$, which is an instantiation of the decoy's social and travel pattern.*

- ***FID** is the unique ID of the profile.*

- *$\mathcal{SP}$ is a set of specific places a decoy may visit. $\mathcal{SP}$ is defined as a matrix $[\langle Name_i, Type_i, Lat_i, Lng_i, [Ws_w]^7\rangle]^s$, where $Name$ is the name of the place, Type is the type of the place defined by the decoy's pattern, Lat and Lng are the latitude and longitude of the place respectively, $[Ws_w]^7$ is the weekly schedule where $\mathcal{SP}_i.Ws_w$ denotes the probability of the place i being visited by a decoy when it decides to visit $\mathcal{SP}_i.Type$ on the w^{th} day of the week, and s $(s > 0)$ is the total number of places that a decoy may visit.*

- *$\mathcal{MB}$ depicts the moving behaviors of a decoy. It is in the form of* $[\langle Mode_i, Speed\_f_i, Speed\_dev_i\rangle]^t$, *where $Mode$ is the travel mode, $Speed\_f$ is the*



PID: Pattern_Student_0001

Types of places: $\mathcal{T}$

Type	Rand	Mean (minutes)	Dev
home	F	180	60
university	R	45	15
restaurant	R	60	30

Travel modes: $\mathcal{M}$

Dis_min (km)	Dis_max (km)	Pd (driving)	Pt (transit)	Pb (bicycling)	Pw (walking)
0	0.5	0	0	0.2	0.8
0.5	10	0.4	0.3	0.2	0.1
10	100	0.8	0.2	0	0
100	1000	1.0	0	0	0

Transition probabilities: $\mathcal{P}$

$\mathcal{P}_{1\sim5}$ (Monday ~ Friday):

Time_st	Time_ed	Probability Matrices		
00:00	08:00	1.0	0	0
		1.0	0	0
		1.0	0	0
08:00	11:30	0.2	0.6	0.2
		0.8	0.1	0.1
		0.5	0	0.5
11:30	13:30	0.2	0.2	0.6
		0.4	0	0.6
		0.5	0.5	0
…	…	…	…	…

$\mathcal{P}_6$ (Saturday), $\mathcal{P}_7$ (Sunday):

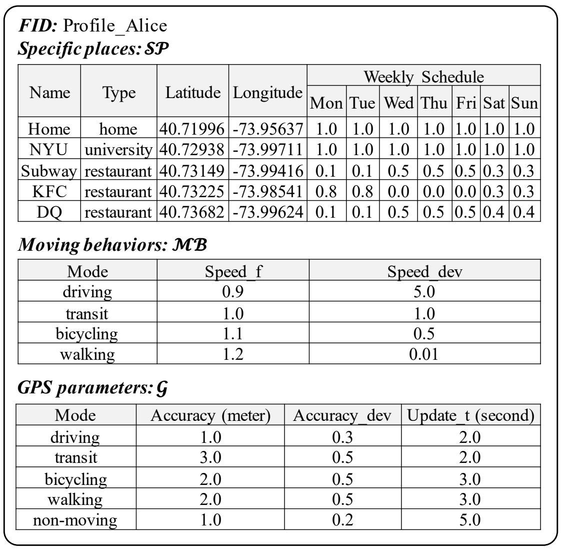
Time_st	Time_ed	Probability Matrices		
00:00	10:00	1.0	0	0
		1.0	0	0
		1.0	0	0
10:00	13:00	0	0.1	0.9
		1.0	0	0
		1.0	0	0
…	…	…	…	…

FIGURE 3.2: An Example of a Social & Travel Behavior Pattern

speed factor (Speed_f > 1.0 *means a decoy may move faster than others and vice versa), and Speed_dev depicts the velocity stability of the decoy.*

- $\mathcal{G}$ *defines the GPS parameters of a decoy under different travel modes.* $\mathcal{G}$ *is in the form of* $[\langle Mode_i, Accuracy_i, Accuracy\_dev_i, Update\_t_i\rangle]^g$, *where Mode is the travel mode (besides the travel modes defined above, we introduce a new mode non-moving so as to simulate the GPS error when a decoy is not moving), Accuracy, Accuracy_dev, and Update_t are respectively the positioning accuracy, the deviation of accuracy, and the update interval of the decoy's simulated locations.*

The decoys' patterns and profiles are used to depict human-like decoys with different behaviors. Here, the *social and travel behavior pattern* refers to a high-level description of daily activities and travel patterns of a group of people, while the *personalized profile* refers to specific places and moving behaviors of a decoy. In particular, a social and travel behavior pattern describes possibly different behaviors of a kind of people on different days of a week. For example, many people usually go to work during weekdays, but stay at home or go to the supermarket/theater on weekends. With this type of social pattern, at the same time of 14:00, a person's location may be at a company on Monday while at a supermarket on Sunday. Another example of social behavior pattern for hospital



FID: Profile_Alice

Specific places: $\mathcal{SP}$

Name	Type	Latitude	Longitude	Weekly Schedule						
				Mon	Tue	Wed	Thu	Fri	Sat	Sun
Home	home	40.71996	-73.95637	1.0	1.0	1.0	1.0	1.0	1.0	1.0
NYU	university	40.72938	-73.99711	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Subway	restaurant	40.73149	-73.99416	0.1	0.1	0.5	0.5	0.5	0.3	0.3
KFC	restaurant	40.73225	-73.98541	0.8	0.8	0.0	0.0	0.0	0.3	0.3
DQ	restaurant	40.73682	-73.99624	0.1	0.1	0.5	0.5	0.5	0.4	0.4

Moving behaviors: $\mathcal{MB}$

Mode	Speed_f	Speed_dev
driving	0.9	5.0
transit	1.0	1.0
bicycling	1.1	0.5
walking	1.2	0.01

GPS parameters: $\mathcal{G}$

Mode	Accuracy (meter)	Accuracy_dev	Update_t (second)
driving	1.0	0.3	2.0
transit	3.0	0.5	2.0
bicycling	2.0	0.5	3.0
walking	2.0	0.5	3.0
non-moving	1.0	0.2	5.0

FIGURE 3.3: An Example of a Decoy's Profile

staff may be a little different, whereby their work schedule may include night shifts and weekends. In addition, social and travel patterns may also need to include other factors such as the travel mode since some people may prefer bicycling for short distance while some may just drive all the time. By composing different patterns and profiles, we can general different kinds of decoys. Figure 3.2 shows an example of a social pattern for a decoy, and Figure 3.3 shows an example of a decoy's profile.

The decoy's patterns and profiles can be obtained via various means, such as user input, common knowledge or results from mining real trajectory datasets. In this work, we assume that a set of patterns and profiles are already been generated, and leave the pattern and profile generation in the future work.

3.2.2 Decoy Simulator

The decoy simulator component takes a set of social and travel behavior patterns and personalized profiles as the inputs and then simulates a set of corresponding decoys. For each decoy, there are several steps to simulate its movements:

3.2.2.1 Initialization

In our system, a decoy is described by a pair of pattern and profile. For example, as shown in Figures 3.2 and 3.3, the combination of "Pattern_student_0001" and "Profile_Alice" depicts a student Alice who is studying at New York University (NYU). The decoy simulator will load the pattern and profile during the initialization phase.

3.2.2.2 State Transition

We model the movements of a decoy as a set of state transitions. For example, if the decoy Alice left home, went to NYU in the morning, and went to the Subway after class at noon, the states and transitions will then be "home", "home $\rightarrow$ NYU", "NYU", "NYU $\rightarrow$ Subway", "Subway". Formally, we employ the probabilistic automaton to model the decoy's transitions among different places:

Definition 3. *A probabilistic automaton [67] is a tuple $\langle S, \Sigma, s, F, M \rangle$, which describes a machine that is in one of the finite states at any given time, and whose state changes according to the transition probabilities with respect to a sequence of input symbols:*

- $S = \{s_1, ..., s_n\}$ *defines a finite set of states.*
- Σ *denotes a finite set of input symbols.*
- s *denotes an initial state.*
- F *defines a set of designated final states.*
- M *defines the transition probability function from* $S \times \Sigma$ *to* $[0, 1]^n$.

In our system, the set of states S equals to $\mathcal{SP}$ (a set of specific places defined by the decoy's personalized profile), and the state s_i indicates that a decoy is staying at a place i. As for now, the set Σ equals to $\{\geq\}$, where "$\geq$" indicates that the stay time of a decoy at a state is larger or equal to the estimated stay time t. If a decoy stays at home at the beginning of a day, the initial state s will be home. Since we mainly use the probabilistic automaton to simulate the transition among different specific places (states) under different conditions (input symbols), the final states F in our system is set to an empty set.

Since the probability matrices in the decoy's pattern only depict the transit probabilities among different types of places, to implement the probabilistic automation, we also need to calculate the transition probabilities among a set of specific places. Let $M(i, b, j)$ denote the probability that a decoy transits from the place (state) i to the place j when the input symbol is b. Equations 3.1 and 3.2 show how to calculate $M(i, b, j)$.

$$M(i,b,j) = \begin{cases} f(i,j) & b = \text{``}\geq\text{''} \end{cases} \quad \begin{array}{r} (1 \leq i, j \leq s, \\ b \in \{\geq\}) \end{array} \tag{3.1}$$

For any i and j $(1 \leq i, j \leq s)$, we have:

$$f(i,j) = \mathcal{P}_w(t).P_{Type_i,Type_j} \times \mathcal{SP}_j.Ws_w \tag{3.2}$$

Where $Type_i$ and $Type_j$ are the types of place i and j, respectively.

Now we can use the probabilistic automaton to simulate the transitions of places. For example, if we want to simulate the decoy's movement at 10am on Friday $(w = 5)$, we first retrieve the transition probability matrix from its pattern in Figure 3.2, which is the following:

$$\mathcal{P}_5(8:00 - 11:30).\mathbf{P} = \begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.8 & 0.1 & 0.1 \\ 0.5 & 0.0 & 0.5 \end{bmatrix}$$

Given the above probability matrix which only describes the transition probability between the place types, we further calculate the transition probabilities between the five exact places (as shown in Figure 3.3) that the decoy may visit. The transition probabilities are represented as $\mathcal{M}_{\geq}$ and Figure 3.4 illustrates the probabilistic automaton constructed based on $\mathcal{M}_{\geq}$.

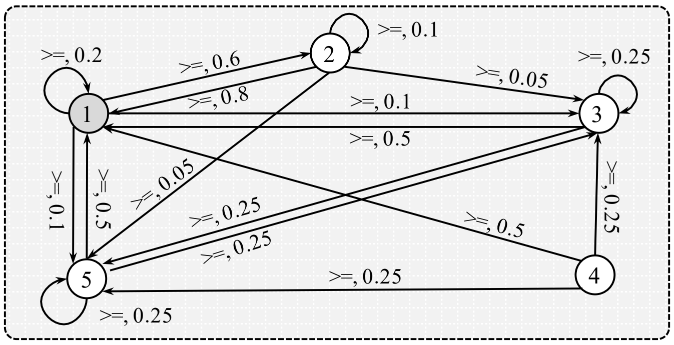


FIGURE 3.4: An Example of the State Transition Diagram

$$
\mathcal{M}_{\geq} = \begin{bmatrix} 0.2 & 0.6 & 0.1 & 0.0 & 0.1 \\ 0.8 & 0.1 & 0.05 & 0.0 & 0.05 \\ 0.5 & 0.0 & 0.25 & 0.0 & 0.25 \\ 0.5 & 0.0 & 0.25 & 0.0 & 0.25 \\ 0.5 & 0.0 & 0.25 & 0.0 & 0.25 \end{bmatrix}
$$

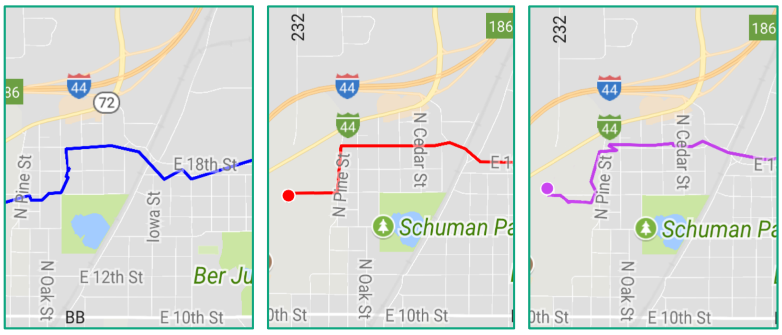
Each time the decoy arrives at a place with $Type_i$, the simulator will generate its stay time t_{stay} following the Gaussian distribution and the parameters in its pattern component $\mathcal{T}$, as shown in Equation 3.3.

$$
t_{stay} = Gaussian(\mathcal{T}_{Type_i}.mean, \mathcal{T}_{Type_i}.dev) \tag{3.3}
$$

When the decoy stays at the place equal to or longer than t_{stay}, the simulator will take "$\geq$" as the input symbol to the probabilistic automaton and find out the next place that the decoy may visit.

3.2.2.3 Movement Simulation

From the previous state transition phase, we obtain the places that the decoy will visit. The movement simulation will generate the detailed path between these places. A straightforward approach is to calculate the routes between these places (stay points) and then choose the positions along the routes. However, such an approach will result in constant moving speed and precise positions, which may be easily identified as fake trajectories by attackers. Figure 3.5 illustrates this problem. Figure 3.5 (a) is a real user's daily trajectory recorded obtained from a smartphone's GPS. Figure 3.5 (b) is a



(a) real user (b) naive approach (c) our approach

FIGURE 3.5: Comparison of Real Trajectories and Fake Trajectories

fake trajectory passing by the same stay points and is obtained using the aforementioned naive approach, from which we can see that this fake trajectory is very smooth.

In order to make the decoy's trajectory look similar to a real human's trajectory, the movement simulation takes the following steps:

Step 1 (Determine travel mode): The simulator will first calculate the distance d between the origin $place_i$ and the destination $place_j$, and then find the distance range in the travel mode matrix $\mathcal{M}$ (Figure 3.2) that satisfies $\mathcal{M}_k.dis\_min \leq d < \mathcal{M}_k.dis\_max$ or closest to d when d does not fall in any range. Next, based on the probabilities of each travel mode within this distance range, i.e. $\mathcal{M}_k.Pd$, $\mathcal{M}_k.Pt$, $\mathcal{M}_k.Pb$, and $\mathcal{M}_k.Pw$, the simulator generates a travel mode for the decoy. For example, if the distance between the decoy's current location to the next place is 4 km, the 2nd row in $\mathcal{M}$ will be selected since its distance range is between 0.5 km and 5 km. The corresponding probabilities for driving, public transit system, bicycling, and walking are 0.4, 0.3, 0.2 and 0.1, respectively, which means the decoy may be more likely to drive than to walk.

Step 2 (Obtain route): Once the travel mode m is determined, the simulator will send out a request using Google Directions API to obtain the route from $(\mathcal{SP}_i.Lat, \mathcal{SP}_i.Lng)$ to $(\mathcal{SP}_j.Lat, \mathcal{SP}_j.Lng)$ at the travel mode m. The obtained route includes the estimated travel time and a series of segments and timestamps. If no result could be returned by the Google Directions API, the simulator will try a different travel mode such as driving until getting a route.

Step 3 (Speed obfuscation): Given the length and the travel time of the segments returned by the Google Directions API, we can further calculate the moving speeds of the decoy at these segments. However, the speeds obtained in this way (denoted as $Speed_{seg_i}$

) may contain too many constant speeds for a sequence of continuous road segments, e.g., “60 km/h”, “60 km/h”, ..., “60 km/h”. This does not look like real human whose traveling speeds are never so constant. Also, constant speeds form unique patterns that can be easily caught by data mining tools. To simulate the decoy’s moving speed in a better way, we multiply Gaussian noise $\gamma_i = Gaussian(\mathcal{MB}_m.speed\_f, \mathcal{MB}_m.speed\_dev)$ to the directly computed segment speed.

$$Decoy\_Speed_{seg_i} = Speed_{seg_i} \times \gamma_i; \tag{3.4}$$

Step 4 (Geographic position obfuscation): Not only can we not use the speeds directly calculated from the optimal travel route as discussed above, we should not use the exact optimal route for the decoy either. This is because real GPS positions are never 100% accurate and trajectories formed by real GPS positions are not as smooth as the optimal route. To mimic the real human’s trajectory, we currently simulate the GPS accuracy rate based on the decoy’s travel mode m, which could be extended to a more complicated model that includes weather conditions or other factors. For each position on the optimal travel route (denoted as $pos = \langle latitude, longitude\rangle$), we first generate a GPS accuracy rate $\alpha = Gaussian(0, Gaussian(\mathcal{G}_m.accuracy, \mathcal{G}_m.accuracy\_dev))$. Then, we randomly generate an angle β and add it to the position pos:

$$\begin{aligned} Decoy.latitude_i &= pos_i.latitude + \alpha \cdot sin(\beta) \\ Decoy.longitude_i &= pos_i.longitude + \alpha \cdot cos(\beta) \end{aligned} \tag{3.5}$$

Finally, the decoy’s positions are published at an interval $\mathcal{G}_m.update\_t$ to simulate the GPS module which has different updating rates under different circumstances.

Figure 3.5 (c) shows the decoy’s trajectory obtained by our approach, where we can see that it behaves more like a real human than the optimal route in Figure 3.5 (b).

3.2.3 Request Interceptor

The request interceptor component is in charge of analyzing user’s requests, mixing the real locations and the decoys’ locations when sending out the location-based service requests to the service providers. There are four steps to realize the request interception:

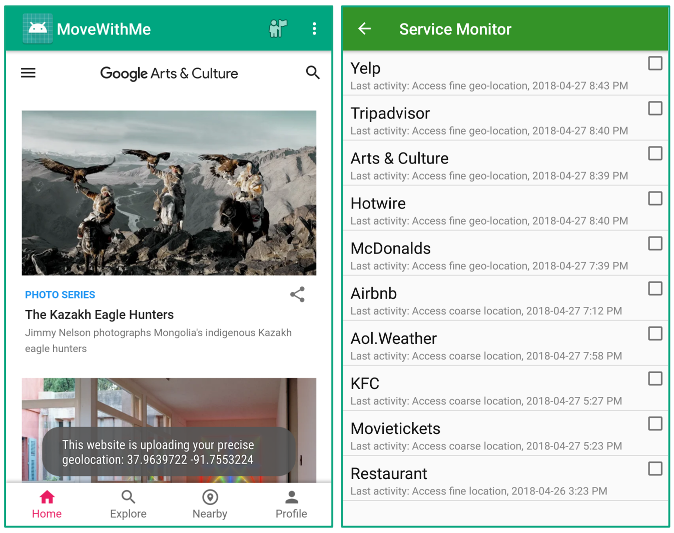
1. **Detect the location information leakage**: The interceptor checks each of the user's service request. This is achieved by intercepting the requesting URL. Specifically, we first override the function "shouldOverrideUrlLoading" of class "WebViewClient" in Android, and the function "shouldStartLoadWith" of class "UIWebView" in iOS. We define a set of regex (Regular Expressions) rules to detect if a URL contains location information. For example, if the requesting URL is "https://abc.xyz/key=ab&lat=34.123&lng=-91.456", our defined regex rule " lat=(.*?)&lng=(.*?)" will be able to extract the latitude and longitude in this URL. If it is confirmed that the requesting URL contains the user's location information, the interceptor will ask the service monitor to keep a record of the location usage and invoke the following decoys' request generation.

2. **Generate decoys' requests**: The interceptor obtains decoys' current locations from the decoy simulator, and then generates service requests for the decoys in the same form of the user's real service request. For example, if the user is looking for nearby Italian restaurants, and the requesting URL is "https://abc.xyz/food/italia n/lat=34.123&lng=-91.456", the requests from the decoys will also be looking for restaurants near the decoys' locations but the restaurants' types may be different such as Mexican restaurants. For example, the decoy's requesting URL would be "https://abc.xyz/food/mexican/lat=40.12&lng=-80.34".

3. **Send out mixed requests**: After generating the decoys' requests, the interceptor mixes them with the real user's request and send all the requests out to the service provider by calling the function "decoyWebView_i.loadUrl(decoyUrl_i)" and "mainWebView.loadUrl(userUrl)".

4. **Filter returned responses**: Upon receiving the responses from the service provider, the query results corresponding to the user's real request will be displayed in the main webview to the user. The responses that are related to the decoys' requests will return to the decoys' webviews which will be invisible to the user unless the user wants to monitor the decoys' activities and clicks a switch button in our interface to switch to the decoys' query result page.

With the support of the decoy simulator component, the request interceptor can act as a middle ware between the user and the service provider to protect user's location privacy without reducing the quality of service.



The Kazakh Eagle Hunters
Jimmy Nelson photographs Mongolia's indigenous Kazakh eagle hunters

This website is uploading your precise geolocation: 37.9639722 -91.7553224

Yelp
Last activity: Access fine geo-location, 2018-04-27 8:43 PM

Tripadvisor
Last activity: Access fine geo-location, 2018-04-27 8:40 PM

Arts & Culture
Last activity: Access fine geo-location, 2018-04-27 8:39 PM

Hotwire
Last activity: Access fine geo-location, 2018-04-27 8:40 PM

McDonalds
Last activity: Access fine geo-location, 2018-04-27 7:39 PM

Airbnb
Last activity: Access coarse location, 2018-04-27 7:12 PM

Aol.Weather
Last activity: Access coarse location, 2018-04-27 7:58 PM

KFC
Last activity: Access coarse location, 2018-04-27 5:27 PM

Movietickets
Last activity: Access coarse location, 2018-04-27 5:23 PM

Restaurant
Last activity: Access fine location, 2018-04-26 3:23 PM

(a) (b)

FIGURE 3.6: Service Monitor

3.2.4 Service Monitor

Upon receiving the detection results from the request interceptor, the service monitor component will first inform the user when his/her location information is being requested by the service provider. Meanwhile, it will keep the record of these accesses. Specifically, when the user taps the "Service Monitor" button on the phone screen, our app will display a list of the current services that require location information. For example, Figure 3.6 (a) shows the notification that the website is uploading user's precise geo-location information, and Figure 3.6 (b) shows the services a user has visited and the corresponding types of location information that have been uploaded to the service providers. The purpose of these two additional features aims to draw user's attention to the location-based services that have collected their location information, and let the users be aware of potential privacy risks that they may not notice before.

3.2.5 Location Recorder

The location recorder component is currently in charge of storing the user's real trajectories and decoys' fake trajectories.

A potential usage of these stored historical trajectories is to provide flexible and adaptive privacy protection. Specifically, the system can present a report about the similarity between the trajectories of a decoy (decoy$_i$) and the user u to the user. Based on the user's input, the decoys' social and travel patterns and profiles may be changed to reach the desired similarity, i.e., adjust decoys' profiles to make decoys perform more (or less) similar to the user. There are several ways to do the similarity calculation. For example, we can directly compare the types of places visited by a decoy with that of the real user; or we can consider also the stay time and transition time during the comparison. This feature is optional and can be turned off without affecting the other functions. Since the focus of this work was only on decoy generation and request interception, this component was simply modeled.

3.2.6 Trajectory Display

The trajectory display component aims to help users visualize their real trajectories as well as decoys' trajectories. In this sense, the users may possibly feel more secure.

We utilize the Google Maps API to display trajectories. When viewing the trace, the users' real location will be automatically placed with a blue map-marker and lines, so that users can visualize the other reported locations compared to their origins. Lines with different colors are connecting locations based on the movement from one place to the next. An example of this trajectory display feature is presented in Figure 3.8 in the experimental section.

3.3 Privacy Analysis

We now discuss the privacy protection achieved by our proposed MoveWithMe system. Recall that in the threat model (Section 3.1), we consider two types of user location information: (i) precise location such as coordinates provided by GPS; (ii) coarse location such as postal code. Service providers may utilize collected location information to learn the user's points of interests. The more precise the location is and the more frequent the user accesses the same service, the respective service provider would have more chances to infer the user's personal information such as hobbies, religions, health status, and political stance.

With the aid of our proposed MoveWithMe system, the user will be able to prevent the service provider (or attackers who compromise the server) from knowing his/her true

Table 3.1: Location-Based Services Tested

Service Providers	**Services Tested**	**Location Information**
Yelp	Search nearby bars	Geo-location
TripAdvisor	Search nearby restaurants	Geo-location
Google Arts & Culture	Search nearby museums & exhibitions	Geo-location
Hotwire	Search nearby hotels	Geo-location
McDonald's	View nearby stores	Geo-location
Airbnb	Search nearby homes	Address
Aol. Weather	View weather forecast	Address
KFC	Search nearby stores	Postal Code
Movietickets	Search nearby theaters	Postal Code

profile. This is because the user's service requests are now accompanied by a group of decoys' service requests. More importantly, these decoys have different profiles (e.g., daily schedule, personal interests) from the real user. For example, if the real user is a student, one decoy may behave like a full-time worker, another decoy may be a part-time worker. Moreover, our decoys behave like the real human so that even advanced data mining tools cannot tell which trajectory belongs to a decoy (as shown in our experiments). As a result, the service provider will receive seemly multiple users' service requests and hard to tell what are the real users' true interests.

It is worth mentioning that for the users to gain such privacy protection from the MoveWithMe system, they need to connect to the Internet via certain VPN or anonymity network TOR so that the service provider cannot identify the users' real locations by analyzing the original IP address. Also, the users should not directly use the location-based services to consume third-party services, such as reserving a restaurant through the TripAdvisor's website, which would lead the users' true locations since decoys are not allowed to purchase anything. Users are suggested to only use the location-based services to browse the needed services, and then directly go to the website of the desired service for the purchasing operations.

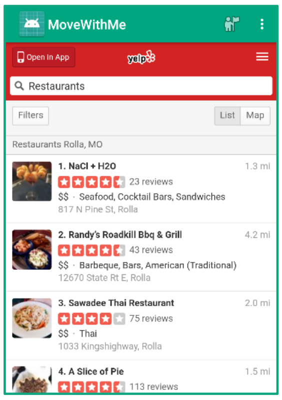
3.4 Experimental Study

We have implemented the proposed MoveWithMe system as mobile apps both in Android 6.0 and iOS 11.3, and conducted a series of experiments to evaluate the effectiveness of the system. In terms of effectiveness, we examine three aspects: (i) we tested various location-based services to see if the requests generated from decoys are also received and responded by the service providers; (ii) we check if the decoys' trajectories are consistent with the designated social patterns as time evolves; (iii) we utilize data mining tools to see if fake trajectories can be identified out of the real trajectories. The devices used for testing include a Samsung Galaxy S4, a Samsung Galaxy S6, a Google Nexus 5X, and an iPhone 7. Unless noted, the results are from Google Nexus 5X.

3.4.1 Effectiveness Testing

In the experiments, we evaluate the effectiveness of the MoveWithMe app when the user is visiting the popular location-based services as shown in Table 3.1. These websites can be classified into three categories. The first category of the websites needs the user's precise geolocation information (latitude and longitude) to perform the services, such as Yelp, TripAdvisor, and Google Arts & Culture. The second category of the websites would submit the user's address information, such as Airbnb and Aol. Weather. The third category of websites such as KFC and Movietickets use the postal code to locate user.

In the experiments, we first use our MoveWithMe app to test whether or not the above websites receive the real user's location and the decoy's locations. The real user's location is in Rolla, MO. Figure 3.7 (b), (c), and (d) shows the query results from the Yelp when the user inquiries nearby restaurants. We can see that these query results are not restaurants near the user's real location (i.e., Rolla), but the results with respect to the decoys' locations at Chicago, Kansas, and Atlanta. That means MoveWithMe has successfully fed fake locations to Yelp. Note that the requests of those decoys are performed in the background automatically. For the user, he/she will browse the Yelp website in the foreground as usual without any interruption. As presented in Figure 3.7 (a), the user will obtain the restaurant information regarding his/her real location. Similarly, in Figure 3.7 (e), the user is searching nearby things to do in Rolla using an iOS device. As presented in Figure 3.7 (f), the first decoy in Chicago is querying things to do at the same time. We observe the similar performance of MoveWithMe for other websites that provide location-based services. Due to the space constraints, we do not include the screenshots here.


MoveWithMe

Open in App | yelp | Restaurants | Filters | List | Map

Restaurants Rolla, MO

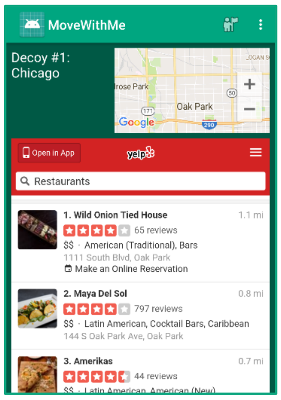
1. NaCl + H2O — 1.3 mi — 23 reviews — $$ · Seafood, Cocktail Bars, Sandwiches — 817 N Pine St, Rolla

2. Randy's Roadkill Bbq & Grill — 4.2 mi — 43 reviews — $$ · Barbeque, Bars, American (Traditional) — 12670 State Rt E, Rolla

3. Sawadee Thai Restaurant — 2.0 mi — 75 reviews — $$ · Thai — 1033 Kingshighway, Rolla

4. A Slice of Pie — 1.5 mi — 113 reviews

(a) Real user in Rolla City (Android)


MoveWithMe

Decoy #1: Chicago

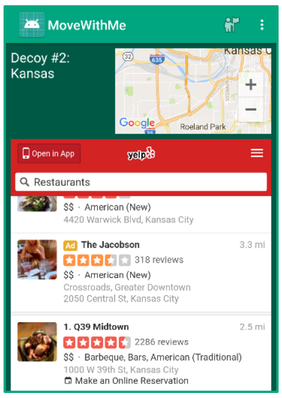
Open in App | yelp | Restaurants

1. Wild Onion Tied House — 1.1 mi — 65 reviews — $$ · American (Traditional), Bars — 1111 South Blvd, Oak Park — Make an Online Reservation

2. Maya Del Sol — 0.8 mi — 797 reviews — $$ · Latin American, Cocktail Bars, Caribbean — 144 S Oak Park Ave, Oak Park

3. Amerikas — 0.7 mi — 44 reviews — $$ · Latin American, American (New)

(b) Decoy in Chicago City (Android)


MoveWithMe

Decoy #2: Kansas

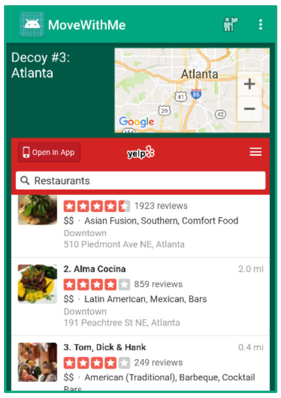
Open in App | yelp | Restaurants

$$ · American (New) — 4420 Warwick Blvd, Kansas City

Ad The Jacobson — 3.3 mi — 318 reviews — $$ · American (New) — Crossroads, Greater Downtown — 2050 Central St, Kansas City

1. Q39 Midtown — 2.5 mi — 2286 reviews — $$ · Barbeque, Bars, American (Traditional) — 1000 W 39th St, Kansas City — Make an Online Reservation

(c) Decoy in Kansas City (Android)


MoveWithMe

Decoy #3: Atlanta

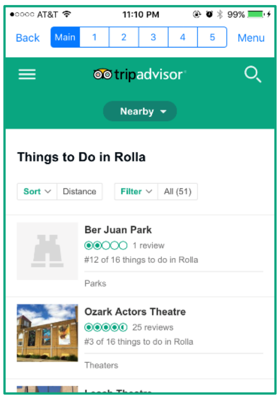
Open in App | yelp | Restaurants

1923 reviews — $$ · Asian Fusion, Southern, Comfort Food — Downtown — 510 Piedmont Ave NE, Atlanta

2. Alma Cocina — 2.0 mi — 859 reviews — $$ · Latin American, Mexican, Bars — Downtown — 191 Peachtree St NE, Atlanta

3. Tom, Dick & Hank — 0.4 mi — 249 reviews — $$ · American (Traditional), Barbeque, Cocktail Bars

(d) Decoy in Atlanta City (Android)


AT&T 11:10 PM 99%

Back | Main | 1 | 2 | 3 | 4 | 5 | Menu

tripadvisor — Nearby

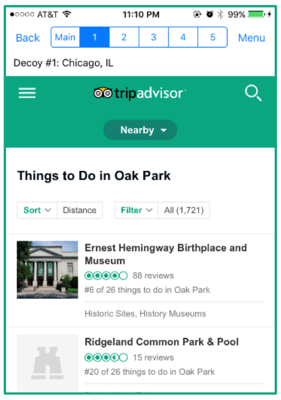
Things to Do in Rolla

Sort | Distance | Filter | All (51)

Ber Juan Park — 1 review — #12 of 16 things to do in Rolla — Parks

Ozark Actors Theatre — 25 reviews — #3 of 16 things to do in Rolla — Theaters

(e) Real user in Rolla City (iOS)


AT&T 11:10 PM 99%

Back | Main | 1 | 2 | 3 | 4 | 5 | Menu

Decoy #1: Chicago, IL

tripadvisor — Nearby

Things to Do in Oak Park

Sort | Distance | Filter | All (1,721)

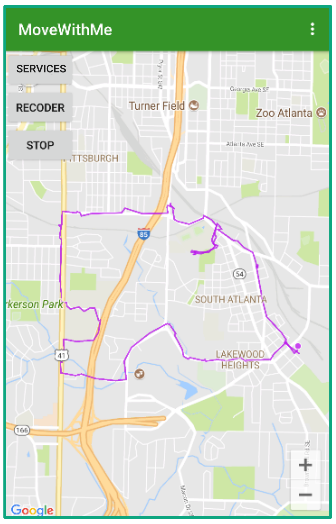
Ernest Hemingway Birthplace and Museum — 88 reviews — #6 of 26 things to do in Oak Park — Historic Sites, History Museums

Ridgeland Common Park & Pool — 15 reviews — #20 of 26 things to do in Oak Park

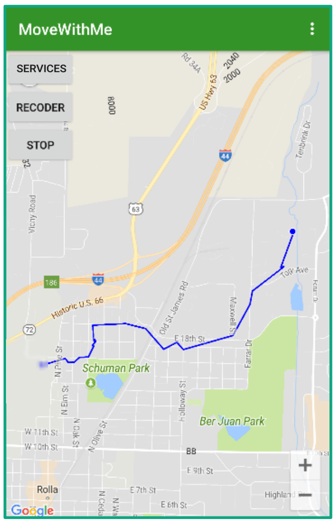
(f) Decoy in Chicago City (iOS)

FIGURE 3.7: Effectiveness Testing

Next, we ran the MoveWithMe system for a whole day and compare the historical trajectories of the real user and decoys. Figure 3.8 (a) and (b) show the results. We can see that it is hard to tell which trajectory is fake since the decoy also follows the speed limit, the human's schedule like lunch break and going back home at night. In addition, we also observe that the decoy's movement pattern is quite different from the user, which means the MoveWithMe app can effectively help prevent the service provider from profiling the user.



(a) Decoy in Atlanta



(b) Real user in Rolla

FIGURE 3.8: Historical Traces of Decoys and the Real User

After that, we test if the fake trajectories generated by the MoveWithMe app can prevent the data mining tools' detection more effectively than other randomly generated dummies. We select three commonly used data mining algorithms: *DecisionTree*, *KNN*, and *GaussionProcesses*. Each algorithm is trained by using 1000 real trajectories extracted from the GeoLife trajectory dataset [68–70] and 1000 fake trajectories from our MoveWithMe app. For each trajectory, n sample points are randomly selected to simulate the number of daily visit to the same service provider. The features used for training include various aspects of a trajectory, which are the minimum segment length, the maximum segment length, the average segment length, the minimum speed, the max speed, the average speed, and the standard deviation of speed. For comparison, we also generate another set of fake trajectories that are formed by randomly selected locations around

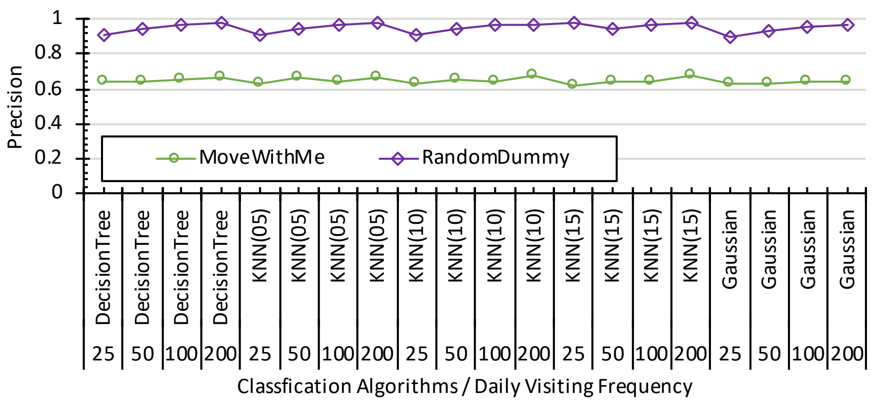


FIGURE 3.9: Trajectory Classification Precision

the real locations in the GeoLife trajectories with less than 1km distance deviation. During the testing, we mix 500 real GeoLife trajectories with 500 fake trajectories.

As shown in Figure 3.9, the detection rate of randomly generated dummy trajectories is also very high (around 95%). This is because the moving patterns of the random dummies are much different from real humans. Compared to random dummies, the decoys generated by our MoveWithMe system are much harder to be correctly classified by the data mining algorithm. The detection accuracy of our decoys is only around 60% to 70%, slightly higher than a random guess (50%). Note that this detection rate is achieved when we give the service provider advantages by assuming that they have correctly labeled 1000 of our decoy trajectories as fake trajectories during the training. When the service provider uses the random dummies for training, their ability of detecting our decoys drops to 45%.

More formally, let k be the number of decoys in the MoveWithMe app, and p be the accuracy of the real trajectory classification. We can calculate the possibility for the service provider to precisely distinguish the real trajectory and rule out the fake trajectories as $P = p^{k+1}$. For example, even if p equals to 70% while k equals to 10, the chance for the service provider to precisely distinguish the real trajectory is only 1.98%. This demonstrates that our MoveWithMe can effectively protect user's location privacy even when the service provider is trying to identify the real trajectories using advanced data mining tools. In addition, we also vary the number of daily visits to the same service provider from 25 to 200 (denoted by the number under the algorithm name in the figure). When there are fewer daily visits (e.g., 25), the detection rate is lower. The reason is straightforward that the less frequent use of the same service, the less location information the service provider will collect from the user.

3.4.2 Response Time Testing

The second round of experiments aims to evaluate the response time of the proposed MoveWithMe app. We vary the number of decoys (the value of k) from 0 to 5 ($k = 0$ means accessing location-based services without our MoveWithMe app's protection). In each instance of a run, we perform 10 different queries and record the response time for each query. Then, we calculate the average response time of the 10 queries.

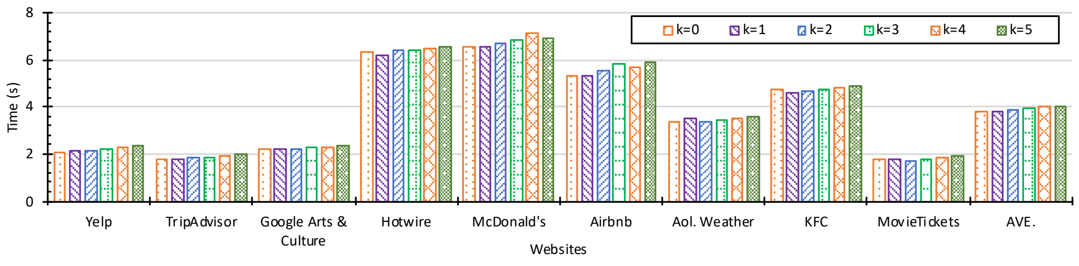


FIGURE 3.10: Response Time

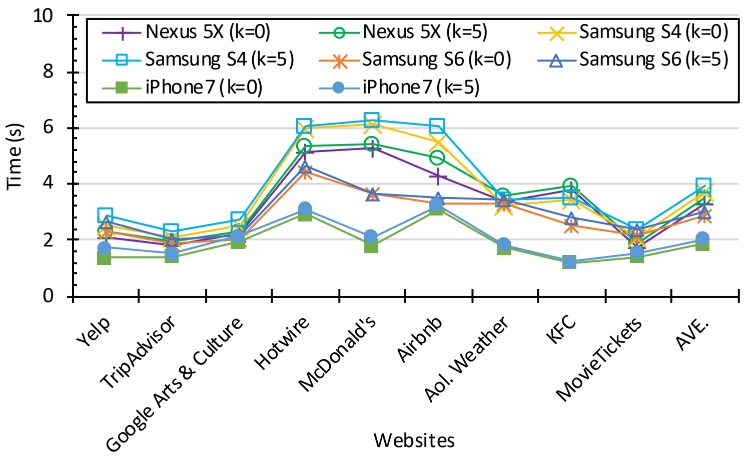


FIGURE 3.11: Response Time on Different Smartphones

Figure 3.10 reports the average response time with respect to each service and the aggregated average response time of all the services (denoted as "AVE."). From the figure, we can observe that the response time of the services slightly increases with the increase of the number of decoys. This is because the requests sent by decoys in the background take a bit of bandwidth. Since the decoys do not need to download the images and videos when sending requests, the impact on the response time is still negligible. Overall, the wait time is similar to the wait time for connecting a phone call, and hence we expect it to be acceptable for users who care about their location privacy. The minor fluctuation among the response time is mainly caused by the continuous generation of decoys' locations.

We also compare the time performance of MoveWithMe app running in four different brands of smartphones: Samsung Galaxy S4, Samsung Galaxy S6, Google Nexus 5X, and iPhone 7. Figure 3.11 shows the average response time for each service when there is 0 decoy (k=0) and 5 decoys (k=5), respectively. We can see that the MoveWithMe app incurs very little delay for all the services tested. Note that the difference of the response time among different services is mainly caused by the network condition and the service providers' servers.

3.4.3 Network Data Usage Testing

The third round of experiments aims to evaluate the network data usage. We tested two scenarios. In the first scenario, we simulate user's daily activities. For each round in the first scenario, we search nearby bars, banks and gas stations in Yelp, restaurants and 'things to do' in TripAdvisor, hotels in Hotels.com, museums in Google Arts & Culture, stores in McDonald's, and then search movies and theaters in MovieTickets. In the second scenario, we refresh the lists of coffee & tea in Yelp, restaurants in TripAdvisor, stores in McDonald's, and theaters in MovieTickets 10 times respectively, and record the data usage.

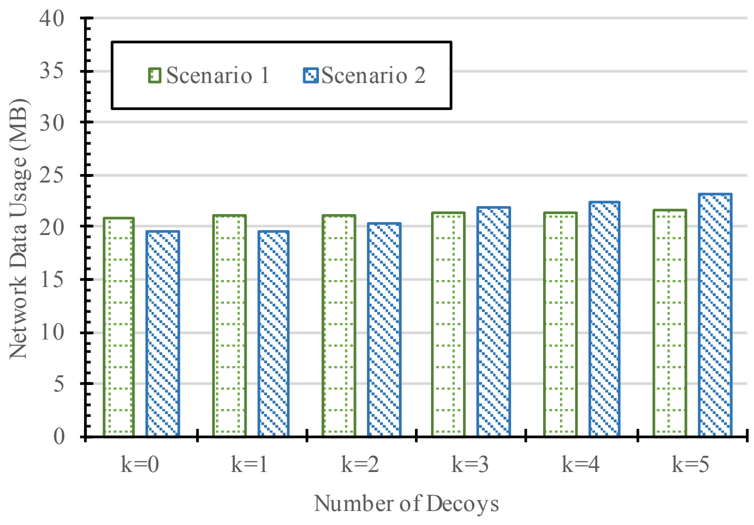


FIGURE 3.12: Network Data Usage

As presented in Figure 3.12, since the decoys need to forge multiple requests related to their locations while the user is browsing the location-based service websites at the same time, it is not surprising to see that the data usage increases with the increase of k (the number of decoys). However, the extra network data usage is very little, which is only 10% more in the first scenario and 18% more in the second scenario while k equals to 5. This is because the decoys do not need to download the images and other large files when sending requests. The service requests from decoys are mainly text contents which do not consume much bandwidth.

3.4.4 Battery Consumption Testing

In the end, we study how the MoveWithMe app affects the battery consumption of the smartphones. We tested two different scenarios. In the first scenario, we compare the total battery consumption with and without running our MoveWithMe app for a duration of 60 minutes. Note that even running in the background, our MoveWithMe app is still generating several decoys' locations continuously to prepare for the use at any time. As presented in Figure 3.13, our MoveWithMe app consumes only 0.5% more battery at the end of the 60 minutes of testing (k=5). The experimental results indicate that MoveWithMe's decoy simulation algorithm is very efficient. Recall that the decoys' profiles only need to be generated once and then the follow-up generation of fake locations is fast.

In the second scenario, we compare the battery consumption with (k>0) and without (k=0) the MoveWithMe app's protection. Specifically, in a time period of 60 minutes, we simulate 10 rounds of user access to each service, i.e., browsing all the nine service websites every 6 minutes. Figure 3.14 reports the battery consumption results. As expected, with the MoveWithMe app running, the smartphone consumes energy slightly faster than just browsing location-based websites without MoveWithMe. However, the additional battery needed for the MoveWithMe app (k=5) is less than 4%, which is almost negligible and hard to be noticed by the user. The same pattern is demonstrated for other smartphones as shown in Figure 3.15.

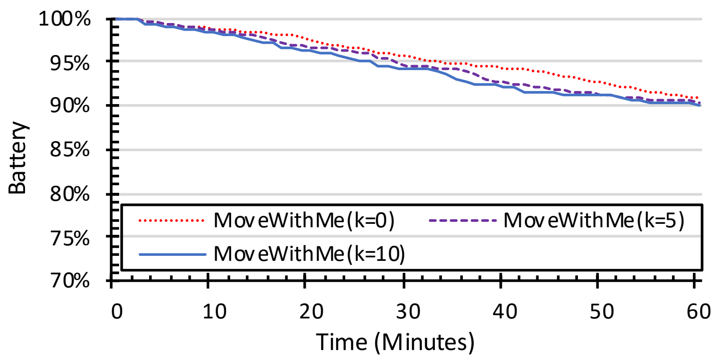


Figure 3.13: Battery Consumption (Running in the Background)

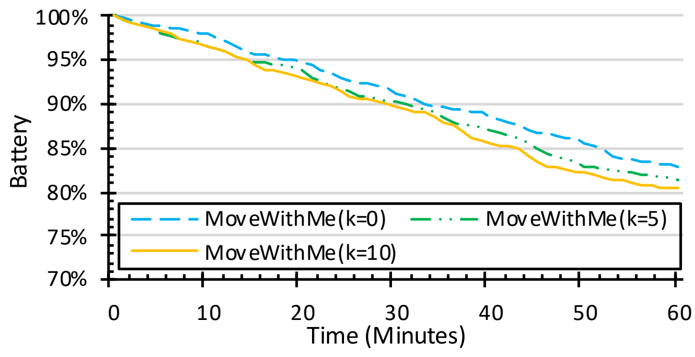


FIGURE 3.14: Battery Consumption (Browsing Websites in the Foreground)

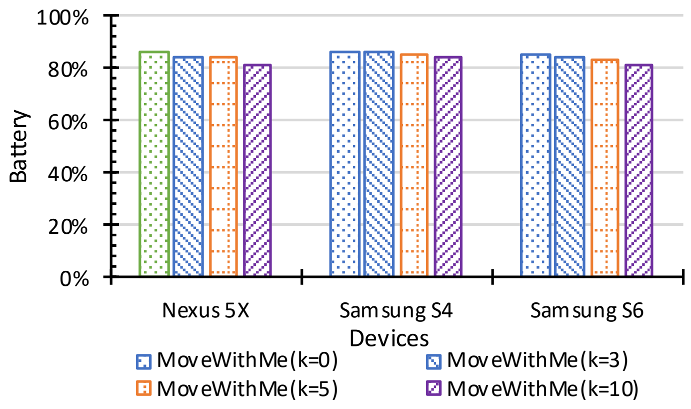


FIGURE 3.15: Battery Consumption on Different Devices

3.5 Summary

In this chapter, we presented a novel location privacy-preserving mobile app–MoveWithMe– to help smartphone users protect their location privacy when they need to frequently expose their locations to location-based services. The MoveWithMe system performs a sophisticated decoy simulation algorithm and automatically generates decoys at the runtime and sends service requests along with the user's real request to the service providers. Our proposed algorithm ensures that these decoys act consistently like real human beings as time passes, making it very hard for the service provider to identify the real user from the group of decoys and profile the real user even by using advanced data mining technologies. By evaluating the prototypes of the proposed MoveWithMe system against a variety of location-based services on various smartphones, we found that the MoveWithMe system is very effective. It is believed that by using our system, users will be able to gain greater privacy when accessing location-based services while still enjoying their full utilities.

Chapter 4

Image Privacy: REMIND

In the this chapter, we will discuss that not only are location-based services a threat to social networking users, but that there are also threats when sharing images online. Our proposed system, REMIND, will be detailed to show how it can be used to remedy online sharing threats. When it comes to social networks, the utility is abundant and allows people to be able to do things they never thought possible - but the risks are also prevalent. As discussed in the previous chapter, the risks can pertain to location-based services in which users' locations may be exploited by adversaries for some gain. In this chapter, our proposed work, REMIND, will offer a solution to the privacy risks related to online image-sharing on sites such as Facebook or Instagram.

4.1 An Exploratory User Study

This user study aims to examine the need of the REMIND system by investigating how a typical user would react if the user knows that sharing with someone may cause

Table 4.1: Questions about Uses of Online Social Networking

Question	**Options**
How many photos do you have in your social accounts?	0 to 50, 51 to 100, ..., 201 to 300, more
How many contacts do you have in your social account?	0 to 10, 11 to 50, ..., 301 to 500, more
How often do you upload photos?	Everyday, a few times a week, ..., a few times a year, rarely
Do you often designate a group of people when sharing photos?	Yes, No (I usually make my photos public)

a privacy breach with different disclosure probabilities. The user study has received the IRB approval from the university. The research is conducted in an anonymous form which means we do not record any information about the participants that could be used to identify them. We created an online survey which asked for demographic information, photo sharing preferences, and then presented users with various sharing scenarios. In what follows, we first describe the demographics information of people who participated in the user study, and then analyze the results of the users' responses.

The user study involves 114 users who are recruited online. There are 33 females and 79 males. Their ages range from 18 to over 50. The study consists of two parts. The first part collects demographics and data about online social networking habits, as shown in Table 4.1. The second part collects participants' reactions regarding privacy settings when they know the probabilities of their photos being seen by unwanted people.

From the response, we see that all of the 114 users have at least one social media account with about 71% of them have more than 2 accounts. This is consistent with current data on social network usage. When asked how often they shared images on social media, more than half of the participants confirmed that they share regularly (i.e., either a few times a week or a few times a month), and 6% admitting they share images every day. Over half of the participants estimate having shared a total of 50 to over 200 images. To understand how conscious the participants were about the privacy of their images, we asked them whether they often designate a group of people that they would like to share when uploading a photo. Although about 72% of participants answered yes, we can see there is still a significant percentage (28%) of users who simply make their personal photos public. We note that there is no statistical correlation between number of accounts or frequency of sharing with users' privacy habits, confirming that users appear unwilling (or unable) to set own privacy settings regardless of the amount of content actually disclosed online. Moreover, even users who set up the privacy configurations during the photo sharing, they still may not have the knowledge what would be the final audience of their images if their friends re-share the received images. To get an idea of how much of an impact the social network connections can make on a user's privacy, we asked how many contacts each user had in their social media accounts. 53% of the participants claim to have between 100 to 500 contacts while 21% claim to have more than 500 contacts. Imagine that even half of those contacts sharing the images they see to their own additional contacts, we can see how quickly an image can spread which may result in undesired privacy breach that the photo owners do not anticipate. Here, we note that frequency of posting is negatively correlated with amount of content posted (Pearson= -.223, $p<0.5$), but again there is no statistically significant correlation between users' frequency of managing sharing settings with number of friends.

In the second part of the study, subjects were presented with three different scenarios. In the first scenario, the photos to be shared are about the photo owner doing an extreme sport that the photo owner does not want his/her close family members to worry about. In the second scenario, the photos are about the photo owner who is doing some crazy stuff in a party and only wants to share with close friends. In the last scenario, the photos show the photo owners in funny costumes which are intended to only share with family members instead of co-workers or managers at work. The first two scenarios are designed to capture the case single-owner content decision making processes, and the last scenario is about multi-owner decision making. For each scenario, we present sample photos to the participants to help them better understand the scenarios. Then, we ask whether they would consider excluding a person from their initial sharing lists if they know there is 90%, 50%, or 10% chance that the photo may be disclosed to unwanted people by that person. Table 4.2 reports the percentage of participants who responded positively that they would change their initial privacy settings as suggested. All responses are positively correlated with a Pearson coefficient of 0.734 (case 90% and 50%) and 0.33 (50% to 10%), and $p<0.05$. From the table, we can clearly observe an increasing trend of privacy concerns when the relationship between the photo owner and the possible viewer becomes loose. Specifically, 61% of participants said they would not share with a person if there is 90% chance that the photo may be disclosed to their close family members who are not in the initial sharing list; the percentage jumps to 78% when there is 90% chance of disclosure to the photo owner's friend who is not supposed to see the photo; the percentage further increases to 85%. When it is about the disclosure to others, users tend to be less concerned about privacy within close social circles. The second observation is that the percentage of participants who agree to change the privacy settings decreases with the disclosure probability. For example, when there is only 10% chance of disclosure to undesired people, only around 30% of people chose to restrict the privacy settings in the first two scenarios. There is still a high percentage (68%) of people who would like to prevent their manager from seeing the photo even if there is only 10% chance of disclosure.

Table 4.2: User Response to Different Scenarios

Privacy Breach Probability	**90%**	**50%**	**10%**
Scenario 1 (Single-owner photo, undesired disclosure to close family member)	61%	57%	34%
Scenario 2 (Single-owner photo, undesired disclosure to friend)	78%	73%	31%
Scenario 3 (Multi-owner photo, undesired disclosure to manager)	85%	80%	68%

To gain early evidence of the potential usefulness of our REMIND system, at the end of the user study, we directly asked the participants if they would like to have such kind of privacy breach reminder provided by social websites. 75% of participants responded that they are interested in using this kind of system. More specifically, majority of the people who usually set up privacy settings (72% of all the participants) are interested in receiving privacy reminders, while a small percentage of this group of people said no, which is probably because they may think they have already configured their privacy settings very privately. Among the group of the people who claimed rarely to configure privacy settings, a small percentage of this group show interests in using the REMIND system which indicates that the users started being aware of privacy issues even through this quick user study. We feel that with the REMIND system in place in the real social networks, it will gradually help enhance public awareness in privacy problems, and eventually help people gain more privacy protections.

4.2 Problem Statement and Assumptions

Our work is developed based on the assumption that online social networking providers have full knowledge of their own social network graphs, users' profiles including privacy preferences, and their users' image sharing history.

We consider the image sharing problem in a finite social network as defined below.

Definition 4.1. (Social Network) A social network is defined as an undirected graph G(Ξ, R), where Ξ is the set of the users in this social network, and R is the set edges connecting pairs of users who have relationship with each other, i.e., $R = \{(u_i, u_j)\}$ where u_i, $u_j \in \Xi$.

Each user can specify a group of people in the same social network who are allowed to access the shared image. The privacy policy is formally defined as follows.

Definition 4.2. (Image Privacy Policy) An image privacy policy is in the form of $Pol = \{img, u, U^+\}$, where u is the image owner, and U^+ is the group of people who are allowed to access user u's image img.

Our work aims to compute the disclosure probability (as defined in Definition 4.3) that the shared image may be seen by people who are in the photo owner's contact list but are not included in the photo owner's original sharing list. The reason to focus on the users who are in the image owner's contact list is because this is the explicitly specified group of people who the image owner clearly knows whether or not to share the image with. In other words, the image owner has the greatest privacy concerns regarding the group of users if they are not included in the sharing list. For example, if Alice would like to share photos of her extreme sport activities with her college friends but not her parents as she does not want them to be worried. The photos may eventually reach a wider audience, such as other college students not specified in Alice's original sharing list, but Alice may not care about those strangers as long as her parents do not receive the photos from others.

Definition 4.3. (Image Disclosure Probability) Let u_o denote the owner of an image img, and U_o denote the set of users in u_o's contact list. Let $Pol = \{img, u_o, U_o^+\}$ denote the corresponding privacy policy for image img. The image disclosure probability $P_{u_o \Rightarrow u_t}$ is the probability that user u_o's image may be seen by a target user u_t, where $u_t \in U^-$, and $U^- = U_o/U^+$ which is the set of the users who are not in the sharing list.

4.3 The REMIND System

We propose a REMIND (Risk Estimation Mechanism for Images in Network Distribution) system that presents the image owner a privacy disclosure probability value that indicates the risk of his/her image being viewed by an unwanted person. The REMIND system not only works for photos with single owners, but can also be utilized to help resolve privacy differences in multiple users depicted in the same image. Figure 4.1 gives an overview of the data flow in the REMIND system.

First, the REMIND will identify the list of people who the image owner u_o does not want to share image with, i.e., U_o^-, by analyzing the policies associated with the image. Note

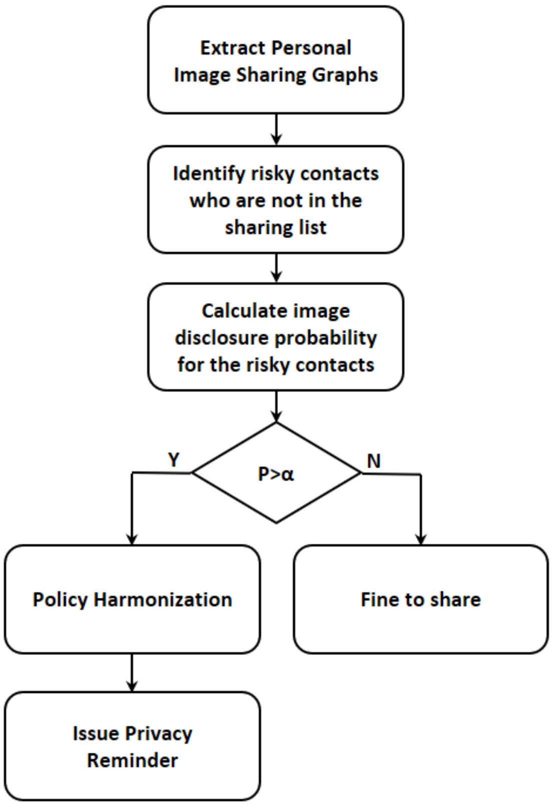

Extract Personal Image Sharing Graphs
Identify risky contacts who are not in the sharing list
Calculate image disclosure probability for the risky contacts
P>α
Y
N
Policy Harmonization
Fine to share
Issue Privacy Reminder

FIGURE 4.1: An Overview of REMIND System

that for an image with multiple users, e.g., u_{o1}, u_{o2}, ..., u_{on}, this step will return a set of $U_{o_i}^-$ whereby the $U_{o_i}^-$ is the list of people that user u_{o_i} does not want to share the photos with. The second step is to conduct the risk analysis for each user in U_o^-. We will first extract the sub-network connected to the owner(s) of the photo and then calculate the image disclosure probability for the image owner(s) with respect to the users (u_t) in U_o^-. If the computed disclosure probability $P_{u_o \Rightarrow u_t}$ is above certain threshold (e.g., 80%), the REMIND system will issue an alert to the image owner u_o regarding this. The alert will clearly indicate through which user who is in the original sharing list, user u_t may have the chance of $P_{u_o \Rightarrow u_t}$ to view the shared image. If the photo has multiple users in it, the REMIND system will conduct a policy harmonization process which combines all the alerts and suggests a possibly smaller group of users to share in to avoid undesired image disclosure. In what follows, we will elaborate the detailed algorithm for each step.

It is worth noting that the disclosure probability of an image is calculated with respect to the historical sharing information of the same category of images. This is because different types of images may have different levels of privacy concerns. For example, photos which are categorized as "funny" are more likely to propagate throughout a much larger portion of the social network than photos which are categorized as "normal daily life". To obtain categories of images, images can be easily classified based on their content [9, 51]. For the ease of illustration, the subsequent calculations and examples are referring to the images of the same category.

4.3.1 Propagation Chain Model

As aforementioned, our goal is to calculate the probability that the photo owner's contact who is not in the original sharing list may view the shared photo via friend-to-friend sharing chains. We model such sharing propagation as an image sharing graph as follows.

Definition 4.4. (Image Sharing Graph) An image sharing graph is a directed graph SG(Ξ, SR, Ψ), where Ξ is the set of users in the social network, and SR is the set of ordered pairs of users $SR = \{\langle u_i, u_j \rangle\}$ which indicates that user u_i shares some images with user u_j, Ψ is the set of detailed image sharing information including the origin of the image and the number of shares received. Specifically, $\Psi = \{\psi_{u_o:u_i \rightarrow u_j}\}$ where $\psi_{u_o:u_i \rightarrow u_j}$ denote the number of images originally owned by u_o and are shared by user u_i with u_j.

Figure 4.2 illustrates a portion of the image sharing graph in a large social network. Let us take user u_o's photo sharing propagation as an example (highlighted red in the

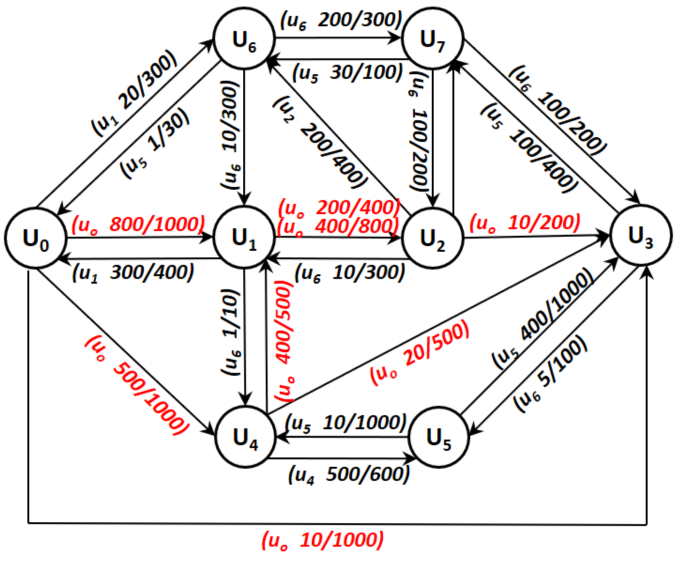


FIGURE 4.2: An Example of Image Sharing Graph

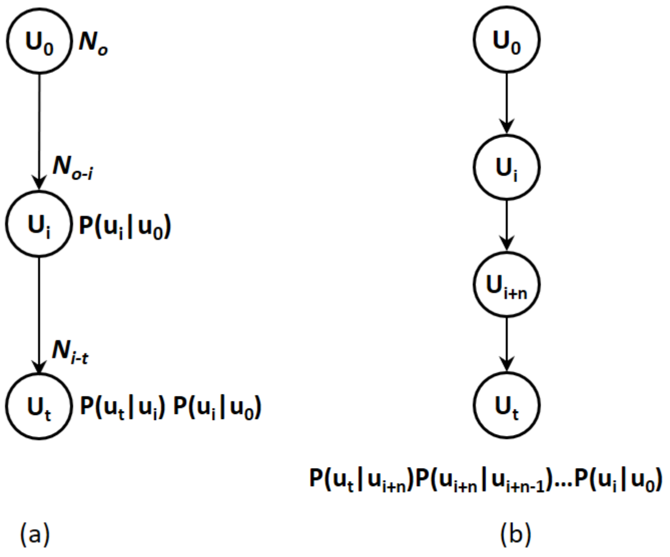


FIGURE 4.3: Single Photo Propagation Chains

figure). Assume that user u_o has 1000 photos of her own. She shares 800 out of 1000 with her contact u_1, denoted as "u_o 800/1000" on the edge from u_o to u_1. User u_o also shares 500 her own photos with user u_4 who forwards 20 of the received photos to u_3 and 400 to u_1. Now user u_1 has u_o's photos from two sources. It is possible that u_1 shares 400 photos out of the 800 shares that she directly received from u_o with u_2, and another 200 photos out of the shares that she received from u_4 with u_2 too. Correspondingly, we see two pieces of sharing information on the arrow from u_1 to u_2. Next, u_2 further shares 10 of u_o's photos from those sent by u_1 with u_3. In addition, u_o also shares 10 out of 1000 photos directly with u_3.

Based on the image sharing graph, we proceed to discuss how to compute the image disclosure probability $P_{u_o \Rightarrow u_t}$, i.e., the probability that user u_o's photo may be viewed by user u_t through the sharing propagation chains. Let us start from the simplest case (Figure 4.3(a)) when there is only one intermediate user connecting the photo owner u_o and the target user u_t. The probability $P_{u_o \Rightarrow u_t}$ can be computed by Equation 4.1.

$$P_{u_o \Rightarrow u_t} = P_{u_o \Rightarrow u_i} \cdot P_{u_o}(u_t|u_i) \tag{4.1}$$

In Equation 4.1, $P_{u_o \Rightarrow u_i}$ is the probability that u_o may share photos with u_i which can also be denoted as $P(u_i|u_o)$, and $P(u_t|u_i)$ is the probability that u_t may receive u_o's photos from u_i when u_i has u_o's photos. Specifically, let N_o denote the original number of photos that user u_o possess, let N_{o-i} denote the number of photos that user u_o shares with u_i, and let N_{i-t} denotes the number of u_o's photos that u_i further shares with u_t. $P_{u_o \Rightarrow u_i}$ can be easily computed by $\frac{N_{o-i}}{N_o}$, and $P_{u_o}(u_t|u_i)$ can be computed by $\frac{N_{i-t}}{N_{o-i}}$. Then, we have the following:

$$P_{u_o \Rightarrow u_t} = \frac{N_{o-i}}{N_o} \cdot \frac{N_{i-t}}{N_{o-i}} = \frac{N_{i-t}}{N_o}$$

Next, we extend the above case to the scenario when there are multiple users in a single chain as shown in Figure 4.3(b). The probability that u_o's photos may reach the target user u_t via multiple users (sharing routes) of sharing can be computed by Equation 4.2.

$$P_{u_o \Rightarrow u_t} = P_{u_o \Rightarrow u_i} \cdot P_{u_o}(u_t|u_{i+n}) \prod_{j=1}^{n} P_{u_o}(u_{i+j}|u_{i+j-1}) \tag{4.2}$$

At the end, we extend the probability formula to the generic scenarios (as shown in Figure 4.4) when there are multiple propagation chains between the photo owner u_o and the target user u_t. The final probability $P_{u_o \Rightarrow u_t}$ is given by Equation 4.3, where P_{c_k} denotes the sharing probability from the chain containing u_t's direct parent u_k, and m denotes the total number of sharing propagation routes.

$$\begin{aligned} P_{u_o \Rightarrow u_t} &= 1 - \prod_{k=1}^{m} (1 - P_{c_k}) \\ &= 1 - \prod_{k=1}^{m} (1 - P_{u_o \Rightarrow u_k} \cdot P(u_t|u_k) \cdot \alpha) \end{aligned} \tag{4.3}$$

In Equation 4.3, the image disclosure probability $P_{u_o \Rightarrow u_t}$ is computed by aggregating disclosure probabilities from various sharing routes. Specifically, P_{c_k} is the probability that u_t may receive u_o's photos from the propagation chain c_k. On the chain c_k, u_k is the u_t's direct sender, and hence P_{c_k} is the product of the probability $P_{u_o \Rightarrow u_k}$ that u_k receives u_o's photos and the probability $P(u_t|u_k)$ that u_k forwards the photos to u_t. Here, α is a random factor which aims to model some abnormal behavior of user u_k that u_k usually does not forward the photo to u_t suddenly decides to do so in rare cases. To achieve this, the random factor α will bring the forwarding probability $P(u_t|u_k)$ to 1 at a very low chance (i.e., 0.1%) in the probability estimation process. Next, $1 - P_{c_k}$ is the probability that u_t will not obtain u_o's photos from the chain c_k. Then, $\prod_{k=1}^{m}(1 - P_{c_k})$

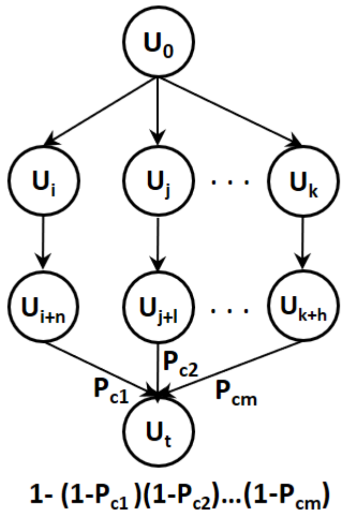


FIGURE 4.4: A Generic Photo Propagation Model

is the probability that u_t will not receive u_o's photos from any of the m propagation chains. Finally, by negating the previous probability, we obtain the probability that u_t may have access to u_o's photos.

4.3.2 Disclosure Probability Calculation

In the previous section, we have discussed how to calculate the image disclosure probability given the possibly multiple sharing routes. The next step is to identify these sharing routes in the social network. However, the real social network is very complex which may contain a huge number of paths between two users. The critical question here is: "Is it possible to compute such image disclosure probability in practice?" The answer is positive. Even though the paths connecting two users in the social network may be huge, the number of active sharing chains is not. This is based on an important observation that people's interests in sharing others' photos typically decrease as the relationship with the photo owner becomes farther away. For example, Alice shares her photo of her first surfing with her roommate Kathy. Kathy further shares the photo with her friend Mary in the same college who may also know Alice with the thought that Mary may be surprised to see Alice is doing extreme sports. It is likely that Mary may share the photo again with other friends who may also know Alice. However, the sharing is likely to stop when it reaches a person who barely knows Alice.

Based on the above observations, we can extract a sub-network that is closely related to the photo owner before the probability calculation. The sub-network is formally defined as *personal image sharing graph* in Definition 4.5.

Definition 4.5. (Personal Image Sharing Graph) Given an image sharing graph SG(Ξ, SR, Ψ), the personal image sharing graph of a user u_o is PSG(Ξ_o, SR_o, Ψ_o) which satisfies the following two conditions:

(1) $\Xi_o \subseteq \Xi$, $R_o \subseteq R$, and $\Psi_o \subseteq \Psi$;

(2) $\forall\ u_j \in \Xi_o$, $\exists \psi_{u_o:u_i \to u_j}$.

The first condition in the personal image sharing graph's definition ensures that PSG is a sub-graph of the entire image sharing graph. The second condition ensures that only the users who received photos from u_o are included in this PSG. For example, reconsider the social network shown in Figure 4.2. We can extract the personal image sharing graph for u_o as shown in Figure 4.5.

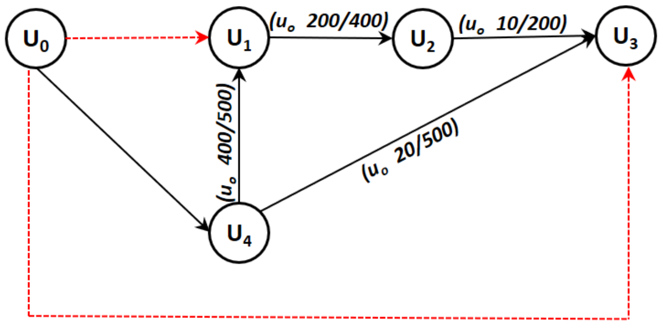


FIGURE 4.5: User u_o's Personal Image Sharing Graph

Assume that the image owner u_o shares a new photo with only u_4. The red dotted arrows in Figure 4.5 indicate that u_1 and u_3 are u_o's contacts but are not in the sharing list of this photo. We now proceed to calculate the probability that two other u_o's contacts, i.e., u_1 and u_3, may also view the image.

$P_{u_o \Rightarrow u_4} = 1$

$P_{u_o \Rightarrow u_1} = P_{u_o \Rightarrow u_4} \cdot P(u_1|u_4)) = 1 \times \frac{400}{500} = 0.8$

$P_{u_o \Rightarrow u_2} = P_{u_o \Rightarrow u_1} \times \frac{200}{400} = 0.8 \times 0.5 = 0.4$

$P_{u_o \Rightarrow u_3} = 1 - (1 - P_{u_o \Rightarrow u_2} \cdot P(u_3|u_2)) \cdot (1 - P_{u_o \Rightarrow u_4} \cdot P(u_3|u_4))$

$= 1$ - (1 - $0.4 \times \frac{10}{200}$)(1 - $1 \times \frac{20}{500}$) $= 0.048$

From the above example, we can see that even though u_o did not directly share the photo with u_1, there is still 80% chance that u_1 may view the photo shared from other channels. On the other hand, there is very little chance (5%) that u_3 may see the photo. To calculate these probabilities, the sequence of the node visit in the personal image sharing graph is important. The calculation sequence is u_1, u_2 and u_3 in the example. If we follow another computation order such as u_3, u_1 and u_2, we will obtain only part of the probability values for u_3, before u_2 is calculated. Once u_2's probability is known, we will have to adjust u_3's probability value. This is obviously inefficient especially in large-scale social networks. Therefore, we need to ensure that the parent nodes' probabilities are computed first. However, identifying the calculation order is not trivial due to the complicated interconnections among nodes in the social network that may create sharing loops. To efficiently and correctly calculate and aggregate the disclosure probabilities, we formally model the problem as the probability serialization (Definition 4.6).

Definition 4.6. (Probability Serialization) Let PSG(Ξ_o, SR_o, Ψ_o) be the personal image sharing graph of a user u_o. The probability serialization process aims to identify a serialization ordering of node visits which minimizes the node visits and ensures that each node's disclosure probability is calculated correctly. The probability serialization ordering is in the form of $u_i \succ u_{i+1} \succ \ldots \succ u_{i+k}$, where $u_i \in \Xi_o$, $\langle u_i, u_{i+1} \rangle \in SR_o$, and $u_i \succ u_{i+1}$ denotes u_i's probability will be computed before u_{i+1}'s probability.

To conduct the probability serialization, we first analyze various sharing scenarios and classify them into two main categories as shown in Figures 4.6 and 4.7, respectively. For clarity, the figures do not include the detailed sharing amounts while the arrows in the figures only indicate that there are some photos belonging to u_o being forwarded to others.

Case 1 depicts the scenario when the disclosure probability of a user needs to be calculated after all its parent nodes have been computed. Specifically, as shown in Figure 4.6, the photo owner u_o shares photos with his friend u_1 but not u_4. User u_1 then forwards some of the photos to u_2. User u_2 further shares the photos with u_3. Moreover, the three users u_1, u_2 and u_3 all forward some of the u_o's photos to user u_4. In this case, the probability that u_o's photos may be seen by u_4 depends on the the disclosure probabilities of u_1, u_2 and u_3 which need to be computed first. The appropriate calculation order of this case is $u_1 \succ u_2 \succ u_3 \succ u_4$.

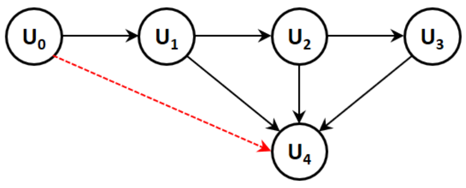


FIGURE 4.6: Sharing Scenario Case 1

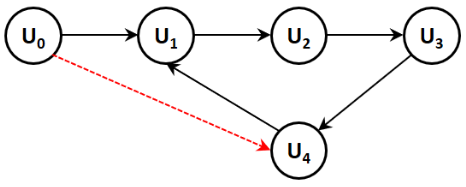


FIGURE 4.7: Sharing Scenario Case 2

Case 2 depicts the scenario when there is a sharing loop. Specifically, user u_1 forwards u_o's photos to u_2, u_2 forwards the photos to u_3, and then u_3 to u_4. Without knowing that u_1 has already seen u_o's photos, u_4 forwards the photos received from u_3 to u_1, thus creating a sharing loop. In this case, even though u_4 is also u_1's immediate parent, u_1's disclosure probability does not depend on u_4 since u_4 is sharing what u_1 originally sent out. The appropriate serialization ordering of this case is $u_1 \succ u_2 \succ u_3 \succ u_4$.

Based on the above classification, we now proceed to present a generic probability calculation algorithm. We employ two main data structures to facilitate the probability serialization. The first structure is a priority queue which stores the uncomputed nodes that have been visited so far. The second structure is a link list that stores the set of uncomputed parent nodes of each uncomputed node. The probability calculation takes the following steps (an outline of the algorithm is shown in Algorithm 1):

1. **Initialization**: Starting from the photo owner node u_o's initial sharing list, we look for the children nodes of the users in the sharing list and add them into the priority queue.

2. **Checking current node in the priority queue**: Then, we examine the node in the priority queue one by one. Let u_i denote the node in the priority queue

Algorithm 1 Probability Calculation Algorithm

```
 1: Input: Image sharing graph
 2: Output: Disclosure probabilities of $u_o$'s friends
 3: Extract $u_o$'s personal image sharing (PIS) graph
 4: for each user $u_i$ in $u_o$'s sharing list do
 5:     Initialize Prob[$u_i$]=1
 6:     Add $u_i$ to priority_queue
 7: end for
 8: while priority_queue is not empty and $U_o^-$ is not computed do
 9:     $u_i$ = priority_queue.pop()
10:     for each parent $u_j$ of $u_i$ do
11:         $P_{ij}$= chain probability (Equation 1)
12:         Prob[$u_i$]=Prob[$u_i$]*(1-$P_{ij}$)
13:     end for
14:     if all of $u_i$'s parents are computed then
15:         Prob[$u_i$]=1-Prob[$u_i$]
16:         Remove $u_i$ from priority_queue
17:     end if
18:     for each $u_i$'s direct friend $u_c$ do
19:         if $u_c$ is not in priority_queue then
20:             Add $u_c$ to priority_queue
21:         else
22:             Break_Loop between $u_i$ and $u_c$
23:         end if
24:     end for
25: end while
```

that is under consideration. For any node in the priority queue, its probability is finalized only after all its parent nodes' probabilities are computed. Therefore, we check if all of u_i's parents' probabilities have already been computed. If so, we compute the probability of u_i, remove it from the priority queue and perform the probability propagation routine. In the case that at least one parent node of u_i whose probability is not yet computed, we will just keep u_i in the priority queue. In both cases, we will proceed to perform the expansion routine for u_i.

3. **Probability propagation**: Given a node u_i whose probability is just computed, we will set the parent flags of all the nodes that take it as the parent to "computed" and calculate a partial probability for these nodes by plugging u_i's probability to Equation 4.1.

4. **Expansion**: This step is to expand the sharing chain by considering u_i's children nodes. If u_i has a child node u_c which has not been visited yet, u_c will be added to the priority queue and u_c's parents including u_i will be added to the u_c's parent

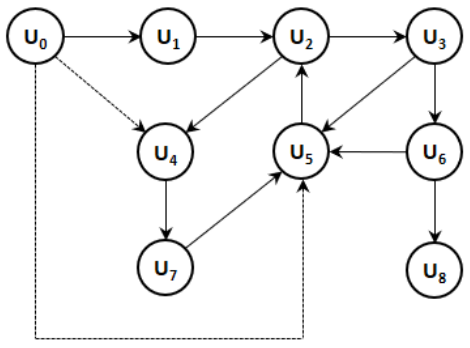


Figure 4.8: An Example of Sharing Graph

list. If some of the u_c's parents' probabilities are known, their parent flags are set to "computed". After the expansion, the algorithm goes back to the second step to check the next node in the priority queue. In the case that u_c has already been stored in the priority queue, that means a sharing loop between u_i and u_c is detected. We will then give u_i a special flag which means the loop-breaking routine is pending until there is no more new node to be added to the priority queue.

5. **Breaking the Loop between u_i and u_c**: Up to this point, all of the u_i's parents should already be in the priority queue. We will compute u_c's probability by using any partial probability that u_i has so far. Note that the partial probability that u_i possesses is definitely from sources other than u_c, so it is important to factor them into u_c's probability calculation. Once u_c's probability is computed, we will remove it from the priority queue, perform the probability propagation and then check the next node in the priority queue (i.e., go back to the second step).

To have a better understanding of the above probability calculation algorithm, let us step through the following example as shown in Figure 4.8. This example shows the image propagation from user u_o. In particular, u_o shares a new photo with u_1 but not two other friends u_4 and u_5. This example combines the two types of sharing scenarios including multi-parent relationship and multiple sharing loops.

Figure 4.9 presents how the information is updated in the priority queue and the parent lists throughout the probability calculation. The first black rows in the tables represent the priority queue at different steps, while the second rows represent the parent lists.

At the beginning, the child node (u_2) of the user (u_1) who is in the photo owner's sharing list is added to the priority queue. Since u_o shares the photo directly with u_1,

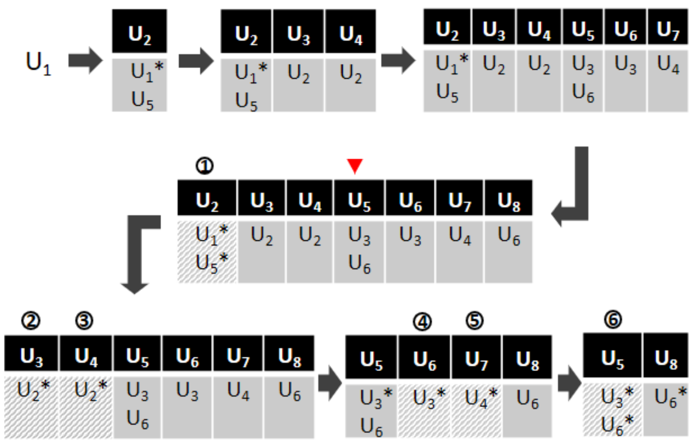


FIGURE 4.9: An Example of Probability Serialization

the probability that u_1 views the photo is 1. We start evaluating the first node in the priority queue, i.e., u_2. Since u_2 has another parent u_5 whose probability is unknown at this moment, we hold on the calculation of u_2's probability and continue expanding the sharing networks from u_2. As a result, u_2's children nodes u_3 and u_4 are added to the priority queue too. Since u_3 and u_4 also need to wait for their parent nodes to be computed, the expansion continues whereby u_3's children (i.e., u_5 and u_6) and u_4's children (i.e., u_7) are added to the priority queue. Next, we encounter the node u_5 whose child u_2 already exists in the priority queue. That means we detect a sharing loop that involves u_2 and u_5. In this case, we give u_5 a special mark indicating that we will revisit u_5 at a later time. We continue the network expansion from u_6 to its child u_8.

Up to this point, all nodes whose probabilities can be computed should have been removed from the priority queue. It is time to deal with the sharing loops. Specifically, we locate the node u_5 which has a special mark due to the sharing loop. Then, we find the node u_2 in the priority queue which has u_5 as a parent. Since the loop starts from u_2 and goes to u_5, it is not necessary to include u_5's probability during u_2's calculation. Therefore, we go ahead to calculate u_2's probability without considering u_5. Once u_2's probability is obtained, it "unlocks" its children nodes u_3 and u_4 whose probabilities are ready for calculation too. Next, we can calculate the probabilities of u_3 and u_4's children nodes which are u_6 and u_7. Finally, we can compute u_5. Note that the calculation stops here without calculating u_8 because all of u_o's contacts in the non-sharing list have been computed. The complete probability calculation ordering is $u_2 \succ u_3 \succ u_4 \succ u_6 \succ u_7 \succ u_5$ (indicated by the circled number on top of each node in the figure).

The above probability calculation algorithm provides the calculation ordering for all the users that are in the photo owner u_o's personal image sharing graph. It is worth noting that the efficiency of probability calculation can be further improved by stopping the calculation for a node if its current probability is already higher than the decision threshold. For example, if through currently explored sharing chains, the disclosure probability is as high as 99%, it is not necessary to keep checking remaining sharing routes.

The complexity of our probability calculation algorithm is O(n) as each node in the personal image sharing graph is first visited once during the personal image sharing graph extraction and then calculated once in the priority queue. It is worth noting that the probability calculation works the same for different categories of images. When multiple categories of image information is available through the image classification tool, we still just need to construct one image sharing graph for each user. The only difference will be the information stored at each node in the sharing graph. Specifically, on each node, there will be multiple tuples, each of which corresponds to a category of image sharing statistics. Since an image only belongs to one category, the calculation of a single image will only access its corresponding statistic information at the nodes, and hence there will not be any impact on the calculation efficiency.

4.3.3 Privacy Harmonization among Multiple Users

In the previous sections, we have discussed how to handle a photo with a single owner. Indeed, the risk estimation algorithm can be easily extended to address the policy harmonization issues occurring in a photo with multiple owners. It is common that different users may have different privacy preferences regarding the same photo. Consider the example when there is a group photo of Alice, Bob and Mary. Alice would like to share the photo with her family members only, while both Bob and Mary would like to share the photo with their close friends. It is possible that some of Bob and Mary's close friends are also Alice's friends who will be able to view Alice's photo although Alice's initial intention is to share only within her family. Our goal is to estimate the risk of privacy breach due to such difference. Our system will calculate the disclosure probability of the photo being seen by people who are in Alice's contact list but not her family members due to the sharing activities from Bob and Mary. We will present the estimated risk to all the photo owners so that they can refine their privacy policies.

In order to achieve the above goal, instead of calculating disclosure probabilities for an individual photo owner as discussed in the previous sections, we need to calculate the following disclosure matrix.

Definition 4.7. (Disclosure Matrix) Let u_1, ..., u_n denote the group of people depicted in a photo *img*, and Pol_1, ..., Pol_n denote the policies belonging to each photo owner, respectively. The disclosure matrix is defined below, where $u_{i_j} \in \bigcup_{w=1}^{n} U_w^+ / \{u_1, ...u_n\}$.

$$\begin{array}{cc} & \begin{array}{cccc} u_{i_1} & u_{i_2} & ... & u_{i_k} \end{array} \\ \begin{array}{c} u_1 \\ u_1 \\ ... \\ u_n \end{array} & \begin{bmatrix} P(U_1^-|u_{i_1}) & P(U_1^-|u_{i_2}) & ... & P(U_1^-|u_{i_k}) \\ P(U_2^-|u_{i_1}) & P(U_2^-|u_{i_2}) & ... & P(U_2^-|u_{i_k}) \\ ... & ... & ... & ... \\ P(U_n^-|u_{i_1}) & ... & ... & P(U_2^-|u_{i_k}) \end{bmatrix} \end{array}$$

The main idea underlying the disclosure matrix is to check the potential privacy breach that may be caused by the union of the groups of people in all the photo owners' sharing list. After the calculating the disclosure matrix, we will identify and suggest the photo owners to remove potentially high-risk sharing activities. The following is an illustrating example.

Suppose that a photo has three owners: u_1, u_2 and u_3. The sharing lists in the photo owners' policies are the following:

$$\text{Pol}_{u1} = \{u_1, u_2, u_3, u_4, u_5\}$$

$$\text{Pol}_{u2} = \{u_1, u_2, u_3, u_4, u_6\}$$

$$\text{Pol}_{u3} = \{u_1, u_2, u_3, u_5, u_7\}$$

The corresponding disclosure matrix considers the unions of the sharing list excluding the photo owners themselves who are assumed to have full access to the photo. Assume that we obtain the probabilities as shown in the following:

$$\begin{array}{c|cccc|} & u_4 & u_5 & u_6 & u_7 \\ \hline u_1 & 0.1 & \mathbf{0.9} & 0.05 & 0 \\ u_2 & 0.1 & \mathbf{0.95} & \mathbf{0.8} & 0.1 \\ u_3 & 0 & \mathbf{0.85} & 0.2 & 0.3 \end{array}$$

From the above disclosure matrix, we can see that $P(U_1^-|u_5)$, $P(U_2^-|u_5)$, and $P(U_3^-|u_5)$ are very high (i.e., above a given privacy threshold), which means the risk that people in the non-sharing lists of all the photo owners may see this photo due to the further propagation from u_5. Therefore, our REMIND system will suggest all the photo owners to remove u_5 from their sharing list. In addition, user u_2's sharing with with u_6 may cause potential privacy breach for him/herself, thus, we would suggest u_2 to remove u_6 from the sharing list. If all the users agree with suggestions, the policy harmonization will result in the following new policies:

$$\text{Pol'}_{u1} = \{u_1, u_2, u_3, u_4\}$$

$$\text{Pol'}_{u2} = \{u_1, u_2, u_3, u_4\}$$

$$\text{Pol'}_{u3} = \{u_1, u_2, u_3, u_7\}$$

4.4 Experimental Study

In this section, we present our experimental studies that evaluate both effectiveness and efficiency of our proposed approach. Specifically, we conducted user studies to see how people would react when presented a probability score of their privacy breach as computed by our system. The goal is to validate the usefulness of our proposed REMIND system. Next, we tested the performance of our system by using real social network datasets with various sharing scenarios. The second set of experiments aims to validate the efficiency of our proposed system.

4.4.1 Effectiveness Study

While we have implemented a prototype of the proposed REMIND system, we could not evaluate it in the real social network settings since its deployment in the real world requires the installation at the service provider side, e.g., installed as an additional function by Facebook or Twitter, so as to gain the access to the image sharing history. Thus, we built a simulated social network environment and conducted an A/B test as follows.

- Environment A is the one without the REMIND system which is similar to the existing social networks where people share images as usual. Specifically, a user is presented with an image and corresponding background story of the image so that the user can feel more personal about the image. Then, the user is asked to select one or more groups of users that they would like to share the image. We provide six common groups for the users to choose, which are close family members, relatives, close friends, friends, co-workers, and boss. This mimics the common practice in the real social networks.
- Environment B is the one with the REMIND function. The difference from Environment A is that after the same user chose the group of people to share, we present the probability of privacy breach (if higher than a threshold say 90%) to the user and ask if they would like to change their initial privacy settings.

We recruited another 88 participants on campus and online. There are 65 males and 23 females, 84% of whom are between 18 to 30 years old. All of the participants have at least one social media account and have experience of sharing photos.

Through the simulation, we have the following interesting findings. The adoption of the risk reminder depends on both the type of the image to be shared and the initial sharing list that the user has chosen. In general, the more sensitive the image is, the higher the chance the user will accept the REMIND system's recommendation of changing their original privacy settings. For example, the images that depict funny or crazy moments are typically considered sensitive and usually shared with close family members and friends. For such kind of images, when there is an alert about potential disclosure to the user's boss, 71.4% of the participants chose to accept the REMIND system's suggestion of removing the person who may cause this breach from the share list.

When the images are less sensitive such as vacation photos, the decisions of whether accepting the privacy alert split. Some users still care about the privacy breach to the people who are not in the original sharing list, but some do not. The decisions are now related to whom are not in the original sharing list. Participants seem to treat friends and co-workers similarly, and close friends and close family members similarly. Specifically, when the user initially shares the image with close friends, 50% of them do not care if the image may also be viewed by close family members. When the user initially shared the image with friends, 46% of them do not care if the image may also be viewed by their co-workers. In addition, there is clear impact of how close the relationship is to the photo owner. When the photo owner shares the vacation photos only with close family member, 70% of them are concerned when their photos may be viewed by their relatives who they did not include in the sharing list. When the photo owner shares their party photos with close friends, 73% of them will try to avoid privacy leakage to their co-workers.

To sum up, among total 366 privacy alerts issued in our simulated social networking environment, more than 50% are accepted by the participants, which means more 50% of privacy configurations can be improved. This shows the potential of adoption of our REMIND system in the real world.

4.4.2 Efficiency Study

We now proceed to evaluate the efficiency of our approach. Since our probability model looks into large-scale historical image sharing data and convoluted social networks, it is critical that the disclosure probability can be computed in a real-time manner to provide the users an immediate reminder when they are uploading new photos.

To examine the efficiency, we test our approach in real social networks released by Facebook and Twitter [71]. Table ?? presents the statistics of the two social networks, and Figures 4.10 and 4.11 depict the network graphs where the black dots represent users and lines represent the connections between users. We can observe that these real social networks are very complicated and nothing close to uniform distribution.

Since current social media sites only release the social network connections, but not the image sharing statistics yet, we simulate a variety of scenarios in terms of image

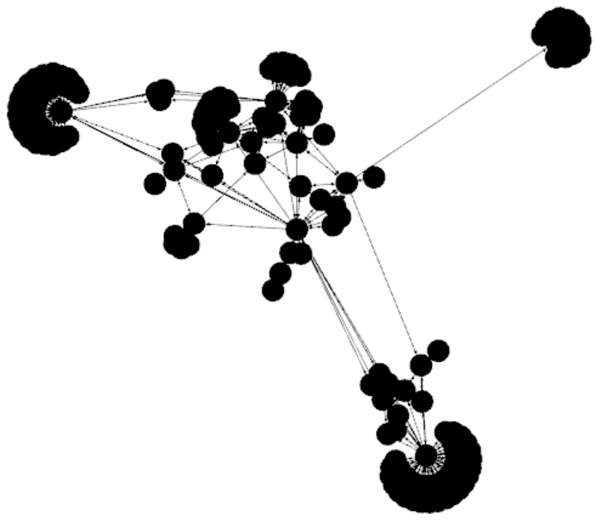


FIGURE 4.10: Facebook Network

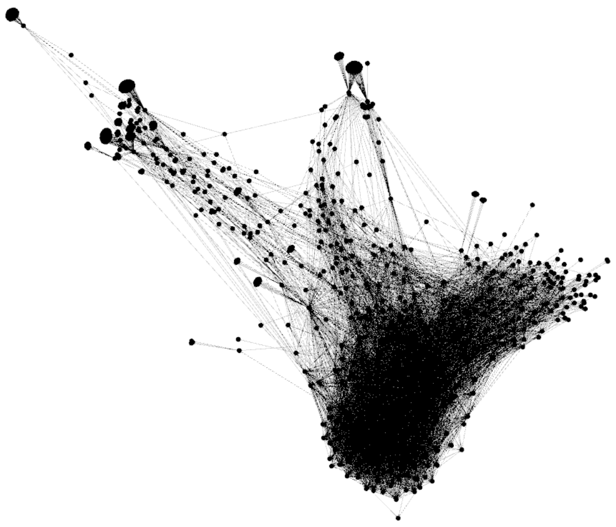


FIGURE 4.11: Twitter Network

TABLE 4.3: Real Social Network Datasets

Dataset	Facebook	Twitter
Total number of nodes	3,908	81,306
Total number of edges	168,194	1,768,149
Average degree	43	21
Maximum degree	293	1635

sharing on these real social networks as described in Definition 4.4. It is worth noting that although the image sharing statistic information is synthetic, it does not affect the efficiency test since the social network topology is real and our sharing parameters cover a wide range of possible sharing scenarios. Specifically, we first generate a random number of photos ranging from 100 to 1000 for each user. Then, for each user, we randomly select a subset of his/her friends to share certain percentage of the photos, and the size of this subset is varied in the following experiments. The receivers of the shared photo will forward a random number of received photos to a random number of their friends. In this way, the photos are propagated in the social network similar to the real world scenario. We control the propagation by setting the maximum number of hops to forward the photos since a personal photo may not be interesting to people who have almost no relationship with the photo owner. We vary the number of people in the initial sharing list. We also vary the speed of image sharing convergence as a photo may becomes less interesting to people who are farther away from the photo owner. Besides real social networks, we also test the synthetic networks with more than 20 million nodes to evaluate the scalability of our algorithm. The following subsections elaborate the detailed experimental settings for each round of experiments and report the corresponding results. All the experiments were conducted in a computer with Intel Core i7-7700K CPU (4.20 GHz) and 16GB RAM.

4.4.2.1 Effect of the Number of Propagation Hops

In the first round of experiments, we evaluate the effect of the number of image propagation hops ranging from 1 to 5. When there is only one hop, the photo owners share the photos with their direct friends and their friends will not forward the photos to anyone else. When there are five hops, the photos will be forwarded by the photo owners' friends to the friends' friends until 5 hops. The reason to choose maximum 5 hops is based on the "six degrees of separation" theory [72] that any two users can be connected through 5 acquaintances, and we choose one degree less to avoid the photos being propagated in the whole social networks which loses the privacy protection sense. Moreover, in social network, the average degree of separation is only 3.5 as reported by a study [73]. Therefore, we chose 3 hops as the default values for the subsequent tests.

Figure 4.12 reports the average time taken to compute the disclosure probability of a photo owner's friend who is not in the initial sharing list. We can observe that the calculation takes less than 1s in all cases for both the Facebook and Twitter datasets. The efficiency could be attributed to the extraction of the personal sharing graphs as well as the probability serialization algorithm, both of which help reduce the amount of users (nodes in the social network) to be examined and calculated. Moreover, we also

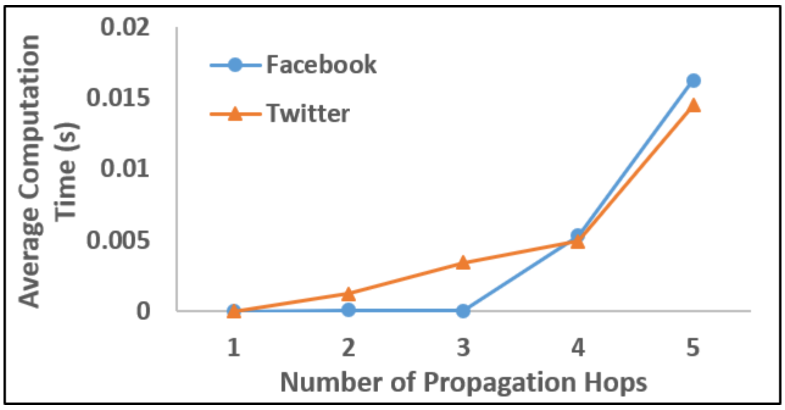


FIGURE 4.12: Effect of the Number of the Hops

observe that the calculation time increases when the photos are propagated through more hops. The reason is that the more hops, the more users may receive the shared photos, resulting in various sharing chains and loops which takes time to calculate. Actually, the average disclosure probability of the friends who are not in the initial sharing list also increases with the hops.

4.4.2.2 Effect of the Number of Friends in the Initial Sharing List

In this round of experiments, we fix the image propagation hops to 3 and vary the number of friends in the initial sharing list from 50 to 200. As shown in Figure 4.13, the average time to calculate the disclosure probability for a user in both datasets can be done in just a few milliseconds.

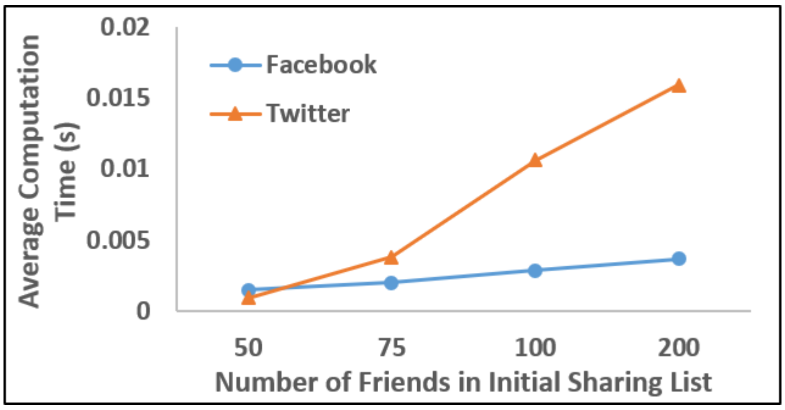


FIGURE 4.13: Effect of the Size of the Initial Sharing List

This again proves the efficiency of our algorithm. In addition, we also observe that the calculation time increases with the size of the sharing list. This is because the more people in the initial sharing list, the wider audience the photos may reach, which leads to a complicated sharing graph. As a result, there may be more ancestor nodes to be computed before finalizing a user's disclosure probability. Note that the wider audience also means the corresponding increase in the average disclosure probabilities.

4.4.2.3 Effect of the Sharing Convergence Speed

We also evaluate the effect of the sharing convergence speed. We simulate this by decreasing the number of friends to share the photos at each hop. Specifically, the statistic sharing information is generated by allowing each user to share the photos with 75 friends. For each friend who received the photo, he/she forwards the photo to a smaller number of friends, e.g., 20% less of the previous hop. The sharing stops when reaching the 3rd hop. Figure 4.14 shows the average probability calculation time for each user. Observe that the calculation time decreases when the sharing convergence speed increases. This is because the number of people in the sharing list at each hop decreases, and hence the overall size of the sharing graph decreases too. In other words, the smaller the scope of the sharing, the faster the calculation.

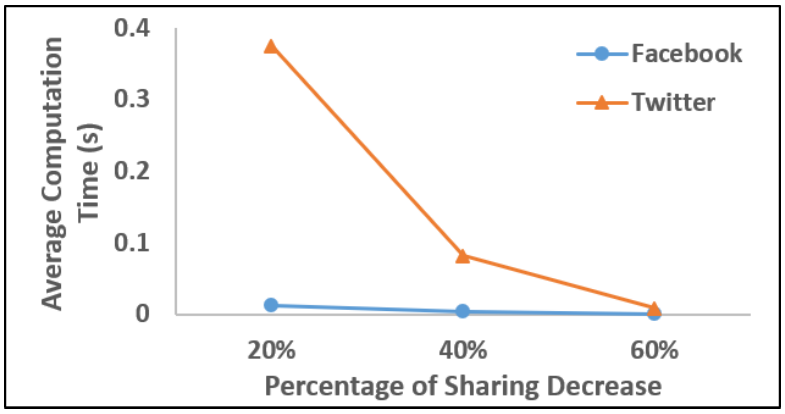


Figure 4.14: Effect of Sharing Convergence Speed

Also, the smaller the sharing scope, the lower the average disclosure probabilities. For example, let us take a closer look at the probability distribution of the Facebook dataset. When the convergence speed is decreasing by 20% per hop, there are about 80% of people in the photo owner's personal image sharing graph (including those in the initial sharing list) may see the photo with probability higher than 0.9 (denoted as "Slow convergence" in Figure 4.15); when the convergence speed is faster (i.e., 60%), the number of people with high disclosure probability drops to 50% (denoted as "Fast convergence" in Figure 4.15).

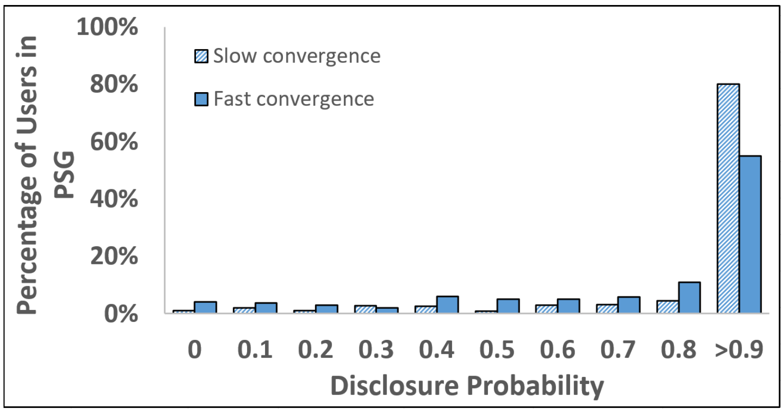


FIGURE 4.15: Probability Distribution

4.4.2.4 Large-Scale Testing

Finally, we evaluate the scalability of our proposed algorithm by using synthetically generated large-scale datasets. Figure 4.16 shows the average probability calculation time for an image when the total number of nodes in the synthetic social network increases from 1K to 20M. The number of hops for the image propagation is set to the default value 3, and each user forwards the images to 15 randomly selected friends. From the figure, we can observe that the calculation time only increases slightly with the total number of nodes in the social network. This again indicates the advantage of our proposed personal sharing graph which does not increase due to the increase of the social network size. In other words, as long as the user's contacts and image sharing behavior stay the same, the calculation scope (i.e., extracted personal sharing graph) is similar for the user no matter the user is in a small social network or a large social network. This result also demonstrates the scalability of our approach.

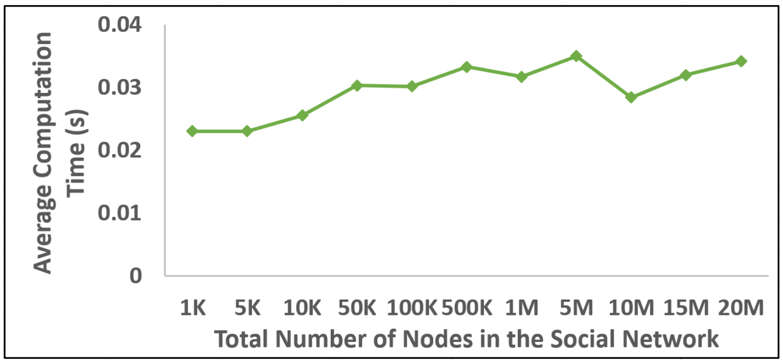


FIGURE 4.16: Effect of Total Number of Nodes in the Social Network

In addition, we also examine an extreme case when there are a small number of users with an extremely large number of contacts in the social network. To simulate this scenario, we randomly select 10,000 users from a 100K-node social network to be the contacts of the photo owner who then randomly selects 1% (100 users) of his contacts to share the images. Among the selected 1% of his contacts, we again randomly select a user to have 10,000 contacts while other contacts only have a few hundred contacts as that in Facebook. We simulate this for 3 hops of propagation and then test the calculation time. The average time to calculate the disclosure probability to a person takes just 6ms. Since the number of contacts who are not in the initial sharing list of the photo owner is large, the total time to calculate the disclosure alert for an image for the photo owner takes about 58s. The calculation time may be further shortened by considering the use of parallel computing for the multiple contacts at the same time, for which we will explore as our future work.

4.5 Summary

In this chapter, we presented a novel risk reminder system that offers the social network users a quantitative view of their image sharing risks due to friend-to-friend re-sharing. Our proposed REMIND system is based on a sophisticated probability model that models the large-scale image sharing statistic information and captures the complicated sharing propagation chains and loops. Our system also addresses the policy harmonization challenges in multi-owner photos. We have carried out performance studies to validate the effectiveness and efficiency of our approach.

Chapter 5

Conclusion

The widespread use of smartphones has enabled people across the world to interact immensely with applications that range from banking and games to air travel. Of particular interest, though, are the social networking apps. These may allow users to tap into location-based services and/or share photos to other users online. While these two specific actions may be harmless from the users' perspective, there are associated risks that hide within these uses.

This dissertation has gone over many of the common risks for LBSs, and explained in detail the risks that may occur when sharing images online. The first system discussed in the dissertation was MoveWithMe - a novel smartphone application available for Android and iOS that can help protect against attacks from apps that use LBSs. The app generates smart decoys that behave like a real human would, moving at appropriate speeds for the area, and ensuring semantic protection by neglecting to reveal the same semantic place type a user visits or even a similar pattern. The second work proposed in this dissertation was REMIND, a risk estimation mechanism for social networking sites. By quantifying the risk probability associated with sharing images to certain people, REMIND is able to show the user in a definitive way just how risky sharing to certain users may be. Then, the user can make a more knowledgeable decision whether or not to share to people. A discussed policy harmonization aspect to REMIND can also be utilized to resolve conflicts in sharing preferences between multiple owners of a single photo when sharing online, so as to help ensure that the privacy of each person in an image is protected appropriately.

In a world that is advancing every day and showing no signs of slowing down, especially in the technology realm, it is of utmost importance to continue considering privacy and security. There is abundant evidence that when privacy and security is not taken into consideration when advancing technology, the users suffer in various ways - privacy breaches, profiling, tracking, etc. While MoveWithMe and REMIND are two methods that can be used to help protect against adversarial threats, there are many more unthought of ways and research must continue to ensure users are protected and their privacy is kept.

5.1 Future Research Directions

While MoveWithMe and REMIND are nearly finished systems, there are more ways to test their effectiveness, usefulness, and efficiency. In the future, we can try fine-tuning MoveWithMe to take into consideration current traffic conditions on the decoys' routes in order to adjust travel times. By doing this, it can greatly help prevent adversaries from looking up the traffic conditions themselves in areas and filtering by that method.

For the REMIND system, a further enhancement to this can be to discuss with service providers such as Facebook or Twitter to see if collection of anonymized image sharing history is available for research. If so, this could help provide us a way to run real-world scenario tests on real data to get the most accurate results for disclosure probability calculation. Running these tests could then help us to refine REMIND to be better and add more functionalities that could address any issues not already covered.

Bibliography

[1] Cuneyt Gurcan Akcora, Barbara Carminati, and Elena Ferrari. Risks of friendships on social networks. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 810–815, 2012.

[2] N. Laleh, B. Carminati, and E. Ferrari. Graph based local risk estimation in large scale online social networks. In *IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, pages 528–535, 2015.

[3] Özgür Kafalı, Akın Günay, and Pınar Yolum. Detecting and predicting privacy violations in online social networks. *Distributed and Parallel Databases*, 32(1):161–190, Mar 2014.

[4] N. Kökciyan and P. Yolum. Priguard: A semantic approach to detect privacy violations in online social networks. *IEEE Transactions on Knowledge and Data Engineering*, 28(10):2724–2737, 2016.

[5] Joseph Bonneau, Jonathan Anderson, and George Danezis. Prying data out of a social network. In *Social Network Analysis and Mining, 2009. ASONAM'09. International Conference on Advances in*, pages 249–254. IEEE, 2009.

[6] Lujun Fang and Kristen LeFevre. Privacy wizards for social networking sites. In *Proceedings of the 19th international conference on World wide web*, pages 351–360. ACM, 2010.

[7] Alessandra Mazzia, Kristen LeFevre, and Eytan Adar. The pviz comprehension tool for social network privacy settings. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, page 13. ACM, 2012.

[8] Ruggero G. Pensa and Gianpiero Di Blasi. A semi-supervised approach to measuring user privacy in online social networks. In Toon Calders, Michelangelo Ceci, and Donato Malerba, editors, *Discovery Science*, pages 392–407, Cham, 2016.

[9] Anna Cinzia Squicciarini, Smitha Sundareswaran, Dan Lin, and Josh Wede. A3p: Adaptive policy prediction for shared images over popular content sharing sites. In *Proceedings of the 22Nd ACM Conference on Hypertext and Hypermedia*, pages 261–270, 2011.

[10] Kassem Fawaz and Kang G Shin. Location privacy protection for smartphone users. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 239–250. ACM, 2014.

[11] Kassem Fawaz, Huan Feng, and Kang G. Shin. Anatomization and protection of mobile apps' location privacy threats. In *Proceedings of the 24th USENIX Conference on Security Symposium*, SEC'15, pages 753–768, Berkeley, CA, USA, 2015. USENIX Association. ISBN 978-1-931971-232. URL `http://dl.acm.org/citation.cfm?id=2831143.2831191`.

[12] B. Niu, Q. Li, X. Zhu, G. Cao, and H. Li. Achieving k-anonymity in privacy-aware location-based services. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pages 754–762, April 2014. doi: 10.1109/INFOCOM.2014.6848002.

[13] Po-Ruey Lei, Wen-Chih Peng, Ing-Jiunn Su, and Chien-Ping Chang. Dummy-based schemes for protecting movement trajectories. *Journal of Information Science and Engineering*, 28(2):335–350, 2012.

[14] Takahiro Hara, Akiyoshi Suzuki, Mayu Iwata, Yuki Arase, and Xing Xie. Dummy-based user location anonymization under real-world constraints. *IEEE Access*, 4: 673–687, 2016.

[15] Tian Wang, Jiandian Zeng, Md Zakirul Alam Bhuiyan, Hui Tian, Yiqiao Cai, Yonghong Chen, and Bineng Zhong. Trajectory privacy preservation based on a fog structure for cloud location services. *IEEE Access*, 5:7692–7701, 2017.

[16] Shuhei Hayashida, Daichi Amagata, Takahiro Hara, and Xing Xie. Dummy generation based on user-movement estimation for location privacy protection. *IEEE Access*, 6:22958–22969, 2018.

[17] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proc. MobiSys*, pages 31–42, 2003.

[18] R. Cheng, Y. Zhang, E. Bertino, and S. Prabhakar. Preserving user location privacy in mobile data management infrastructures. In *Proc. Workshop on Privacy Enhancing Technologies*, 2006.

[19] M. F. Mokbel, C. Y. Chow, and W. G. Aref. The new casper: Query processing for location services without compromising privacy. In *Proc. VLDB*, pages 763–774, 2006.

[20] Toby Xu and Ying Cai. Feeling-based location privacy protection for location-based services. In *Proceedings of the ACM conference on Computer and communications security*, pages 348–357, 2009.

[21] M Kamenyi Domenic, Yong Wang, Fengli Zhang, Imran Memon, and Yankson H Gustav. Preserving users' privacy for continuous query services in road networks. In *International Conference on Information Management, Innovation Management and Industrial Engineering*, pages 352–355, 2013.

[22] Xiaoen Ju and Kang G Shin. Location privacy protection for smartphone users using quadtree entropy maps. *Journal of Information Privacy and Security*, 11(2): 62–79, 2015.

[23] Fizza Abbas and Heekuck Oh. A step towards user privacy while using location-based services. *JIPS*, 10(4):618–627, 2014.

[24] Dan Lin, Elisa Bertino, Reynold Cheng, and Sunil Prabhakar. Location privacy in moving-object environments. *Transactions on Data Privacy*, 2(1):21–46, 2009.

[25] Hui Zang and Jean Bolot. Anonymization of location data does not work: A large-scale measurement study. In *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking*, MobiCom '11, pages 145–156, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0492-4. doi: 10.1145/2030613.2030630. URL `http://doi.acm.org/10.1145/2030613.2030630`.

[26] Minhui Xue, Yong Liu, Keith W Ross, and Haifeng Qian. I know where you are: thwarting privacy protection in location-based social discovery services. In *IEEE Conference on Computer Communications Workshops*, pages 179–184, 2015.

[27] Hongli Zhang, Zhikai Xu, Xiangzhan Yu, and Xiaojiang Du. Lpps: Location privacy protection for smartphones. In *IEEE International Conference on Communications*, pages 1–6, 2016.

[28] Fan Fei, Shu Li, Haipeng Dai, Chunhua Hu, Wanchun Dou, and Qiang Ni. A k-anonymity based schema for location privacy preservation. *IEEE Transactions on Sustainable Computing*, 2017.

[29] Hai Liu, Xinghua Li, Hui Li, Jianfeng Ma, and Xindi Ma. Spatiotemporal correlation-aware dummy-based privacy protection scheme for location-based services. In *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*, pages 1–9. IEEE, 2017.

[30] 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 &#38; Communications Security*, CCS '13, pages 901–914, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2477-9. doi: 10.1145/2508859.2516735. URL `http://doi.acm.org/10.1145/2508859.2516735`.

[31] Qiuyu Xiao, Jiayi Chen, Le Yu, Huaxin Li, Haojin Zhu, Muyuan Li, and Kui Ren. Poster: Locmask: A location privacy protection framework in android system. In *Proceedings of the ACM SIGSAC conference on computer and communications security*, pages 1526–1528, 2014.

[32] H. Ngo and J. Kim. Location privacy via differential private perturbation of cloaking area. In *2015 IEEE 28th Computer Security Foundations Symposium*, pages 63–74, July 2015. doi: 10.1109/CSF.2015.12.

[33] Zhigang Chen, Xin Hu, Xiaoen Ju, and K. G. Shin. Lisa: Location information scrambler for privacy protection on smartphones. In *2013 IEEE Conference on Communications and Network Security (CNS)*, pages 296–304, Oct 2013. doi: 10.1109/CNS.2013.6682719.

[34] Wei Wang and Qian Zhang. A stochastic game for privacy preserving context sensing on mobile phone. In *IEEE INFOCOM*, pages 2328–2336, 2014.

[35] Gabriel Ghinita, Panos Kalnis, Ali Khoshgozaran, Cyrus Shahabi, and Kian-Lee Tan. Private queries in location based services: anonymizers are not necessary. In

Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pages 121–132. ACM, 2008.

[36] X. Y. Li and T. Jung. Search me if you can: Privacy-preserving location query service. In *2013 Proceedings IEEE INFOCOM*, pages 2760–2768, April 2013. doi: 10.1109/INFCOM.2013.6567085.

[37] Saikat Guha, Mudit Jain, and Venkata N. Padmanabhan. Koi: A location-privacy platform for smartphone apps. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, pages 14–14, Berkeley, CA, USA, 2012. USENIX Association. URL `http://dl.acm.org/citation.cfm?id=2228298.2228317`.

[38] Krishna P. N. Puttaswamy, Shiyuan Wang, Troy Steinbauer, Divyakant Agrawal, Amr El Abbadi, Christopher Kruegel, and Ben Y. Zhao. Preserving location privacy in geosocial applications. *IEEE Transactions on Mobile Computing*, 13(1):159–173, January 2014. ISSN 1536-1233. doi: 10.1109/TMC.2012.247. URL `http://dx.doi.org/10.1109/TMC.2012.247`.

[39] Yan Huang, Peter Chapman, and David Evans. Privacy-preserving applications on smartphones. In *HotSec*, 2011.

[40] Wei Wei, Fengyuan Xu, and Qun Li. Mobishare: Flexible privacy-preserving location sharing in mobile online social networks. In *Proceedings IEEE INFOCOM*, pages 2616–2620, 2012.

[41] Russell Paulet, Md. Golam Koasar, Xun Yi, and Elisa Bertino. Privacy-preserving and content-protecting location based queries. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ICDE '12, pages 44–53, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-0-7695-4747-3. doi: 10.1109/ICDE.2012.95. URL `http://dx.doi.org/10.1109/ICDE.2012.95`.

[42] Hui Zhu, Fen Liu, and Hui Li. Efficient and Privacy-Preserving Polygons Spatial Query Framework for Location-Based Services. *IEEE Internet of Things Journal*, 4:536–545, 2017.

[43] A. Pingley, N. Zhang, X. Fu, H. A. Choi, S. Subramaniam, and W. Zhao. Protection of query privacy for continuous location based services. In *2011 Proceedings IEEE INFOCOM*, pages 1710–1718, April 2011. doi: 10.1109/INFCOM.2011.5934968.

[44] Peter Hornyack, Seungyeop Han, Jaeyeon Jung, Stuart Schechter, and David Wetherall. These aren't the droids you're looking for: Retrofitting android to protect data from imperious applications. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pages 639–652, 2011. ISBN 978-1-4503-0948-6.

[45] R. Shokri, G. Theodorakopoulos, P. Papadimitratos, E. Kazemi, and J. P. Hubaux. Hiding in the mobile crowd: Locationprivacy through collaboration. *IEEE Transactions on Dependable and Secure Computing*, 11(3):266–279, May 2014. ISSN 1545-5971. doi: 10.1109/TDSC.2013.57.

[46] Douglas Steiert, Dan Lin, Quincy Conduff, and Wei Jiang. Poster: A location-privacy approach for continuous queries. In *Proceedings of the 22nd ACM on Symposium on Access Control Models and Technologies*, pages 115–117, 2017.

[47] Fabeah Adu-Oppong, Casey K Gardiner, Apu Kapadia, and Patrick P Tsang. Social circles: Tackling privacy in social networks. In *Symposium on Usable Privacy and Security (SOUPS)*, 2008.

[48] Anna Squicciarini, Sushama Karumanchi, Dan Lin, and Nicole Desisto. Identifying hidden social circles for advanced privacy configuration. *Computers and Security*, 41:40–51, 2014.

[49] Peter Klemperer, Yuan Liang, Michelle Mazurek, Manya Sleeper, Blase Ur, Lujo Bauer, Lorrie Faith Cranor, Nitin Gupta, and Michael Reiter. Tag, you can see it!: Using tags for access control in photo sharing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 377–386. ACM, 2012.

[50] Eleftherios Spyromitros-Xioufis, Symeon Papadopoulos, Adrian Popescu, and Yiannis Kompatsiaris. Personalized privacy-aware image classification. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, pages 71–78. ACM, 2016.

[51] J. Yu, B. Zhang, Z. Kuang, D. Lin, and J. Fan. iprivacy: Image privacy protection by identifying sensitive objects via deep multi-task learning. *IEEE Transactions on Information Forensics and Security*, 12(5):1005–1016, 2017.

[52] A. C. Squicciarini, D. Lin, S. Sundareswaran, and J. Wede. Privacy policy inference of user-uploaded images on content sharing sites. *IEEE Transactions on Knowledge and Data Engineering*, 27(1):193–206, 2015.

[53] J. Yu, Z. Kuang, B. Zhang, W. Zhang, D. Lin, and J. Fan. Leveraging content sensitiveness and user trustworthiness to recommend fine-grained privacy settings for social image sharing. *IEEE Transactions on Information Forensics and Security*, 13(5):1317–1332, 2018.

[54] Yan Li, Yingjiu Li, Qiang Yan, and Robert H Deng. Privacy leakage analysis in online social networks. *Computers & Security*, 49:239–254, 2015.

[55] Kun Liu and Evimaria Terzi. A framework for computing the privacy scores of users in online social networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(1):6, 2010.

[56] Agrima Srivastava and G Geethakumari. Measuring privacy leaks in online social networks. In *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 2095–2100. IEEE, 2013.

[57] Georgios Petkos, Symeon Papadopoulos, and Yiannis Kompatsiaris. Pscore: a framework for enhancing privacy awareness in online social networks. In *2015 10th International Conference on Availability, Reliability and Security*, pages 592–600. IEEE, 2015.

[58] Masahiro Kimura and Kazumi Saito. Tractable models for information diffusion in social networks. In *European conference on principles of data mining and knowledge discovery*, pages 259–271. Springer, 2006.

[59] Eytan Bakshy, Itamar Rosenn, Cameron Marlow, and Lada Adamic. The role of social networks in information diffusion. In *Proceedings of the 21st international conference on World Wide Web*, pages 519–528. ACM, 2012.

[60] Adrien Guille, Hakim Hacid, Cecile Favre, and Djamel A Zighed. Information diffusion in online social networks: A survey. *ACM Sigmod Record*, 42(2):17–28, 2013.

[61] Hongxin Hu, Gail-Joon Ahn, and Jan Jorgensen. Multiparty access control for online social networks: model and mechanisms. *IEEE Transactions on Knowledge and Data Engineering*, 25(7):1614–1627, 2013.

[62] Jose M Such and Natalia Criado. Resolving multi-party privacy conflicts in social media. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1851–1863, 2016.

[63] Nadin Kökciyan, Nefise Yaglikci, and Pinar Yolum. An argumentation approach for resolving privacy disputes in online social networks. *ACM Transactions on Internet Technology (TOIT)*, 17(3):27, 2017.

[64] Dilara Kekulluoglu, Nadin Kokciyan, and Pinar Yolum. Preserving privacy as social responsibility in online social networks. *ACM Transactions on Internet Technology (TOIT)*, 18(4):42, 2018.

[65] Prathima Rao, Dan Lin, Elisa Bertino, Ninghui Li, and Jorge Lobo. Fine-grained integration of access control policies. *Computers and Security*, 30(2):91 – 107, 2011. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2010.10.006. URL `http://www.sciencedirect.com/science/article/pii/S0167404810000891`. Special Issue on Access Control Methods and Technologies.

[66] Yi Liang, Zhipeng Cai, Qilong Han, and Yingshu Li. Location privacy leakage through sensory data. *Security and Communication Networks*, 2017, 2017.

[67] Michael O Rabin. Probabilistic automata. *Information and control*, 6(3):230–245, 1963.

[68] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. Mining interesting locations and travel sequences from gps trajectories. In *Proceedings of the 18th international conference on World wide web*, pages 791–800. ACM, 2009.

[69] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. Understanding mobility based on gps data. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 312–321. ACM, 2008.

[70] Yu Zheng, Xing Xie, and Wei-Ying Ma. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.*, 33(2):32–39, 2010.

[71] Jure Leskovec. Stanford large network dataset collection. In *https://snap.stanford.edu/data/*.

[72] Wikipedia. Six degree of separation. In *https://en.wikipedia.org/wiki/Six_degrees_of_separation*.

[73] Smriti Bhagat, Moira Burke, Carlos Diuk, Ismail Onur Filiz, and Sergey Edunov. Three and a half degrees of separation. In *https://research.fb.com/three-and-a-half-degrees-of-separation/*.

Vita

Douglas Steiert was born in St. Charles, Missouri. His first choice for college was the Missouri University of Science & Technology, where he earned his bachelors degree in Computer Science in 2015. After that, Douglas was granted the Scholarship for Service (SFS) that allowed him to pursue his doctoral degree in Computer Science. He continued pursuing this degree for approximately three years before transferring to the University of Missouri - Columbia to follow his advisor.

With the assistance from SFS, Douglas has been able to attend conferences at venues in California, Texas, and even in Germany, which have helped grow his knowledge in different research areas related to cybersecurity. He has also had the opportunity to work with various employers such as Ameren, Sandia National Laboratories, and the Department of Homeland Security.

Currently, Douglas is a Research Assistant for Dr. Dan Lin at University of Missouri - Columbia, while living in Foristell, Missouri. He hopes after achieving his final degree he can go back to work with his former employer, Sandia National Labs.