Spring 2019

Privacy preservation in social media environments using big data

Katrina Ward

Department: Computer Science

Recommended Citation

Ward, Katrina, "Privacy preservation in social media environments using big data" (2019). *Doctoral Dissertations*. 2797.
https://scholarsmine.mst.edu/doctoral_dissertations/2797

PRIVACY PRESERVATION IN SOCIAL MEDIA ENVIRONMENTS USING BIG DATA

by

KATRINA JOHANNA WARD

A DISSERTATION

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

2019

Approved by

Dan Lin, Advisor
Daniel Tauritz
Jennifer Leopold
Yanjie Fu
Donald Wunsch

PUBLICATION DISSERTATION OPTION

This dissertation consists of the following two articles which have been submitted for publication, or will be submitted for publication as follows, with the first publication being a journal extension of the author's previous work:

PAPER I: A Parallel Algorithm for Anonymizing Large-Scale Trajectory Data: Pages 14-62 have been accepted to ACM Transactions on Data Science 2019.

PAPER II: Risk Estimation Mechanism for Images in Network Distribution: Pages 63-94 have been submitted to ACM Conference on Computer and Communications Security.

ABSTRACT

With the pervasive use of mobile devices, social media, home assistants, and smart devices, the idea of individual privacy is fading. More than ever, the public is giving up personal information in order to take advantage of what is now considered every day conveniences and ignoring the consequences. Even seemingly harmless information is making headlines for its unauthorized use (18). Among this data is user trajectory data which can be described as a user's location information over a time period (6). This data is generated whenever users access their devices to record their location, query the location of a point of interest, query directions to get to a location, request services to come to their location, and many other applications. This data could be used by a malicious adversary to track a user's movements, location, daily patterns, and learn details personal to the user. While the best course of action would be to hide this information entirely, this data can be used for many beneficial purposes as well. Emergency vehicles could be more efficiently routed based on trajectory patterns, businesses could make intelligent marketing or building decisions, and users themselves could benefit by taking advantage of more conveniences. There are several challenges to publishing this data while also preserving user privacy. For example, while location data has good utility, users expect their data to be private. For real world applications, users generate many terabytes of data every day. To process this volume of data for later use and anonymize it in order to hide individual user identities, this thesis presents an efficient algorithm to change the processing time for anonymization from days, as seen in (20), to a matter of minutes or hours. We cannot focus just on location data, however. Social media has a great many uses, one of which being the sharing of images. Privacy cannot stop with location, but must reach to other data as well. This thesis addresses the issue of image privacy in this work, as often images can be even more sensitive than location.

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere appreciation and thanks to my advisor, Dr. Dan Lin, for her support, knowledge, and encouragement during my Ph.D and related research. I am grateful for her patience which has guided me throughout my research.

Aside from my advisor, I want to express my sincere gratitude to the rest of my committee: Dr. Daniel Tauritz, Dr. Jennifer Leopold, Dr.Yanjie Fu, and Dr. Donald Wunsch. Their comments, suggestions, recommendations, and encouragement have contributed greatly to my success and I could not imagine having made it without them. My research went in directions I didn't imagine because of their questions and guidance and for this I am grateful.

Finally, I would like to thank the Computer Science Department, including faculty and staff. They have provided me with many opportunities to grow as a person and a student. They have worked tirelessly to make sure I had everything I needed to succeed and I could not have done it without them.

TABLE OF CONTENTS

SECTION

APPENDICES

LIST OF ILLUSTRATIONS

Figure Page

PAPER I

PAPER II

LIST OF TABLES

Table Page

PAPER I

PAPER II

SECTION

1. INTRODUCTION

Since the early 1990's, social media has grown in popularity and has now become a significant part of everyday users' lives. Users are now able to share pictures, their locations, information about their lives, and other data with hundreds of other users without leaving their own home. For some, this has enriched lives and connections with people so much that it is deemed a necessity. This sharing of data has led to new challenges in terms of privacy, however. Before, when someone shared information, it spread to local communities and was unlikely to do much harm to that person. Now, however, information shared reaches hundreds of people in seconds and can spread to millions in a matter of hours. Information that once was fairly harmless can now destroy lives (7). In addition to social media, smart phones have risen in popularity and use since their debut in the early 2000's. Over half a billion mobile devices were added to networks in 2013 (51). With these phones, users no longer need to stop at a computer to share with their friends. Instead, they carry their computer with them, snap pictures and send messages instantly, play social games with countless people anywhere in the world from any location, and so much more. We are more connected than ever.

74% of adults use their smart phones to get directions and other information based on their current location (51). In a survey completed in 2018 (31), nearly all users have at least one social media account, with many having multiple. 30% of adults with an account on social media sites say they have at least one of those accounts include their current location in their posts (61). Even if we ignore hand held devices, as many as 96% of cars produced in 2013 are built with event recorders that include GPS.

Location information can be used by an adversary to track a user's movements, infer work locations and other habits, and even locate sensitive targets such as children, elderly, or government officials (20). As of now, this information is accessed and sold at alarming rates (6; 18). The immediate solution is to not make location data available at all. This poses two issues however. First, in order for some convenient applications to work, locations need to be sent to a server to be part of some sort of analysis. It is difficult to think of a GPS app functioning without location information. This information produces income revenue to telecommunication companies. Even though the data can be limited by not being available to the public, data breaches happen with alarming frequency (10). The frequency in which locations are collected is also alarming. A politician in Germany went to court to find out how often and what kind of data was collected by his cell phone company. The results were staggering. In just a six month period, his coordinates were recorded and stored over 35,000 times (4). Users are becoming increasingly cautious to trust that their data is being kept safe.

The second reason to keep location data available is that the data is very useful for analysis and predictions. With a large amount of user trajectory data, we could determine the best route for emergency vehicles during a specific time period, assist with traffic congestion prevention, infrastructure and evacuation planning, analysis of social behavior to provide more conveniences, advertising campaigns, and control of spread of diseases. If the data could safely be used, it has great potential for both private and public projects.

Solving the challenges of using location information is not a trivial task. The simple solution to protect users and still use the data is to hide their user id's and other identifiable information. The issue, however, is that even data related to the user can be used to identify them if used together. For example, if we were to hide a user's name and address from a dataset and use zip code, birth date, and gender, we would not be able to identify the user from any of these data categories individually. Many people in an area will share any single pieces of this information. However, used together, we could infer who the data is taken from. These are called $quasi - identifiers$ and are not sufficient to protect user privacy entirely (56). Instead, we would need to completely anonymize all of the data. This introduces another challenge, the size of the data. Mobile device users generate 7.2 EB of data every month, and is projected to grow to 49 EB per month by 2021 (51), and most of the data contains location information. The amount of information is exceeding the data management and analysis ability of existing traditional database type storage systems (15). While there has been research conducted to anonymize trajectories to address the privacy concern considering small data sets, there has been limited work on anonymizing large-scale datasets. Some of these methods proposed are *k-anonymization*, spatial-cloaking and data transformation (1; 4; 8; 16; 17; 19; 20; 22; 35; 39; 40; 41).

Unfortunately, few prior works addresses the scalability issue while keeping the privacy and high utility of the trajectory data, which is a newly occurring problem brought by the information explosion, i.e., the current need of dealing with exabytes of trajectory data. A naive approach to this problem would be to take a centralized approach and simply write it into a distributed framework such as Hadoop or Spark. However, there are additional challenges related to the distributed approach that create bottlenecks and significantly slow down processing. In addition, centralized approaches ported into a distributed environment often leads to significant errors and data loss, which undermines the need for accurate data utility (54).

In this thesis, the author presents a novel solution to this problem called MELT (53; 54). In this work, we present a three round map-reduce algorithm that guarantees strict k-anonymity on a dataset of user trajectory data in a fraction of the time it takes centralized approaches. This means that no user will have their data published for use anywhere unless there are at least $k - 1$ other users with identical trajectories. The intuition behind this approach is that if there are 100K users sharing the same trajectory, it will be impossible to pick a particular person out among them. In order to maintain the utility of the data, however, we need to retain as many trajectories as possible by performing transformations on the trajectories while also keeping those transformations minimal in order to keep the data as close to the original as possible. We also need to prevent inference of user identities through low frequency traveled routes. We present a way to achieve complete anonymity and maintain high data utility.

As stated earlier users are cautious when trusting their privacy to someone else. Rather than force every user to depend on a global k value to achieve k-anonymity, users would be more trusting if they could define their own level of privacy. The author presents an extension to MELT where we add the capability of users to define their own k value and base the final result off the more strict requirement (54). We show that this addition requires alteration of the original algorithm but adds little complexity change and utility loss.

In addition to adding the ability for users to define their own privacy requirement, we also address the idea of privacy vs utility. In most works, the two are considered inversely dependent on one another. That is, if we have high privacy, we have low utility and vice versa. For example, if we want 100% user privacy, we hide all of their data. With no data published, users are completely protected but there is no data utility as it cannot be used. However, if we publish all of the user's data we get 100% data utility, as it all can be used, but users have no privacy. In spite of this, for all cases in between, the two are not necessarily dependent. In a recent paper, (52), the authors claim that since privacy and utility are measured differently and compared to different datasets, the two cannot be referred to as dependent. Using observations from their work, we analyze our own solution and show that we maintain both high data utility and user privacy.

User privacy in social media is not limited to location data. Another source of data sensitivity is user images. Based on a user survey, almost every user using social media posts images of their personal life (31). With a single image, information about a user's home, children, work, or other private information can potentially be leaked. Still, many users prefer to share their images, though often with only certain people. This has been addressed in a limited capacity with many social media sites allowing the creation of groups for users to share certain information with, such as family, friends, work, etc. Maintaining groups can be tedious at best. Within family we can have close family and extended. A user may want to share with all family except one person or perhaps to only best friends. Creating groups to share to the right users can become inconvenient and cause users to not see the service as easy to use and therefore stop using it or ignore the the privacy implications. In addition, though images are shared to a specific person, it is often impossible to tell who that person will share to or who will see the image later. If a user shares a party picture with her sister, who is married to a person that has a friend who works with the original user, then that user's supervisors could see the picture. This may not be in the best interest of the user. Or perhaps the user shared a picture of her children to her sister who has a friend who is a friend of someone who may be an unknown predator.

In this thesis, the author presents REMIND, an algorithm designed to calculate the probability of an image reaching unintended audiences and make suggestions on who to share an image with to limit the risk that the image will reach those audiences (31). In addition, it prevents users not in the chosen group from viewing the image based on the preferences of everyone in the photo. Finally, based on user history and choices, it develops policies for new images being shared based on who is in the images so that share filtering becomes automatic and requires little to no user input. Not only does it address the current user, it addresses everyone in the image. If a user shares a picture of themselves and their best friend, it considers the policies and restrictions of the best friend as well. Not only do we protect the privacy of the user, but also other users in the same image and with little interference from the users. This makes sharing images convenient and maintains their privacy choices. When combined with social media, users will be able to share their images without worrying about tedious group sharing policies and their privacy will be protected in the background.

The current growth of social media, mobile devices, and connectivity between people means user privacy is a concern that will only continue to grow. The work presented in this thesis is a strong contribution to protecting users while they enjoy these conveniences while also making use of the data to improve physical safety and improve overall quality of life. The author demonstrates that there can be a balance between user privacy and data utility and that instead of fighting against user data being used, we can embrace using the data safely.

This thesis formally presents two major pieces of work: (i) A Parallel Algorithm for Anonymizing Large-Scale Trajectory data which includes earlier work presented at the Mobile Data Management conference as MELT: Mapreduce-based Efficient Large Scale Trajectory Anonymization, and (ii) REMIND: Risk Estimation Mechanism for Images in Network Distribution. In addition, as summarized in Appendix A, these works are supplemented by additional publications by the author.

2. LITERATURE REVIEW

2.1. PRIVACY PRESERVING TRAJECTORY PUBLISHING

There have been many centralized approaches for trajectory anonymization. Most of them (17; 19; 22; 35; 39; 40; 41) output anonymized trajectories in the form of cloaking regions or centers of clusters. For example, in (19), the spatial-temporal cloaking technique is applied to generate cloaking regions covering segments of trajectories. In (1), Abul et al. consider a trajectory as a cylindrical volume where the radius represents the location imprecision. They then perturb and cluster trajectories with overlapping volumes to ensure that each released trajectory volume encloses at least $k - 1$ other trajectories to achieve k-anonymity. Similarly, in (35), each trajectory is an ordered set of spatio-temporal 3D volumes. In (34), Monreale et al. cluster trajectories and transform them into a sequence of centroids of Voronoi cells. In these approaches, the anonymized trajectories do not follow the road network constraints. They can be located even in the middle of two parallel roads, and hence, are not beneficial for traffic analysis on individual roads, which our approach aims to achieve. Also, GPS systems are becoming more and more accurate. A cellphone can tell a person's location within a few feet, making methods such as these unusable in practice.

In (11), trajectories are clustered based on a distance function and then a location time triple in an anonymized trajectory is replaced by an existing triple with close proximity in the original trajectory to satisfy k-anonymity. However, two triples, though close in proximity, may belong to two different roads making it easy for the adversary to identify fake trajectories given that the road map is publicly known. In (1), Abul et al. used a coarsening strategy which removes one or more spatial points in a trajectory to achieve anonymization. An anonymized trajectory may contain disconnected paths. This is different from our approach which preserves continuous trajectories based on road-network information. Similarly, in (8), Chen et al. adopt a greedy algorithm to suppress locations in the trajectories to achieve anonymity. However, using suppression alone may decrease the utility of the anonymization results. They did not provide any experimental study to prove the effectiveness of the approach. Unlike the previous works which are based on the similarity of trajectories, Yarovoy et al. (56) group trajectories based on so-called quasi-identifiers, which are hard to be selected in practice. Anonymization using these quasi-identifiers is nearly impossible as knowledge of more than one of these can lead an adversary to infer information about a user.

Rather than representing trajectories as a set of coordinates, some works (34) represent trajectories as landmarks or locations of interests. However, these kind of trajectories provide mainly moving patterns rather than real trajectories as considered in our work. In addition, a few works (1; 8) make the assumption that attackers have certain prior knowledge and take such prior knowledge as input for anonymization, while our anonymization is more general and does not need such assumptions.

Finally, we would like to discuss the more closely related work (20; 38; 40) which generate anonymized trajectories following the road-networks. In (38), Pensa et al. proposed a prefix-tree based anonymization algorithm which guarantees k-anonymity of the published trajectories in a way that no trajectories with support less than k will be published. Later, Gurung et al. (20) identified a so-called inference-route problem in the anonymization results produced by (38), and proposed an improved anonymization algorithm that achieves stronger privacy guarantee. Our proposed parallel trajectory anonymization algorithm is developed based on this latest centralized trajectory anonymization strategy (20) while addressing new challenges raised by the scalability and the MapReduce technique. In (39; 40; 41), Poulis et al. have proposed a cluster based anonymization which is similar, but they do not address scalability and also suffer from the inference route problem as defined in (20) Most recently, He et al. (23) has proposed a method to protect user privacy by not publishing the real data generated by users. Instead, they take samples of real datasets and then generate a synthetic trajectory dataset based on information from the real data, probabilities, and noise added. While this does protect user privacy, it causes a loss of utility which the authors don't analyze. Additionally, there is still the inference route problem later discussed in Section 3. Our solution is proven to retain utility in large scale while avoiding the inference route problem. Additionally, MELT allows the user to select the level of privacy, k, and thus adjust the trade-off between privacy and utility. This is something not allowed in (23).

In addition, an overview of our work was introduced briefly in (53). In the journal version (54), we have made the following new contributions: our new algorithm allows each user to define his/her own privacy requirement; we conducted detailed analysis regarding the trade-off between privacy and utility as well as complexity; we added a detailed description and examples for the proposed algorithms; we implemented our work in both Hadoop and Spark.

2.2. PROCESSING LOCATION DATA IN PARALLEL

Many works (59) have been proposed to utilize map-reduce to perform data mining, pattern classifications, document retrieval, etc., in large-scale datasets. However, there are very few works on large-scale trajectory data processing. Yang et al. (21) proposed a method called TRUSTER for query processing over trajectory data using MapReduce. In (29; 43), MapReduce-based algorithms have been proposed to handle trajectory matching. Eldawy and Mokbel (12) proposed SpatialHadoop as an extension to MapReduce to provide support for spatial data management. However, to the best of our knowledge, while these works address indexing and querying trajectories very well, none of the existing works address the issue of anonymizing a large volume of trajectory data.

Aside from Hadoop, there are other parallel approaches. In the most recent work (44; 45), Shang et al. find similar trajectories and group them together in parallel. Compared to their work which requires multiple passes over the data, our approach is capable of scanning the dataset only once and therefore further reducing the computation time and increasing utility.

2.3. PRIVACY VS UTILITY

In a recent paper (52) on the discussion of privacy versus utility in regards to the anonymization of data, Li and Li discuss the trade-offs while measuring privacy versus utility. In many works, privacy and utility are viewed as dependent on one another. For example, if we took the extreme example and hide all user data, therefore achieving 100% user privacy, we also have zero utility as the data cannot be used for anything. However, if we publish 100% of the user data, then we can say we have 100% data utility but no user privacy. The authors of (52) claim that for all other circumstances however, privacy and utility are measured differently and against different datasets, and therefore should not be compared to one another as though dependent. They also discuss the way privacy and utility should be measured to have an accurate analysis. Using their observations and calculations, we analyze our own work and give detailed examples.

2.4. PRIVACY POLICY RECOMMENDATION SYSTEMS

Our work 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 that has not been explored in the past. More details are elaborated in the following.

There have been many privacy policy recommendation systems (2; 5; 13; 33; 37; 47). They typically utilize certain types of machine-learning algorithms to analyze users' profiles, historical privacy preferences, image content and meta data, 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 (26; 46; 49; 58). 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 (48; 57). For example, Squicciarini et al.(48) 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.(57) 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.

2.5. PRIVACY BREACH FROM FRIEND-TO-FRIEND SHARING

Since our work considers the privacy breach caused by friend-to-friend sharing, we review works that also examine this aspect. Akcor et al.(3) 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 of 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.(27) 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.(25) 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 preassumes 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, Kokciyan et al. (36) 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 prevents the potential privacy breach at the beginning of the sharing.

2.6. IMAGE POLICY CONFLICTS

Lastly, there have been works on resolving image policy conflicts among multiple users. Such and Criado (50) look at conflicting policies between users in images for when an image is to be shared. They propose a set of concession rules that model how users would actually negotiate to reach the common ground. In addition, there have also been general approaches for integrating access control policies of collaborating parties (42) which however requires the users to clearly specify how these policies should be combined.

Compared to all the existing works on image privacy preservation, our work distinguishes itself in two main aspects. First, it is probably the first time that the large volume of image sharing statistic data 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, our system 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.

PAPER

I. A PARALLEL ALGORITHM FOR ANONYMIZING LARGE-SCALE TRAJECTORY DATA

Katrina Ward

Department of Computer Science

Missouri University of Science and Technology

Rolla, Missouri 65409

Tel: 417-217-4273

Email: kjw26b@mst.edu

Dan Lin

Department of Computer Science

University of Missouri

Columbia, Missouri 65211

Tel: 573-884-6628

Email: lindan@missouri.edu

Sanjay Madria

Department of Computer Science

Missouri University of Science and Technology

Rolla, Missouri 65409

Tel: 573-341-4856

Email: madrias@mst.edu

ABSTRACT

With the proliferation of location-based services enabled by a large number of mobile devices and applications, the quantity of location data, such as trajectories collected by service providers, is gigantic. If these datasets could be published, they will be valuable assets to various service providers to explore business opportunities, to study commuter behavior for better transport management, which in turn benefits the general public for day to day commute. However, there are two major concerns that considerably limit the availability and the usage of these trajectory datasets. The first is the threat to individual privacy as users' trajectories may be misused to discover sensitive information, such as home locations, their children's school locations, or social information like habits or relationships. The other concern is the ability to analyze the exabytes of location data in a timely manner. Although there have been trajectory anonymization approaches proposed in the past to mitigate privacy concerns. None of these prior works address the scalability issue since it is a newly occurring problem brought by the significantly increasing adoption of location-based services. In this paper, we conquer these two challenges by designing a novel parallel trajectory anonymization algorithm that achieves scalability, strong privacy protection and high utility rate of the anonymized trajectory datasets. We have conducted extensive experiments using MapReduce and Spark on real maps with different topologies, and our results prove both effectiveness and efficiency when compared with the centralized approaches.

Keywords: Trajectory anonymization, Large-scale, Divide and conquer, MapReduce

1. INTRODUCTION

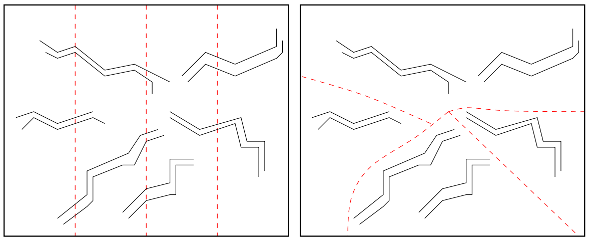
One of the fastest growing trends in mobile technology is the use of location-based services for applications such as social networking, vehicle tracking, and targeted advertising. As reported by Census, more than 471,000 people commute into Los Angeles county everyday which results in more than 10 million trajectories each month (9). According to Cisco, global mobile data traffic has reached 7.2 exabytes a month and is increasing rapidly (51). In 2013, 526 million mobile devices were added to cellular and WiFi networks, including a large increase in data usage for location-based mobile applications. Currently, 74% of adults use their smart phones to get directions and other information based on their current location. 30% of adults with an account on social media sites say they have at least one of those accounts include their current location in their posts (61). Even if we ignore hand held devices, as many as 96% of cars produced in 2013 are built with event recorders that include GPS. As a result, a huge amount of location information has been collected and stored for analytics.

These collected location data have great potential for statistical usage in various applications such as traffic congestion prevention, infrastructure and evacuation planning, analysis of social behavior, advertising campaign, and control of spread of diseases. While the benefits provided by location datasets are indisputable, many challenges remain to be addressed to actually realize the benefits. One critical challenge is how to ease customers' worries on location privacy (30) when they are told their location data have been collected and may be used for analysis. A politician in Germany went to court to find out how much and what kind of data was collected by his cell phone company. The results were staggering. In just a six month period, his coordinates were recorded and stored 35,000 times (4). Without a strong privacy guarantee, cautious customers will soon be reluctant to subscribe to location-based services. To address the privacy concern, several methods have been proposed such as *k-anonymization*, spatial-cloaking and data transformation (1; 4; 8; 16; 17; 19; 20; 22; 35; 39; 40; 41). Unfortunately, none of the prior works addresses the scalability issue which is a newly occurring problem brought by the information explosion, i.e., the current need of dealing with exabytes of location data compared to only millions of data in the past.

In this paper, we propose a novel parallel trajectory anonymization algorithm under the map-reduce (14) programming paradigm, called MELT, which encloses the following four features: **M**apreduce-based, **E**fficient, **L**arge-scale **T**rajectories, and “melting" the trajectories for anonymization to preserve their privacy. We present our algorithms by following the MapReduce programming paradigm since it clearly exemplifies the divide and conquer spirit which is heavily used in our proposed parallel algorithm.

It is not a trivial task to convert existing centralized anonymization approaches to a parallel version. To better understand the challenge, let us look at the following naive solutions. A common approach for handling a large-scale dataset is to partition the big trajectory dataset into many small datasets and then anonymize individual small datasets in parallel using the existing centralized approach. For dataset partitioning, there are two straightforward methods. One is to partition the road map vertically (or horizontally) into equal-width stripes, and the other is to partition the map into grids of equal-size cells. Then, anonymize trajectories falling in each partition separately. However, such partitioning of trajectory datasets can easily split a single trajectory into multiple segments as shown in Figure 1(a) where solid lines indicate trajectories and dashed lines indicate partitioning. As a result, groups of popular trajectories that are across multiple partitions will be treated separately, which could severely lower the utility of the anonymized data. This is because some trajectories will be anonymized segment by segment rather than being considered as a whole. The anonymization results may contain disconnected or shorter trajectories which would have lower data utility rate compared to that produced by the original centralized approach that directly works on the entire dataset. Moreover, anonymizing partial trajectories can introduce additional privacy breach, called inference-route problems as elaborated later.

An ideal case is to have the map partitions match the natural clusters of trajectories as shown in Figure 1(b) so that the quality of the anonymization results will be as close as that obtained from the centralized approach and original data. To achieve this, one may think of



(a) Vertical Partitioning (b) Ideal Partitioning

Figure 1. (a)Naive map partitioning approaches break up trajectories into small segments which will lower data utility.(b) Ideal partitioning to keep trajectories whole.

using existing trajectory clustering algorithms (24) to get the map partitions. However, this would not work either because clustering algorithms typically requires scanning the dataset multiple times. Multiple scanning of exabytes of trajectory data is very time and resource consuming, and thus it may not be practical. Instead, we expect to scan the dataset at most once.

To overcome the aforementioned challenges, in this paper, our proposed MELT algorithm makes the following new contributions:

- We propose a dynamic data partitioning algorithm that automatically adapts to the distribution of trajectories and partitions the map to capture the natural clusters of trajectories.

- We propose three rounds of map-reduce processes that fully parallelize the conventional trajectory anonymization process within a single scan of the dataset.

- We ensure that the anonymization results achieve *k-anonymity*, *inference free*, and high data utility rate. That is, any trajectory in the published anonymization results will have at least $k - 1$ other identical trajectories, and no further information can be inferred from analyzing and comparing trajectories in the anonymization results.

- We analyze the trade-off between privacy and data utility and also conduct extensive experiments on real road maps using Hadoop and Spark. The experimental results demonstrate that our approach is capable of efficiently handling large-scale datasets that cannot be processed by the centralized approach.

The rest of the paper is organized as follows. Section 2 formally presents our problem statement. Section 3 elaborates the detailed algorithms of our proposed MELT. Section 4 analyzes the trade-off between privacy and data utility. Section 5 reports the experimental study.

2. PROBLEM STATEMENT

In this work, we aim to anonymize large-scale trajectory datasets in parallel while maintaining high data utility.

2.1. DATA REPRESENTATION

Following the same setting as that in (20), the road network is modeled as a directed graph, where each edge corresponds to a road with objects moving in one direction, and each node represents an intersection. Raw data collected by location-based applications contains their user information as a three-tuple $\langle ID, loc, t \rangle$, where ID is the user ID the user's location, loc which corresponds to a GPS coordinate at an intersection, at timestamp t, respectively. Each road segment between intersections is assigned a road id r_i. The anonymization is carried out on the set of trajectories (Definition 1) within the same time interval t_{int} to preserve the time relationship among trajectories.

Definition 1 *(Trajectory): User u's trajectory is represented as* Trj_u=$\{r_1, r_2, ..., r_n\}$*, where* r_1*, ...,* r_n *are IDs of roads visited by u in sequential time order.*

After anonymization, the output dataset contains information in the form of:

$$\langle r_1 dir_1, r_2 dir_2 ..., r_n dir_n, support\rangle$$

where r_i is a road id in the representative trajectory of nodes visited by users, dir_i is the direction the user traveled on r_i, and support is the number of users who have traveled the entire representative trajectory. Such representation is sufficient to derive trajectories or traffic flow information.

2.2. PRIVACY DEFINITION

We aim to achieve *strict k-anonymity*. That is, no trajectory will be published unless at least $k - 1$ other trajectories are identical. The intuition is that if k is sufficiently high, it is impossible to identify an individual user among k identical users.

Definition 2 *(Strict k-anonymity over trajectories): Let Trj be a trajectory. We say Trj satisfies strict k-anonymity if Support(Trj) is no less than k, where Support(Trj) is the number of identical trajectories matching Trj.*

Trajectories that satisfy strict k-anonymity do not contain any inference-route problem (20). The notions regarding the inference-route problem are given below.

Definition 3 *(Road frequency): Let W be a time interval, and let k be a threshold. We say a road is a frequent road if the number of objects moving along one direction on this road is no less than k within time interval W. We call the number of moving objects as the frequency of the road.*

Definition 4 *(Inference route): Let Υ be an intersection of roads r_1, ..., r_m, and let U_i^+, U_i^- be the sets of objects moving toward and outward Υ on road r_i $(1 \leq i \leq m)$ during W, respectively. If $\exists\, U_i^+, U_j^-, |U_i^+| \geq k, |U_j^-| \geq k$, and $(0 < |U_i^+ - U_j^-| < k$ or $0 < |U_j^- - U_i^+| < k)$, then we say Υ has an inference-route problem.*

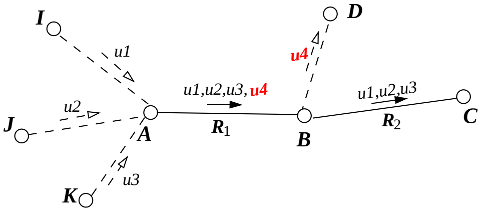


Figure 2. An Example of the Inference-Route Problem with u_4's location inferred.

In the above definition, the constraints $|U_i^+| \geq k$, $|U_j^-| \geq k$ ensure that only frequent road segments are considered, and (0< $|U_i^+ - U_j^-| < k$ or 0< $|U_j^- - U_i^+| < k$) check if there is a chance for a road segment to be inferred. To have a better understanding, let us consider a toy example in Figure 2 which shows four users leaving their homes (I, J, K, A) for work. Let k be 3, which means a trajectory can be published if at least three users have this trajectory. If trajectories are anonymized segment by segment due to map partitioning (illustrated by dashed dotted lines), the anonymization output will contain two roads r_1 and r_2 with supports 4 and 3 respectively, and has the following the inference-route problem. Specifically, if an adversary observes that Alice passes by road r_1 and r_3 every weekday, then Bob can infer that u_4 is Alice's ID from the published anonymized trajectory dataset because Alice is the only one who may enter road r_3 by comparing the set of anonymous IDs in the published road segment. Using this ID, the adversary would be able to track Alice's other movement. According to Definition 4, node B is an intersection of three roads. On road r_1, $U_{r_1}^+ = \{u_1,u_2,u_3, u_4\}$; on road r_2, $U_{r_2}^-=\{u_1,u_2,u_3\}$. Since $U_{r_1}^+ - U_{r_2}^- = \{u_4\}$, $|U_{r_1}^+ - U_{r_2}^-|$ = 1< k, node B has an inference-route problem. In general, given a threshold k, if an adversary can link any anonymous ID to a particular user with probability greater than $\frac{1}{k}$ by using the above method, then we say there is an inference-route problem.

2.3. PERFORMANCE METRICS

Our proposed trajectory anonymization approach will be evaluated in terms of three factors: (i) privacy guarantee (discussed in the previous section), (ii) efficiency (CPU time), and (iii) data utility rate.

The data utility of the anonymized trajectory dataset will be evaluated using the following two commonly accepted metrics: Precision and Recall. Intuitively, the less difference between the anonymized dataset and the original dataset, the better quality the anonymized dataset is. It is clear that some error will be introduced due to the divide and conquer technique inherent to a distributed approach. To gauge the utility of the final dataset, we compare the results from our approach to the original dataset. Recall will allow us to see how much data is lost by splitting the data across multiple nodes that are unable to share information with one another. Precision will tell us how similar our results match those of the original data, thereby showing how much error is introduced. The calculations for these are as follows:

$$Precision = \frac{Pairs\_Matching\_Traj}{Traj\_Output\_By\_MapReduce}$$

$$Recall = \frac{Pairs\_Matching\_Traj}{Pairs\_Matching\_Traj+Missing\_Traj}$$

In the above equations, $precision$ is calculated by comparing the anonymized trajectories obtained from our distributed approach using MELT to the original dataset. Specifically, for each trajectory obtained by our algorithm, we look for the most similar trajectory from the original data set, i.e., the trajectory with the largest number of common nodes and compare the support from each approach. If the identified pair of trajectories are identical, it means we were able to find the same cluster and that trajectories were grouped together correctly and error was not introduced. Then, the pair are removed from the data sets when searching for the next pair of identical trajectories. In the equation for $Recall$, $Missing_Traj$

refers to the number of trajectories that the centralized approach (20) found that our algorithm did not. This is due to similar trajectories not being partitioned to the same region of the map or similar trajectories becoming dissimilar if broken up across multiple regions, and so they were removed for not meeting our *k* threshold. The higher the precision and recall, the better the accuracy of our approach and the less error has been introduced.

3. THE PROPOSED MELT

In this section, we first present an overview of our proposed MELT algorithm, and then elaborate the detailed steps.

3.1. AN OVERVIEW OF MELT

Figure 3 illustrates the overall data flow in the MELT system. The proposed MELT system consists of two major phases: (i) map partitioning; and (ii) divide-and-conquer-based trajectory anonymization. The second phase is further divided into three rounds of processes to ensure the maximum parallelism of the anonymization steps, which are *Trajectory Assignment*, *Pre-Classification*, and *Melt*.

- *Map partitioning*: At the beginning, we divide the road map into multiple regions with the goal of keeping as many similar trajectories as possible in the same region. A desirable partitioning (as shown in Figure 3) will prevent long frequent traffic flows from being separated or wrongly pruned, and retain as much of the original data as possible to maximize the data utility after the anonymization. However, the map partitioning is an extremely challenging task because it is done before seeing all the actual data.

- *map-reduce-based trajectory anonymization*: We design three rounds of map-reduce processes to fully parallelize the trajectory anonymization process. The *Trajectory Assignment* round takes the map partitioning results as input, assigns original trajecto-

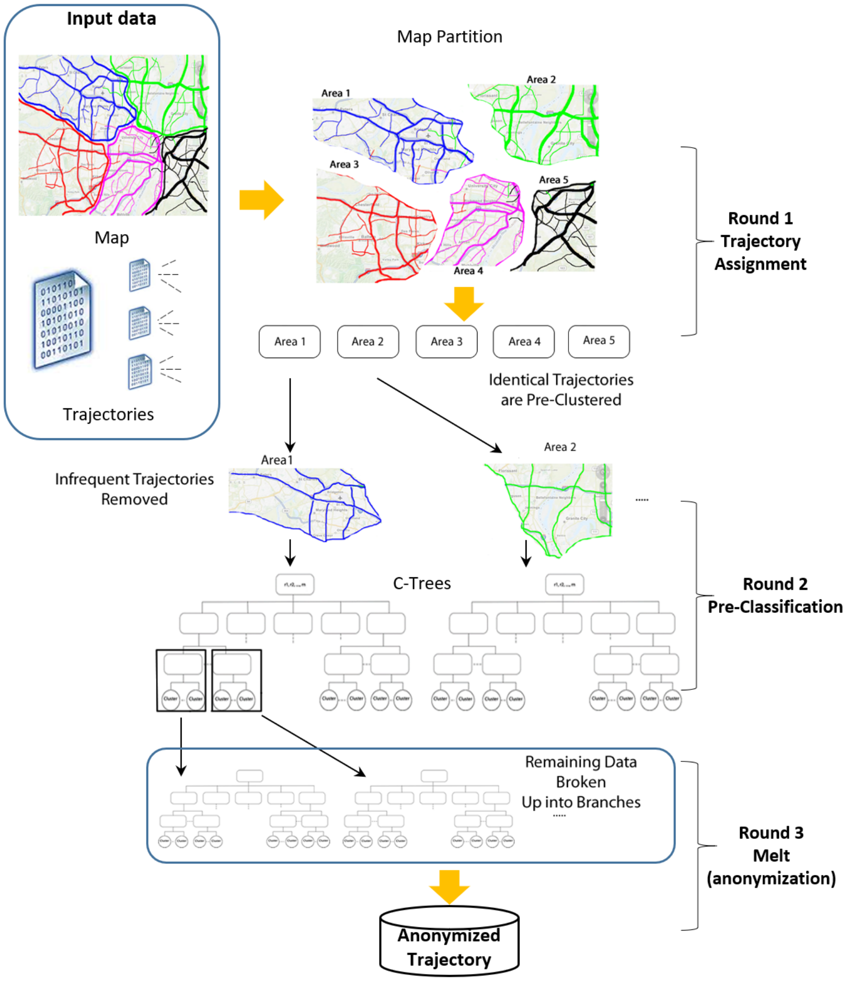


Figure 3. An overall representation of the architecture of our approach.

ries to the corresponding regions, and calculates the supports of the same trajectories in each region. The *Pre-Classification* round removes the roads that could cause the inference route problem, and then organizes the remaining trajectories into a tree structure based on their similarity. Finally, the *Melt* round conducts the fine-grained anonymization on the sub-trees obtained from the previous round.

3.2. ROAD MAP PARTITIONING

The first step in our approach is to partition the map into regions that meet the following conditions: (i) each region contains an approximately equal number of trajectories, so that the workload can be balanced for the worker nodes during the parallel anonymization phase; (ii) each region contains trajectories that are as similar as possible to one another so that the anonymization results will preserve high data utility rate as previously discussed. However, achieving these two goals is very challenging since we do not want to scan the entire location dataset to determine the regions, which would otherwise be too time consuming. Therefore, we propose an adaptive map partitioning algorithm (shown in Algorithm 1) that partitions the map based on the concept of the traffic flow. Though we could also parallel this step in our approach, we note that even for the largest city map, we are able to partition it in an order of milliseconds. Thus, we chose not to complicate this step by making it parallel.

In order to obtain map partitions adapted to different traffic flow datasets, the key idea of our approach is to identify hot spots on the road-networks using a small sample dataset, and then expand them to reconstruct possible traffic flow that forms the sub-regions. To determine a good sampling rate, we conducted extensive tests on various datasets and found that any sample above 30% preserves nearly the same accuracy compared to using the entire dataset. From the sample dataset, we extract hot spots as follows. Hot spots are popular road intersections. Observe that trajectories tend to frequently pass by some hot spots such as commercial centers, major road-highway intersections, and intersections in between residential areas. Based on this observation we aim to identify hot spots as hubs in each map partition. Specifically, we compute node frequency (i.e., the sum of trajectories from the sample data set passing by the node/intersection) and road frequency (Definition 3) from the sample trajectories. We sort all the nodes in a descending order of its frequency.

Then, we start a depth-first road expansion from the most frequent nodes. It is worth noting that we can parallel this road expansion process by starting the expansion simultaneously from the top m frequent nodes which are at certain distance from each other. Given a starting node (n_0), the road segments connected to n_0 will be considered in a descending order of their road frequency. At the first expansion, the road (r_0) with the highest road frequency will be selected. After that, the road r_1 with the most similar road frequency to r_0 will be selected, and then r_2 with the most similar road frequency to r_1. In general, each time we select the subsequent road segment that minimizes $\Delta = f(r_i) - f(r_{i-1})$, where $f(r_i)$ and $f(r_{i-1})$ denote the frequency of the road r_i and its previous road r_{i-1}. In this way, we project possible traffic flow. There are three stopping criteria for the expansion, of which only one needs to be met: (i) the length between the hot spot and the current node is equal to or greater than the average length of the trajectories in the data set; (ii) the road frequency difference (Δ) exceeds the threshold $\rho = \frac{max f(r_i)}{d_{r_i} - 1}$, where d_{r_i} is the degree of the node (the number of the roads connected to this intersection). The threshold computes the case when an incoming traffic flow is equally distributed among outgoing roads; (iii) the current area being calculated exceeds $\frac{total\_map\_area}{num\_regions}$. Then, the next round of expansion will start again from n_0 to see if more traffic flow can be identified. If not, we move to the node with the next highest frequency that is not already included in the previous road expansion, and continue the same road expansion as we did for n_0. The roads associated with the same hot spot form a region.

After we visit all the hot spots, there may still be some roads which are not yet included in any region. These roads are usually sparsely distributed and mixed in the roads that have already been classified. In this case, we conduct a quick breath-first search starting from unvisited nodes found in the road map and find the closest region for each unvisited node. If two regions are of the same distance to the unvisited node, the region with the less difference in road frequency will be selected. At the end of our map partitioning, every road will be matched to a region.

ALGORITHM 1: Road Map Division

```
Data: List of Hot Spots(sampled), HS; road-network (V, E); number of
      reducers, RNO
Result: List of road map divisions, Regions
$index_{HS} \leftarrow 0$
totalArea ← RNO
startFlag ← true
while startFlag = true do
    node ← $HS[index_{HS}]$
    if node.areaData is not empty then
        define list of nodes NList
        define region r
        add node to region r
        add node to NList at index 0
        node.depth ← 1
        add r, node.depth to node.areaData with key r.id
        while NList is not empty do
            firstnode ← NList[0]
            remove element of NList at index 0
            for each $node_{neigh}$ in firstnode.Nbrs do
                if $node_{neigh}$.areaData does not have key r.id then
                    if $node_{neigh}$ does not violate the stopping criteria then
                        $node_{neigh} \leftarrow firstnode.depth+1$
                        add $node_{neigh}$ to region r
                        add r, $node_{neigh}.depth$ to $node_{neigh}.areaData$ with key
                          r.id
                        add $node_{neigh}$ to NList at index 0
                    end
                end
            end
        end
        add r to Regions
    end
    else
        totalArea ← totalArea+1
    end
    if $index_{HS} \geq$ HS.size() || $index_{HS} \geq$ totalArea then
        startFlag ← false
    end
end
FindAreaForUnvisitedNodes(V, Regions)
return Regions
```

3.3. PARALLEL TRAJECTORY ANONYMIZATION

After the map partitioning, we now proceed to discuss how to conduct trajectory anonymization in parallel. A straightforward way to convert the centralized anonymization algorithm to a parallel version is to directly execute the centralized algorithm for the sub-dataset in each region obtained from the map partitioning. However, this straightforward approach does not fully parallel many functions in the anonymization process, is still unable to process the volume of data, and it is much less efficient than our proposed approach as we will soon observe in the experiments.

In order to parallelize the entire anonymization process as much as possible, we study the centralized algorithm (20) in details. The original centralized algorithm consists of the following main phases: (i) compute the road frequency; (ii) remove infrequent roads from trajectories; (iii) cluster similar trajectories and compute the representative trajectory for each cluster. Moreover, the centralized algorithm needs to scan the trajectory dataset twice, one for road frequency computation and the other for trajectory clustering.

After examining the detailed anonymization steps, we propose three-rounds of map-reduce processes, which ensures that the whole dataset is scanned only once. Specifically, we parallelize the road frequency calculation in the first round of process. Then, we examine the road frequency in parallel and remove infrequent roads. While examining the road frequency, we also build an index that group same trajectories together to prepare for the clustering. Finally, we anonymize the different sub-trees of the trajectory index in parallel based on the global and individual k requirements.

In what follows, we elaborate the detailed algorithms in the map-reduce programming style which consists of two major functions "map" and "reduce". To put it simple, the map function of each round is in charge of "divide" while the reduce function is charge of "conquer". Additionally, we leverage the built in nature of the shuffle-sort phase between map and reduce to further optimize clustering, shuffling, and sorting to the reducers for the best time optimization and load balancing.

3.3.1. Round 1: Trajectory Assignment. In this first round, the map function takes raw trajectory data as input, whereby trajectories are still represented as coordinates. The number of reducers corresponds to the number of regions we obtained during the map partition phase. The map function will first map the trajectories to a region in the road network map. Specifically, the map function assigns an initial score of 0 to each region on the map for the current trajectory, and increases the score of the region by 1 when a road segment in the trajectory is found belonging to that region as seen in Algorithm 2 on in the beginning. We compare each node in the trajectory to a partitions list and update the score in the locations list. Recall that each road has already been marked with the region ID during the map partitioning.

When the entire trajectory has been analyzed, the mapper finds the region with the highest score for the current trajectory. If the region contains more than $\alpha\%$ of the trajectory, the trajectory is considered to belong to that region. Otherwise, we may split the trajectory across two regions that contain the majority of the trajectory. While the value of $\alpha\%$ has no affect on run times in our experiments, it does affect the data utility rate. The greater the value of α the more likely we are to place the trajectory in the correct area with the most similar trajectories. A small α could lead to the same problems we have when naively partitioning the map. In the case where one area does not contain more than $\alpha\%$ of the trajectory, we propose two methods: (i) splitting the trajectory as mentioned above; (ii) consider the entire trajectory as part of the region that contained the greater portion of the trajectory.

When the correct region has been identified, the mapper then converts the coordinates in a trajectory to a list of road IDs. The conversion to the road IDs makes it easier for trajectory comparison and frequency computation and reduces the intermediate data size. Finally, the map function outputs two types of key-value pairs. One is $\langle (A_i, Trj(R_1, ...R_k)), 1\rangle$, where A_i is the region of the trajectory Trj represented by road IDs $(R_1, ...R_k)$. Also, for each road examined, the map function outputs a key value pair

$\langle (A_i, R_j), 1\rangle$, where A_i is the region ID of the road R_j. In each mapper, we further utilize a combiner to sum up the frequency of each trajectory and each road so as to reduce the amount of intermediate results to be sorted and shuffled among the follow-up reducers. This intermediate sum is referred to as *support* in Algorithm 2.

The reduce function then computes the sum (or frequency) of each trajectory and each road. Note that each reducer handles one region of the map, thereby keeping similar trajectories together. The road information is saved into a hash table while the trajectories are output to an intermediate file. The final output of this round is the result of a key-value conversion, where trajectory information has been moved from the key section to the value section so that we obtain trajectories for each map partition: *<Ai, (Trj$_i$, count)>*.

To better understand our approach, we use the following running example throughout the paper. Suppose that we have the map and trajectory data for a city. For easy illustration, we partition the map into four regions denoted as *A1, A2, A3,* and *A4* respectively. Let the anonymization parameter k=3 and the trajectories in our data set be *Trj$_1$, Trj$_2$,...,Trj$_n$*, whereby x_{ij} and y_{ij} denote the x and y coordinates for the j$^{\text{th}}$ node in the i^{th} trajectory:

Trj$_1$ = $\{(x_{11},y_{11}), (x_{12},y_{12}), ..., (x_{1i}, y_{1i})\}$

...

Trj$_\text{n}$ = $\{(x_{n1},y_{n1}), (x_{n2},y_{n2}), ..., (x_{nj}, y_{nj})\}$

These trajectories are sent to mappers, and the mappers would map each trajectory to a region and convert the coordinates to road ID's. Assuming that these trajectories are mapped to region $A1$, the output from this first round of map-reduce contains two types of key-value pairs, the trajectories and their assigned areas as well as each road segment and their assigned area, as follows:

<(A1, $(r_1, r_2, r_3, ...)$), 1>,

<(A1, $(r_1, r_2, r_3, ...)$), 2>,

<(A1, $(r_3, r_5, r_2, ...)$), 4>,

...

ALGORITHM 2: First Round of Map Reduce

```
Data: $TrajFile$ = trajectory file, $PartList$ = Map Partitions List, $RoadMap$ =
      map of coordinates to road ids
Result: List of trajectories[Key = partition, trajectory: Value = support]
List of Roads[Key = partition, road: Value = support]
for $each\ traj\ in\ TrajFile$ do
    $locationsList$ for $each\ node\ in\ traj$ do
        $PartList$.find($node$) $locationsList$ ← node location
    end
    $largestLocation$ ← NULL
    for $each\ location\ in\ locationList$ do
        if $node\ count\ in\ location > node\ count\ in\ largestLocation$ then
            $largestLocation = location$
        end
    end
    $partition$ ← $largestLocation$ MapperOut ← [($partition$, $traj$), 1] for
      $each\ Road\ in\ newTraj$ do
        MapperOut ← [($partition$, $Road$), 1]
    end
    Reducer Input: MapperOut([$partition$, $TrajOrRoad$), $cntList$] while
      $MapperOut\ not\ empty$ do
        $count$ ← 0
        for $each\ cnt\ in\ cntList$ do
            $count$ ← $count + cnt$
        end
        ReducerOut ← [$partition$, ($TrajOrRoad$, $count$)]
    end
end
```

<(A1, r_1), 1> ((A1, r_1), 2>...

<(A1, r_2), 1> ((A1, r_2), 2> ((A1, r_2), 4>...

The reducer computes the overall frequency of trajectories and roads and output the following:

<(A1, r_1), 4> <(A1, r_2), 4> <(A1, r_3), 4> <(A1, r_4), 3>...

<A1, (($r_1, r_2, r_3, ...$)), 3) >,

<A1, (($r_3, r_5, r_2, ...$)), 4) >,

...

3.3.2. Round 2: Pre-Classification. In this round, the mappers will first process all the key-value pairs containing road frequency. We leverage the sorting in the shuffle-sort phase to ensure road frequency information arrives before trajectory frequency. For each road, the mapper checks its frequency. If the frequency is lower than the anonymization threshold k, the mapper records this infrequent road ID in a hash table. In algorithm 3, this is shown as a $RoadList$. As trajectories arrive at the reducers, the infrequent roads are removed from the trajectories, thus removing a source of the inference route problem. If the infrequent road is at one of the ends of the trajectory, it is simply removed. However, if it is in the middle, the trajectory is split into two if both segments contain at least two road segments. This helps us maintain the road network constraints and maintain natural road trajectories.

Although it is now possible to execute the centralized anonymization algorithm directly on the above dataset in the reducers, we propose another round of map-reduce to further parallelize the anonymization process. Specifically, the centralized anonymization algorithm requires to construct a C-Tree (20) to group trajectories based on similarity before final anonymization. The original C-Tree construction algorithm considers all the trajectories in a sequential manner within that reducer. However, with all the reducers operating in parallel, this means all the data would be held in memory at once with several C-Trees being constructed. Instead, we propose a parallel algorithm to build the C-Tree in a distributed fashion which also enables the subsequent anonymization to be executed in sub-trees in parallel. Put simply, we take advantage of map reduce's batch processing by processing branches of the C-Tree in parallel, rather than the entire dataset at once.

To construct the C-Tree in parallel, the reducers create a list of slots whereby each slot corresponds to a branch of the original large C-Tree. In Algorithm 3, these slots are called $Branches$. The number of branches chosen depends on the number of available reducers and the size of the data, meaning that in the final round we will utilize more reducers and leverage map-reduce's batch processing at the reduce phase to process all

ALGORITHM 3: Second Round of Map Reduce

```
Data: $RoadList$ = Key, Value Pairs[Partition, (Road, Support)
$TrajList$ = Key, Value Pairs[Partition, (Traj, Support)]
Result: [part.id, (newTraj, Support)]
$infrequentRoads \leftarrow$ NULL
for each Road in RoadList do
    if $support < k$ then
        $infrequentRoads$.add($Road$)
    end
end
$Branches \leftarrow$ NULL
for each $Traj$ in $TrajList$ do
    for each $roadid$ in $Traj$ do
        if $roadid$ in $infrequentRoads$ then
            $newTrajs \leftarrow Traj$ - $road$
        end
    end
    for each $branch$ in $Branches$ do
        if $(newTrajs, branch).error < \gamma$ then
            $branch$.add($newTrajs$)
        end
        else
            $closestBranch$.add($branch$)
        end
    end
    if $branch \leftarrow unassigned$ then
        if $Branches$ not full then
            $Branches$.add($newBranch$($newTrajs$))
        end
        else
            $closestBranch$.add($newTrajs$)
        end
    end
end
```

branches of each C-Tree faster. Similar to the map partitioning in the beginning, we want to make sure trajectories most similar to one another are placed into the same branch just as they would in the original C-Tree. We also observe that the density of the data plays a role in the partitioning as well. Dense data, that is trajectories all close together, can be broken into more branches than sparse data. An example of branching is shown in Figure 4.

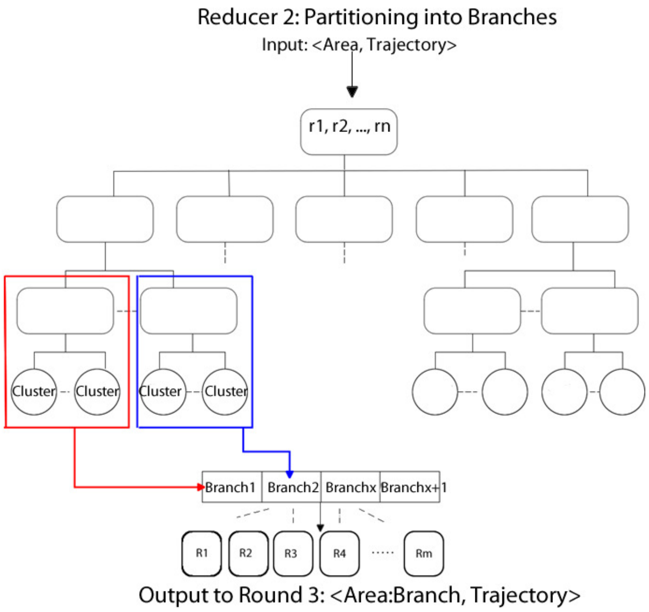

Reducer 2: Partitioning into Branches
Input: <Area, Trajectory>
r1, r2, ..., rn
Cluster
Cluster
Cluster
Cluster
Branch1
Branch2
Branchx
Branchx+1
R1
R2
R3
R4
.....
Rm
Output to Round 3: <Area:Branch, Trajectory>

Figure 4. In the second reducer phase, instead of using one large c-Tree, we break off similar branches into smaller chunks to benefit from map-reduce's batch processing in the reducers.

For each trajectory, if a branch exists such that the trajectory is at least γ similar to that branch, the trajectory is assigned to the same branch. γ is a system parameter that controls how similar the trajectories in each branch are to one another. A γ value of 100% means the trajectories are identical. Otherwise, if there are unassigned branches, the trajectory is assigned to a new branch. Finally, if there is no more room for more branches and the trajectory is not very similar to the given branches, it is assigned to the most similar branch. This means that the most similar trajectories will be assigned to the same branch, minimizing errors as defined in Section 2.3. The output is represented as $\langle part.id, (Traj_i, support)\rangle$ where $part.id$ is the partitioning id consisting of the original region of the map concatenated with the branch index the trajectory was assigned to and $support$ containing the number of identical trajectories currently sharing the same roads. The design of the key ensures that each area is broken up into multiple branches and processed in chunks in the final stage, rather than all at the same time, while still making sure to keep the most similar trajectories in the same chunk.

Referring to our running example, using the road frequencies in the hash table for the correct area, we remove all infrequent roads from the trajectories and output the new trajectories with only the frequent roads. As a result, our trajectories have been converted to the following:

$$Trj_1 = r_1, r_2, r_3$$

$$Trj_2 = r_1, r_2, r_3$$

$$Trj_3 = r_1, r_2, r_3$$

$$Trj_4 = r_1, r_2, r_3, r_4$$

$$Trj_5 = r_6, r_7$$

$$Trj_6 = r_6, r_7$$

$$Trj_7 = r_2, r_3, r_6, r_7$$

If we remember that we set $k = 3$, then in Trj_4, r_4 would be removed for being infrequent. The trajectories are sent to the second reducers where trajectories 1, 2, 3, and 4 are partitioned to a branch, and the remaining would be grouped in another branch based on their similarity.

3.3.3. Round 3: Melt. This final round of our algorithm conducts trajectory anonymization in parallel. The algorithm is outlined in Algorithm 4. The mappers here are just identity mappers which directly pass the datasets obtained from the previous step to the reducers without additional process. Each reducer will receive the dataset that corresponds to a single branch of the C-Tree, and construct the C-Tree as defined in (20). nlike a two round algorithm where an entire area would be anonymized at once in each reducer, processing branches instead of entire areas significantly increases efficiency by taking advantage of map-reduce's's batch processing and load balancing. More specifically, for each trajectory, if the total support is greater than k, it forms a cluster and is added to the tree. If the trajectory has a support lower than k, it will traverse the tree and find the most similar cluster and attempt to merge with that cluster. Note that this is significantly more efficient than the centralized approaches which need to compare trajectories in the entire dataset, not a small subset such as in our approach. When merging, a representative trajectory is calculated for the cluster and error is calculated.

When we attempt to merge trajectories with low frequency into a cluster, we calculate the representative trajectory of the cluster as though the trajectories were added and if the error is increased by less than β%, the trajectories are added. The error threshold is an adjustable system parameter which can also be set by the user depending on the amount of error that is acceptable to them. If no clusters are suitable for merging, a new cluster is created for the trajectory and added to the tree. By doing this, we are able to possibly merge multiple trajectories with low support together in order to meet the k threshold, should they be suitably similar. When all possible merges are done, representative trajectories of any clusters with at least k support will be output. Any trajectories remaining with less than $\frac{k}{2}$ support will be removed from the data set. Any remaining trajectories with more than $\frac{k}{2}$ support, but less than k will have false trajectories added, increasing the support to k. This is done to increase data utility while keeping error low and preserving privacy. The final anonymization output contains road IDs and road frequency in the representative trajectories of all the clusters. Our anonymization result guarantees strict k-anonymity.

Finishing up our running example, in the last round of reducers, trajectories 1, 2, and 3 would be put into a cluster since their frequency is greater than k and the representative trajectory would be calculated. Trj_4 would also be merged into this cluster since after the infrequent road was removed and no additional error is added, it now matches this cluster. In another reducer, we cluster trajectories 5 and 6; however, if we try to merge Trj_7 with trajectories 5 and 6, the error is too great and is placed in its own cluster. Since the frequency for Trj_7 is less than $\frac{k}{2}$, it is removed entirely. The frequency for the cluster containing trajectories 5 and 6 however, is greater than $\frac{k}{2}$. So we simply add a dummy trajectory to it to bring the frequency up to k. The final output of the anonymized set would be:

ALGORITHM 4: Third Round of Map-Reduce

```
Data: $Key$ = areaID, Branches index
$Value$ = RID$_{\text{traj}}$, sum of traj
Result: Anonymized Trajectory List
ClusList ← list of clusters
for each traj, sum pair do
    new CLUSTER ← traj, sum pair
    $CLUSTER_i$.repTraj ← traj
    $CLUSTER_i$.error ← 0
end
for each CLUSTER$_j$ with fewer than k trajectories do
    Calculate distance between $CLUSTER_j$ and each other cluster
    Calculate representative trajectory, REP of adding CLUSTER to closest
      cluster, CLUSTER’
    Calculate error of adding $CLUSTER_j$ to CLUSTER’, NEWERROR
    if ERROR change $\leq$ 5% then
        add $CLUSTER_j$ to CLUSTER’
        CLUSTER’.repTraj ← REP
        CLUSTER’ ← NEWERROR
    end
end
for each remaining CLUSTER with fewer than k trajectories do
    if number of trajectories in CLUSTER $\geq \frac{k}{2}$ then
        while number of trajectories in cluster $\leq k$ do
            add CLUSTER.repTraj to CLUSTER
        end
    end
end
for each CLUSTER do
    if number of trajectories in CLUSTER $\geq k$ then
        for each traj in CLUSTER do
            return (" ", CLUSTER.repTraj)
        end
    end
end
```

r_1, r_2, r_3	r_1, r_2, r_3	r_6, r_7	r_6, r_7
r_1, r_2, r_3	r_1, r_2, r_3	r_6, r_7	

The trajectories are now k-anonymized. The data still maintains utility and individual users cannot be identified.

3.4. PRIVACY PRESERVATION WITH WITH VARYING k

So far in this work, the privacy of users depends on a globally fixed k value chosen before trajectories are anonymized. However, for some users, their privacy requirements may vary compared to other users. This can be observed in many social media environments. One user may share everything to the public, while others choose not to use social media at all due to privacy concerns. While these two examples represent both extremes, there are users all across the spectrum between them. Therefore, we propose to extend our approach to accommodate users with different privacy requirements and allow each individual user to set his/her own value of k. As of now, this is the first work to allow user control for their privacy while maintaining the high efficiency and accuracy of the final results.

ALGORITHM 5: Clusters with varying k

```
Data: $TrjList$ = List of trajectories in cluster with individual $k$
$RepTrj$ = Representative Trajectory of Cluster
$Support$ = Overall Support of Cluster
$k$ = Global Minimum Support Threshold
Result: Anonymized Trajectory List
for each $Trj_i$ in $TrjList$ do
    $k_i \leftarrow$ User Required Support for $Trj_i$
    if $k_i > Support$ then
        if $(k_i - Support) < 0.05Support$ then
            Increase Support to $k_i$
        end
        else
            Remove trajectory from cluster
            $Support \leftarrow Support - \beta$
            Recalculate representative trajectory
        end
    end
    if $Support < \frac{k}{2}$ then
        break
    end
end
```

In order to achieve variable k during the anonymization, we propose the following modifications to our parallel algorithm which are also represented in Algorithm 5. The first change is to maintain the personal k value for each trajectory throughout all the three rounds of parallel processes. Note that although it seems that keeping the personal k value attached to the user may be more space efficient, it is impossible to distinguish users since no identifiers or even quasi-identifiers are stored in the system. Therefore this information is attached to the trajectory itself as a local constraint. Specifically, it is carried throughout the map reduce process as part of the value attached to the support for the trajectory.

The second change occurs in the third round of anonymization, particularly during the output of representative trajectories. When deciding whether or not to output a cluster, we will check to make sure each individual k requirement is met. The naive solution is to compare each personal k with its cluster support. If the cluster support is lower than one of the users' k, meaning that the user's privacy is not met, the cluster will be removed. However, this naive solution can result in removing a large number of clusters simply due to some users who have a higher k value than their other cluster members. To reduce loss on the data utility rate, we propose Algorithm 5 that attempts two options. Instead of directly removing the whole cluster when its support is lower than some users' k values, our first option is to try to increase the cluster support and recalculate the error. If the introduced error is within the desired threshold, the cluster will be preserved. If the first option does not work, that is, it produces too much error, we try the second option which is to remove the trajectory with particularly high k from the cluster and recalculate the representative trajectory for that cluster. This additional step provides users with more controls on their privacy while introducing very little time overhead to the performance as later shown in Section 5.

4. SYSTEM ANALYSIS

4.1. PRIVACY AND UTILITY ANALYSIS

Regardless of how private we can make information, we recognize that it makes little difference if there isn't any utility in the final result. All related work on trajectory anonymization as well as other works on privacy subscribe to the common belief that privacy and utility are inversely related and impossible to reconcile together. In a recent paper (52), Li and Li discuss the trade-offs and measurements of both privacy and utility and show that this is not necessarily the case. In our work, we look at the three main observations from their work and compare our approach to show that we maintain both high privacy and high utility.

The first observation is that knowledge gained on a small population or individual has the largest impact on privacy, while knowledge gained overall on a population increases utility. This makes sense since utility is the information we gain from the data. While individual data helps in some specific cases, information on an entire population is veritably useful. However, general knowledge tells us little about a single individual. This means that privacy must be considered on an individual basis rather than on an entire population as a whole, while information gained on an individual has little utility compared to an entire population. If any individual can be identified, then we must assume any individual can be identified. In our work, we consider each individual's privacy to be unique and important, therefore, our work has a large impact on user privacy. In Figure 5 we see that at a k value of 0.25% of the data, we can find the original trajectory of 52% of the trajectories with at least 65% confidence. However, that number drops significantly and at 0.5% of the data as k, we can no longer accurately identify any individual trajectory from the anonymized dataset. For this reason, we have chosen 0.5% as our k threshold for our experiments. Similarly, we are looking for large groups of trajectories in order to study and gain knowledge on patterns while removing information on individuals and small groups; therefore we also have a large utility gain. In Figure 9, we see that a *k* threshold up to 1% of the data keeps a high utility, however, after this, the utility drops. This shows we are able to maintain the privacy of every single individual while also looking at the optimal utility.

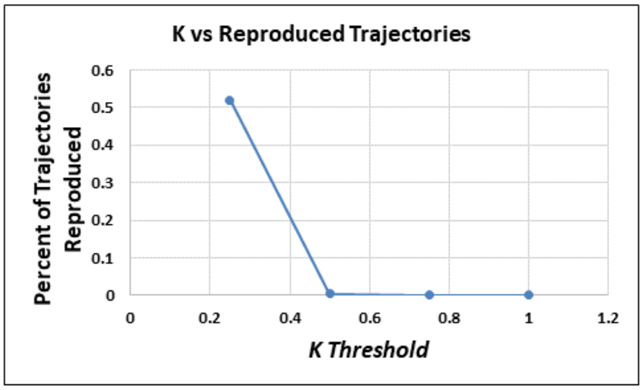


Figure 5. At a k value of 0.5% we are no longer able to identify the original trajectories with at least 65% confidence.

The second observation states that privacy should be measured at its worst case scenario for reasons stated above, while utility should be measured at its best case. This means that the unacceptable privacy loss of even a single individual should be considered the privacy loss for the entire population in the data while utility should be looked at as how much information is ultimately gained. In our research, we observe that no individual's data is published if it is below our *k* threshold, therefore it is appropriate to measure privacy loss in terms of *k* since we do not have any quasi-identifiers in our data. Additionally, since we allow users to choose their own level of privacy and we consider it a strict constraint, we can say that we focus strongly on individual privacy. Similarly, we always look at the highest and average precision and recall in order to determine data utility. This means we look at the best and average utility of the final data sets and therefore measure at its best case. As mentioned above, we are able to maintain the privacy at it's worst case while also maintaining optimal utility gain.

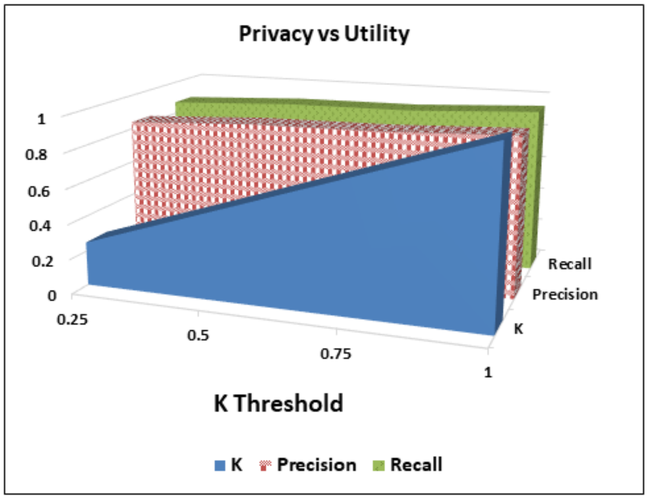


Figure 6. Comparison of K vs Utility showing they are measured differently, and therefore cannot be immediately assumed to be inversely proportional.

Most often, privacy and utility are compared to one another and are considered to be inversely dependent. However, the last observation from (52) is that privacy is to be measured against another anonymized dataset to determine if privacy is maintained, improved, or lost while utility should be measured against the original dataset to determine how much of the original data is maintained. More specifically, privacy and utility are measured differently and against different datasets and should not be compared to one another as though they are dependent. In Figure 6, we see that as k increases, the utility remains the same for reasonable k. Paired with Figure 5, we see we are able to maintain strict privacy requirements while maintaining high utility. The two are clearly not inversely related as previous assumptions would suggest. We can understand this by remembering that we do not use any quasi-identifiers in our data that would create a link between the two.

In our tests, we compare privacy to the anonymized data produced by the centralized approach in (20). This is to make sure that through our divide and conquer technique, we are maintaining the same privacy standards and not unintentionally relaxing our privacy requirement. We measure our utility by calculating the number of trajectories retained from the original data set and how similar those trajectories are to determine how much utility the anonymized dataset has.

In our work, we calculate utility loss by calculating the precision and recall of the results from our approach compared to the centralized approach and the original data set to determine: (i) How much of the original dataset we retained and (ii) How similar are our results compared to the centralized and original data. This allows us to see what percentage of the data we kept after anonymization and how much error we introduced in our approach.

Privacy loss, however, is more challenging to calculate. While we can give an arbitrary k value that sounds secure enough, we choose to use the measurement included in (52) $Ploss = \frac{max}{t} Ploss(t)$ This calculation matches with the second observation, that privacy should be measured at its worst case and considered for the entire population. We do not use any quasi-identifiers in our work, so we do not calculate privacy loss for each individual attribute. Instead, we look at privacy on each hop in a trajectory for a user and calculate it as:

$$Ploss = \sum_{i=1}^{h}(P(n_i) * 2P(\sum_{j=1}^{m}(t_i))$$

$P(n_i)$ is the probability that an adversary can identify an individual user from published users on a road segment. $2P(\sum_{j=1}^{m}(t_i)$ addresses the inference route problem in that it is the probability that both a user chose to take an infrequent road and that the adversary was able to correctly choose the same road in identifying the user. Since infrequent roads are not published, this is considered very random for both the user and the adversary. There are possibly several infrequent roads a user can take, therefore it is a summation of all those possible paths. We can make it a little more simple by remembering that the minimum number of people on a published road segment is k and the worst case being that no unknown, infrequent paths were taken. Therefore we can say that for i hops in a published trajectory, privacy loss can be calculated as:

$$Ploss = \sum_{i=1}^{h}(P(k))$$

Since we use no quasi-identifiers in our data and we remove infrequent roads to protect against the inference route problem, we can say that k is a sufficient measure of privacy loss since we maintain all of the observations listed above and in (52).

4.2. COMPLEXITY ANALYSIS

In this section, we will look at each round of our algorithm and determine the worst and average case scenario. Map partitioning is not analyzed, as performing this task requires only the list of road intersections and a small sample set of the data. The size of this data is trivial compared to the trajectory dataset. As mentioned earlier, we only scan the entire dataset once. Therefore the processing time is mainly dominated by the second round of map-reduce of our algorithm. A more detailed explanation of this analysis can be found in Appendix B.

4.2.1. Round 1 Analysis. In the first round, we examine the time taken to assign trajectories to map partitions and output each trajectory and the roads in those trajectories as well as their assigned partitions. Based on this, the time for round 1 is: $\frac{cn+n}{x}$ where c is a constant representing the average number of nodes in a trajectory, n is the number of trajectories and x is the number of map partitions.

4.2.2. Round 2 Analysis. The data to reach the second round will be the roads and their assigned map partitions. The first portion of round 2 is to identify infrequent roads. The processing time for this can be represented as $\frac{m}{i}$ where m is the number of roads and i is the number of mappers. According to the number of occurrences of roads, we remove the infrequent roads from the trajectories, which takes the time $\frac{cn}{i}$.

Once infrequent roads are removed, we perform the pre-clustering algorithm. We represent the number of chunks we are breaking the data into as j and the value we define for k-Anonymity as k. In the worst case scenario, there will be only one reducer and each kind of the trajectory has no more than k similar trajectories. This will result in the maximum number of chunks of data. The amount of calculation can be represented as $\frac{n*\frac{n}{k}}{j}$. j is an arbitrary value set by the user and is calculated based on data size, therefore we can say the worst case scenario is $\frac{n^2}{k}$. The chance that every trajectory will need to be compared to every other trajectory is unrealistically small. Such a data set would have no use, as that would mean no two trajectories are even similar to one another. This implies a very tiny data set, in which this algorithm would not be used anyway, or a very unrealistic dataset where no data will be published at all in the end and the rest of the algorithm would terminate, resulting in faster, but useless processing. Therefore, the realistic worst case scenario is when the data is clustered evenly into k clusters, which would make the run time $\frac{n}{k}$. Finally, we consider the I/O cost as $\frac{n}{k}$.

4.2.3. Round 3 Analysis. In round three, each of the data chunks is put into C-Trees, and the representative trajectory of clusters meeting our k requirement are the output. For each trajectory, the algorithm finds the correct cluster in the C-Tree. The time to find these clusters is $nlog(cn)$, which is split over x reducers. However, in the previous round, we split the data going to the reducers into j chunks, therefore the time is $\frac{nlog(cn)}{jx}$. For each of these clusters, we calculate a representative trajectory: $\frac{cn}{x}$, and finally output the representative trajectories of the clusters that meet the k requirement: $\frac{n}{k}$.

4.2.4. Overall Complexity. Combining the three rounds of map-reduce, removing constants, and considering the altered worst case in round two, we are left with a run time of $O(n + nlog(n))$, which is significantly faster than current published methods. This run time assumes the worst case of the data being split into as many clusters as possible, none of which will be removed and everything published in the end. This is unlikely considering many side roads that are infrequently traveled, local residential areas, and suburban country roads. We can assume then, that that the run time will be realistically faster in practice.

4.2.5. Comparison to Other Approaches. In the most recent related work (45), the authors are able to find the trajectories meeting k by finding the top most frequent trajectories. This means they need to scan through the dataset at least once to count the number of times each trajectory exists. They also perform joins to fins similar trajectories which, in parallel is fairly fast using their approach. However, while merging has a constant cost, per trajectory searches for similar trajectories is costly, even in parallel. For large datasets like those tested in this work, the time would increase dramatically. Assuming worst case, the cost would be $n * n/r$ where r is the number of instances running in parallel and n is the number of trajectories in the data. Partitioning is done uniformly and not based on the map and location, therefore we have to assume that similar trajectories are not necessarily kept together. This means a greater utility loss or longer processing. Since all trajectories are compared, we can assume the longer processing time. This leaves at least a processing time of $O(n^2)$. For their work, no anonymization is done, only clustering of similar trajectories. Therefore, the anonymization step must be added to the time.

In our experimental algorithms, we compare to the centralized approach which requires scanning the entire dataset at least twice, while MELT only scans once. In addition, there is no distribution of the data or tasks in the centralized approach, so each piece of data is handled one at a time. Though the number of reducers operating through most of MELT is dynamic and changes based on data size, we know that for the first round the number of reducers is based on the number of map partitions. For a real map and data size, we find that eight partitions is ideal. This means that the centralized approach is eight times slower than MELT during the process of assigning trajectories to map partitions, and even slower for later portions of the algorithm. We see this again in Table 4 where the centralized approach is exponentially slower and even fails.

5. PERFORMANCE STUDY

In this section, we first present the experimental settings and then report the experimental results.

5.1. EXPERIMENTAL SETTINGS

All the tests were conducted on a cluster of four physical machines, each being Dell PowerEdge R730, Intel Xeon CPU E5-2650 v3 processors at 2.30GHz, 2 socket, 14 cores per socket. The machines are managed using VMWare and divided up into eight virtual machines, each running CentOS 64bit and having 14 CPU's and 60GB RAM. We used Apache Hadoop 2.3.0 and Spark 2.3.0.

Our experiments were conducted on both real and synthetic datasets. The real datasets (60) contain taxi data from drivers in New York City over a period of three years, all taken from the same three hour time frame to consider the potential privacy issues stemming from temporal information. The real datasets are represented in Table 2 and contain the data file sizes as well as the number of trajectories. Note that one smaller dataset was taken in order for the centralized and single round approaches to be able to process it and compare accuracy. In addition to the real datasets, we also generate several large synthetic datasets over real road maps so that we can show our approach is consistent over several sets of data and locations. The synthetic datasets contain trajectories with similar average length in the real datasets, which is about 15 to 20 nodes. The trajectory node distribution is random to avoid any bias. The synthetic dataset size goes up to 100GB, which is sufficient to represent commuter traffic in a large city such as Los Angeles or New York City over a three month period (9). Throughout our experiments, we refer to our datasets by the number of trajectories in the input data file. Table 1 presents the statistic summary of the synthetic datasets while Table 2 presents the real datasets.

Table 1. The number of trajectories and the corresponding file size

Number of Trajectories	5K	50k	500k	2M	5M	50M	250M	500M
Dataset Size	3M	14M	141M	540M	1.4G	14G	55G	103G

In our experiments, we compare the performance of the following algorithms in terms of efficiency and effectiveness.

Table 2. Real trajectory datasets tested for efficiency

Dataset Name	RS	RS50	RS100	RS200
Number of Trajectories	17K	175M	420M	862M
Data Size	137K	50GB	100GB	200GB

- Centralized: The centralized anonymization approach proposed in (20).
- Single-round MapReduce: This approach executes the original centralized anonymization algorithm directly on each sub-dataset after the map partitioning.
- Stripe: The map is partitioned into even sized stripes and then trajectories are anonymized using MELT-Hadoop.
- Grid: The map is partitioned into even sized squares and then trajectories are anonymized using MELT-Hadoop.
- MBR: The roads in the map are partitioned using minimum bounding rectangles (MBR) such as those in the R-trees. Specifically, we divide the roads into equal-sized groups based on their distance and enclose them in minimum bounding rectangles to form partitions. Anonymization is then performed using MELT-Hadoop.
- MELT-Hadoop: The full version of our proposed MELT algorithm in Hadoop.
- MELT-Spark: The full version of our proposed MELT algorithm in Spark.

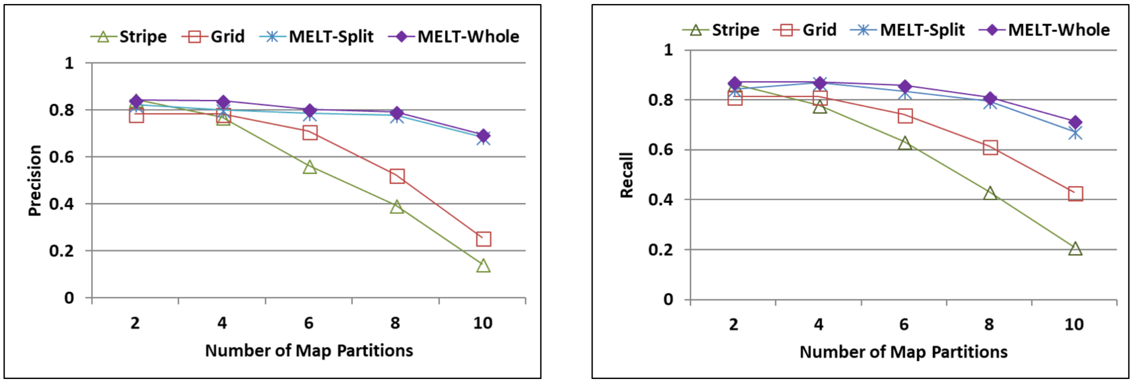
For the map partitioning, we compare our solution with the naive partitioning methods, as described in Section 2. To evaluate the anonymization accuracy, we compare our results with the centralized approach proposed by (20), as well as a single round map-reduce algorithm and naive partitioning methods. We test the effects of changing the k-threshold, the number of map partitions, dataset size, and multiple map topologies.

Finally, we test the effects of having user defined k on the data utility and processing time. Effectiveness is evaluated based on the precision and recall defined in Section 2. Efficiency is evaluated using CPU time.

5.2. EFFECT OF MAP PARTITIONING ON ANONYMIZATION ACCURACY

In the first round of experiments, we aim to study how map partitioning will affect the anonymization accuracy. We compare our dynamic partitioning approach to the naive partitioning methods introduced in Section 2 and the approach using the simple minimum bounding rectangles (MBRs). More specifically, our approach has two versions: MELT-whole, MELT-split. In the MELT-whole approach, we always keep the trajectories as a whole even if it crosses partitions, and assign the trajectory to the partition in which the majority of the trajectory lies. The MELT-split approach, instead, splits the trajectory if it crosses partitions, and assign each part of the trajectory to its respective partition. Figure 7 shows the anonymization accuracy of all the approaches in terms of precision and recall. Recall that both precision and recall need to be calculated against the original anonymization results produced by the centralized approach. Since the centralized approach cannot handle very large datasets, in this experiment, we adopt the largest dataset that the centralized approach can handle which contains 500K trajectories and set the anonymization parameter k to 0.5% of the number of trajectories.

As show in the figure, both versions of our proposed MELT yield significantly higher accuracy results compared to other approaches especially when the number of partitions is large. This conforms with our analysis that equally partitioned maps do not match the trajectory distributions as well as our proposed dynamic partitioning. Also, since our approach performs even better for the larger number of partitions, it indicates the advantages of our approach in large-scale datasets which typically needs more partitions to handle. We also observe that the MELT-whole achieves slightly better accuracy than MELT-split since MELT-whole would be able to keep more trajectories intact. In addition,



(a) Precision

(b) Recall

Figure 7. Anonymization accuracy when dividing the map into different number of partitions

it is expected that the more partitions, the lower the anonymization accuracy. This is because when the number of partitions is large, the area of each partition becomes small and hence more trajectories may cross partitions resulting in more difficulties in identifying similar trajectories. However, more partitions could help reduce the overall processing time for large-scale datasets since each partition handles a smaller amount of data. Therefore, we carefully choose the number of partitions based on the dataset size in the following experiments to balance the anonymization accuracy and the anonymization time.

5.3. EFFECT OF THE ANONYMIZATION PARAMETER *k* ON DATA UTILITY

As we already know that the proposed MELT has better anonymization accuracy than other approaches, we now take a further look only at the anonymization results of the MELT in terms of data utility. Note that both MELT-Hadoop and MELT-Spark versions have the same data utility rate since they implement the exactly same algorithms.

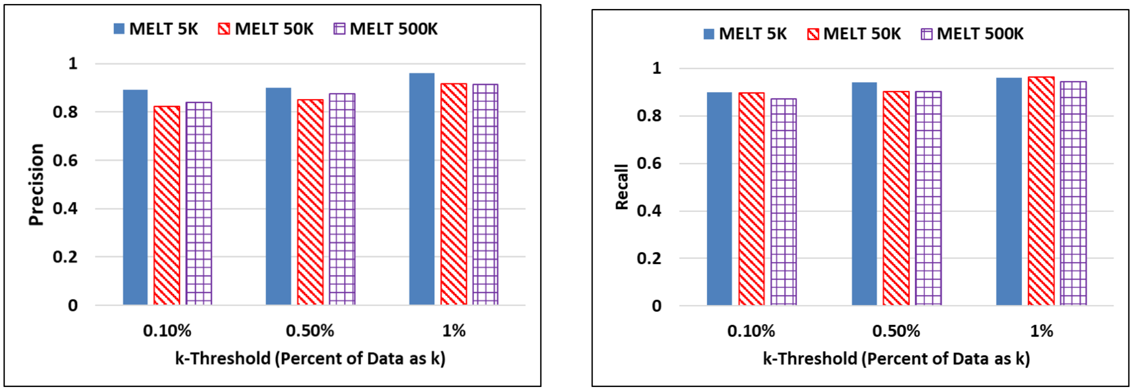
Specifically, we investigate the effect of the anonymization parameter k on data utility in terms of the number of trajectories retrained in the anonymization result. Since k may have more impact on smaller datasets, we tested three datasets ranging from as small as 5K to 500K trajectories. For each dataset, we vary k from 0.1% to 1% of the number of trajectories.

Table 2 shows the number of trajectories left after the anonymization. As expected, the smaller dataset with 5K trajectories ends up with just 20% of the original trajectories after the anonymization when $k = 1\%$ of the number of total trajectories (i.e., k=50). The larger the datasets, the more percentage of trajectories are kept after the anonymization. This is because in the larger dataset, the chances of finding similar trajectories is higher and hence more trajectories can be anonymized. In addition, we also observe that there is a significant reduce in the number of anonymized trajectories when k increases from 0.5% to 1%, which suggests that the anonymization parameter k should not be set to be too large. The larger the k value, the lower the probabilities will be to obtain similar trajectories in the group larger than k.

Table 3. Number of trajectories in the anonymization results

Number of trajectories in the dataset	5K	50K	500K
k=0.1% of data	4255	35809	285995
k=0.5% of data	3228	30910	279500
k=1% of data	878	18626	132602

Next, we are also interested in examining the effect of k on the precision and recall. As shown in Figure 8, we can see there is little difference in the accuracy across the different datasets, meaning using 1% of the data in the 5k dataset is about the same accuracy as taking 1% of the data in the larger datasets. This means that regardless of the size of data, keeping the same k percentage of the dataset will yield the same accuracy results. k does not need to change based on the data size.



(a) Precision

(b) Recall

Figure 8. Anonymization accuracy when varying *k*

It is true, however, that changing the percentage of the data that *k* represents has a significant impact on the accuracy. This is an expected outcome due to the fact that the larger the *k* value, the more trajectories need to be the same to be published. This means many more trajectories will be eliminated for not meeting our *k* threshold and therefore more data is lost. Based on our experiments, approximately 1% of the data is the optimal percentage of the data *k* should represent. As we see in Figure 9, the accuracy drops quickly after 1%. This same threshold occurred in all of our dataset tests.

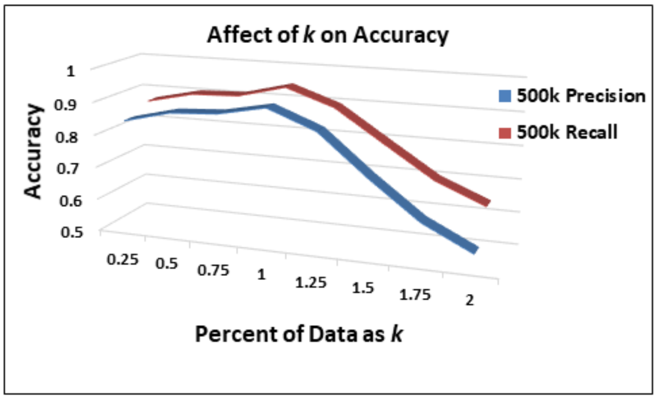


Figure 9. The accuracy of the data drops significantly when *k* represents more than 1% of the data.

5.4. SCALABILITY TEST

After obtaining the insights regarding the map partitions and the parameter k, we now proceed to evaluate the scalability of our proposed MELT which is indeed the major goal of our design. We first tested the centralized annonymization approach to see its scalability limit. As shown in Table 4, the centralized approach crashes when the dataset contains more than 2M trajectories.

Table 4. The anonymization time using the centralized approach

Number of trajectories in the dataset	5K	50K	500K	2M	RS(17K trajectories)
File size	3M	14M	141M	540M	137M
Processing time (minutes)	0.9752	22.6	1157.72	NA	1441.49

Since the centralized approach is impossible to complete the anonymization as shown in Table 4, we only plot the processing time of the parallel approaches in the following figures for clear comparison. In the experiments, the number of partitions is set to 8 and the value of k is set to 0.5% of the number of trajectories. As shown in Figure 10, the full version of the MELT-Hadoop is similarly fast as the MELT version with simple map partitioning methods, i.e., stripe and grid. This indicates that our dynamic map partitioning does not introduce much computational overhead. Moreover, we also observe that the single-round MapReduce approach does not scale well and crashes when the number of trajectories reach 5M. This proves the need of having multi-round of divide-and-conquer as in the full version of the MELT. Also, it is not surprising to see that the Spark version of MELT is about twice faster than the Hadoop version. The performance gain is attributed to the in-memory access by Spark.

In addition to the synthetic datasets, we also compare the same approaches using the real datasets in Table 2. Based on our results in Figure 11, our approach of MELT-Hadoop and MELT-Spark show much better scalability with the growing data size, with MELT-SPARK showing more than 120% improvement in CPU time compared to MELT-Hadoop.

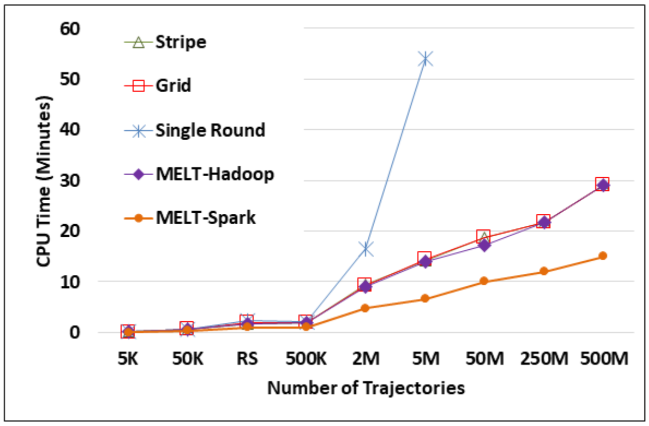


Figure 10. Effect of data size on anonymization time

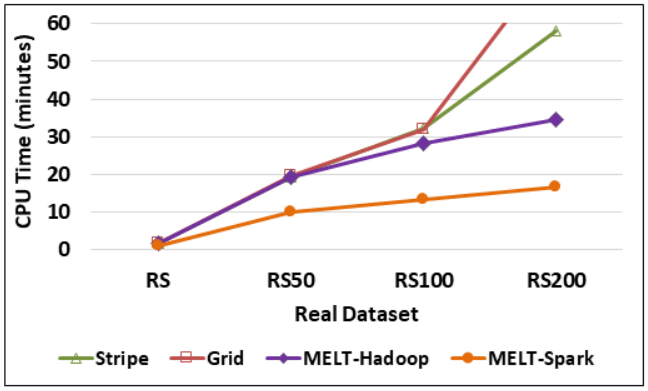
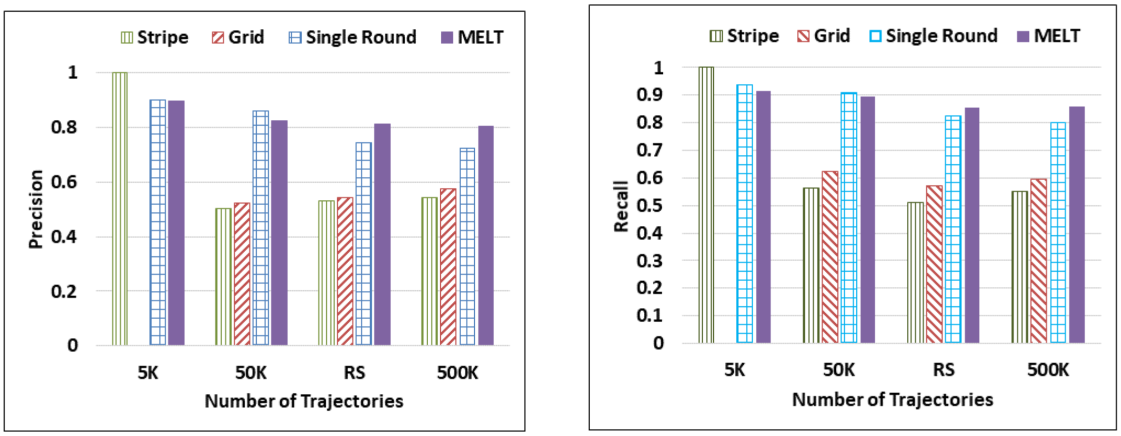


Figure 11. CPU time comparison on real datasets.

Further, we also compare the anonymization accuracy of all the parallel approaches. Since the anonymization accuracy is calculated by comparing the anonymization results with the centralized approach, Figure 12 shows the comparison results with the dataset up to 500K which is the maximum size that the centralized approach can process. Observe that the full version of the MELT achieves much better precision and recall than the naive partitioning approaches when the dataset grows larger. Note that the full version of MELT here refers to both MELT-Hadoop and MELT-Spark since they produce exactly same anonymization results. When the dataset is very small (say 5K), the anonymization accuracy is similar among all the approaches because the number of trajectories in the anonymization results is small too and hence less difference. Note that only two partitions are used for 5K dataset whereby the Grid partitioning approach is basically the Strip partitioning, and thus no result is reported for the Grid approach for the 5K dataset.



(a) Precision

(b) Recall

Figure 12. Effect of data size on anonymization accuracy

Additionally, for the real dataset tests, we calculated the recall as recall is compared to the original dataset and not the centralized anonymized dataset like precision is. We achieved an average of 0.81376 recall, meaning we maintained a high data utility rate in terms of number of trajectories retained from the original dataset. Precision could not be measured as the centralized anonymization approach could not process the datasets for us to compare.

5.5. EFFECT OF MAP TOPOLOGY ON EFFICIENCY AND ACCURACY

In this experiment, we evaluate the effect of the map topology on the anonymization efficiency as well as accuracy. In order to measure the accuracy, we generate 500K trajectories (the maximum size that the centralized approach can handle) on four different real maps representing big cities. We set the number of the partitions to 6 and k to 0.5% of the number of trajectories. As we can see in Figures 13, our proposed MELT-Spark is the fastest approach among all regardless of the map topology used. The single-round MapReduce approach is the slowest. This is because MELT, Strip, and Grid not only partition the original input dataset but also break the anonymization task into sub-tasks and hence achieve better parallelism than the single-round MapReduce.

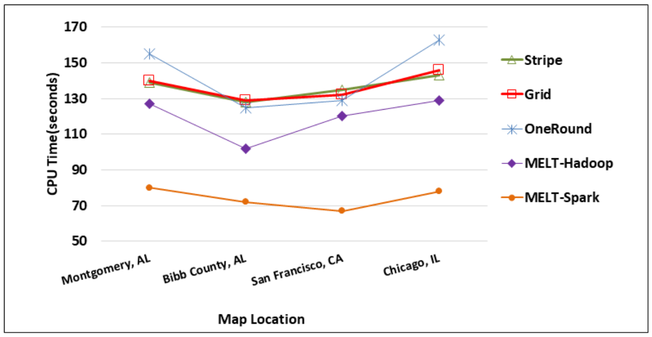
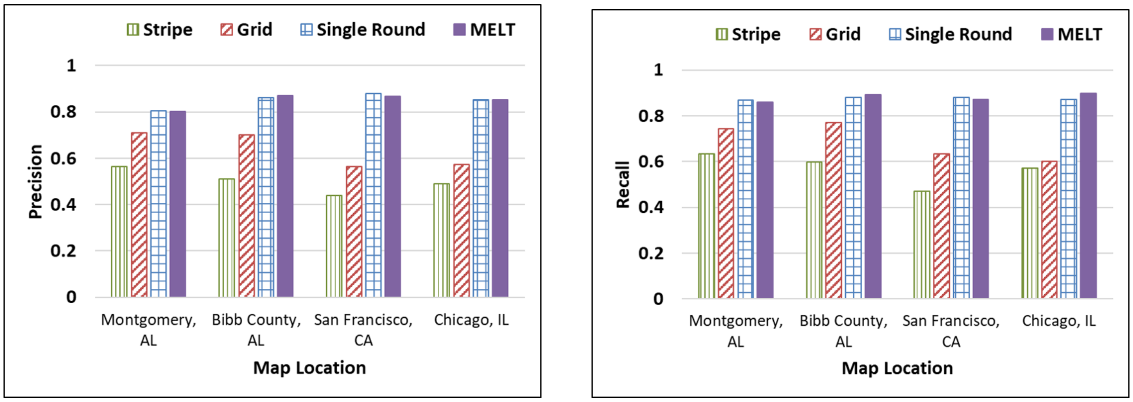


Figure 13. Effect of map topology on anaonymization time

When comparing the anonymization accuracy, MELT again achieves the highest accuracy among all. The reason is the same as previously discussed. It is because the equal partitioning cannot capture the real trajectory distribution. In addition, it is worth noting that the single-round MapReduce has the similar accuracy as the MELT since they adopt the same map partitioning approach.



(a) Precision

(b) Recall

Figure 14. Precision and recall for different map topologies

5.6. EFFECT OF USER DEFINED *k* ON ANONYMIZATION ACCURACY

Finally, we examine the effect of user defined k on the anonymization accuracy. In the dataset of 500K trajectories, for each user, we randomly generate its k value ranging from 0 (no privacy protection needed) to 0.5% of the number of trajectories. Then, we compare the precision and recall between the anonymization results using the same value of k (0.5% of the number of trajectories) and the anonymization result that used the variable k for different users. Table 5 shows that the changes in precision and recall are very little after introducing customized k. Such good performance is because our algorithm carefully adjusts the clusters of trajectories based on individual ks and maximizes the number of trajectories to be kept. The processing time after introducing the customized k is also almost the same as the version with the same global k, and hence we did not include the figure here. The processing time is similar because the algorithm for the customized k adds only small conditions to the original version which do not affect the overall processing time.

Table 5. Change of anonymization accuracy when using individual k parameters

Number of Trajectories	Precision Change	Recall Change
5k	-0.003	-0.002
50k	-0.003	0.00005
500k	-0.009	-0.004

We do note, however, that for other map partitioning methods, allowing user define k introduces more error and sets MELT apart from those approaches even further. This is because the size of the clusters are already smaller since similar trajectories are split across multiple partitions, and therefore removed from the final anonymized dataset. By introducing a larger k value in some clusters, causes more clusters to be lost entirely, where as if more similar trajectories were clustered together, the cluster could still be published. Therefore, MELT is best suited for user control over their privacy. Other recent approaches mentioned in the Literature Review do not have the ability to allow for user defined privacy requirements, and therefore cannot be compared.

6. CONCLUSION

In this paper, we propose a novel parallel algorithm called MELT to efficiently anonymize large data sets of trajectories. The proposed MELT employs three rounds of divide-and-conquer strategies that break the centralized anonymization task into sub-tasks that can be performed in parallel. The MELT also provides flexibility to individual users to define their own privacy levels. Our anonymization results on both real and synthetic datasets demonstrate that our approach successfully protects the personal data from direct knowledge or inference attacks while at the same time achieves high data utility rate. The extensive experimental study also shows that our method performs much better when compared with recent works in terms of computation speed, volume of data, and data utility. To further show the benefits of our work, we have also analyzed the relationship between the privacy and utility aspects of our approach.The outcome of this research can be used by cellular and other location service providers to publish anonymized trajectory data much faster so that it can be used in a wide variety of services in timely fashion without sacrificing user privacy.

Acknowledgements : This work was funded by the National Science Foundation (NSF-DGE-1433659, NSF-IIP-1332002), Department of Education (P200A120110).

REFERENCES

[1] Shang, S., Chen, L., Wei, Z., Jensen, C. S., Zheng, K., and Kalnis, P., *"Trajectory similarity join in spatial networks"* Proc. VLDB Endow., 2017,Vol 10, pp. 1178-1189, ISSN 2150-8097.

[2] Shang, S., Chen, L., Wei, Z., Jensen, C. S., Zheng, K., and Kalnis, P., *"Parallel trajectory similarity joins in spatial networks"* The VLDB Journal, 2018,Vol 27, pp.395-420

[3] The Apache Foundation. *"Apache Spark"*. https://spark.apache.org. [Online Access January 2019].

[4] Burke, M., *"Miami teen commits suicide in two-hour long facebook livevideo, the third in as many weeks"* http://www.nydailynews.com/news/national/miami-teen-commits-suicide-two-hour-long-facebook-live-video-article-1.2955175,2018, [Online Accessed 2019].

[5] Granville, K., *"Facebook-Cambridge Analytica Explained"* https://www.nytimes.com/2018/03/19/technology/facebook-cambridge-analytica-explained.html, 2018, [OnlineAccessed 2017].

[6] Brodkin, J.,*"Ajit Pai gives carriers free pass on privacy violations during FCC shutdown"*. https://arstechnica.com/tech-policy/2019/01/ajit-pai-gives-carriers-free-pass-on-privacy-violations-during-fcc-shutdown,2019,[Online accessed July 2019].

[7] DiGiacomo, J., *"2017 security breaches: Frequency and severity on the rise"* https://revisionlegal.com/data-breach/2017-security-breaches/, 2018, [Online accessed2018].

[8] Yeung, S. Ward, K. Madria, M. *"Ridesharing-Inspired Trip Recommendations"*. MDM '18. doi = 10.1109/MDM.2018.00019.

[9] Ward, K., Lin, D., and Madria, S., *"A Parallel Algorithm for Anonymization of Large-Scale Trajectory Data"* Transactions on Data Science '2019.

[10] Lin, D., Steiert, D., Ward, K., Squicciarini, A., and Fan, J., *"REMIND: Risk EstimationMechanism for Images in Network Distribution"*. CCS '2018.

[11] Lin, D., Bertino, E., Cheng, R., and Prabhakar, S., *"Location privacy in moving-object environments"* Trans. Data Privacy, 2009,Vol 2, pp. 21-46, ISSN 1888-5063.

[12] Jensen, C. S., Lin, D., and Ooi, B. C., *"Continuous clustering of moving objects"*. IEEETransactions on Knowledge and Data Engineering, 2007,Vol 19, pp. 1161-1174.

[13] Abul, O., Bonchi, F., and Nanni, M., *"Never Walk Alone : Uncertainty for Anonymityin Moving Objects Databases"*. ICDE, 2008.

[14] Bonchi, F. and Wang, H. W., *"Trajectory Anonymity in Publishing Personal Mobility-Data"*. SIGKDD, 2011,Vol 13, pp. 30-42.

[15] Chen, R., Fung, B. C. M., Mohammed, N., Desai, B. C., and Wang, K., *"Privacy-preserving trajectory data publishing by local suppression"* Information Sciences,2013,Vol 231, pp. 83-97, ISSN 0020-0255, doi:10.1016/j.ins.2011.07.035.

[16] Domingo-Ferrer, J. and Trujillo-Rasua, R., *"Microaggregation and permutation-based anonymization of movement data"*. Information Sciences, 2012,Vol 208, pp. 55-80, ISSN 00200255, doi:10.1016/j.ins.2012.04.015.

[17] Eldawy, A. and Mokbel, M., *"A demonstration of SpatialHadoop: an efficient mapreduce framework for spatial data"*. VLDB, 2013, Vol 6, pp. 1230-1233

[18] Apache Software Foundation. *"What is Apache Hadoop?"* http://hadoop.apache.org/, 2016,[Online Accessed 2017].

[19] Gedawy, H. K., *"Dynamic Path Planning and Traffic Light Coordination for Emergency Vehicle Routing"*. Carnegie Mellon University Thesis, 2009, pp. 1-9.

[20] Ghasemzadeh, M., Fung, B. C. M., Chen, R., and Awasthi, A., *"Anonymizing trajectory data for passenger flow analysis"*. Transportation Research Part C, 2014,39, pp.63-79, ISSN 0968-090X, doi:10.1016/j.trc.2013.12.003.

[21] Gruteser, M. and Grunwald, D., *"Anonymous Usage of Location-Based Services-Through Spatial and Temporal Cloaking"*. Proceedings of the 1st international conference on Mobile systems applications and services MobiSys ’03, 2003, pp. 31-42, doi:10.1145/1066116.1189037.

[22] Gurung, S., Lin, D., Jiang, W., Hurson, A., and Zhang, R., *"Traffic InformationPublication with Privacy Preservation"*. ACM Trans. Intell. Syst. Technol., 2014, Vol5, pp. 1-26, ISSN 2157-6904, doi:10.1145/2542666.

[23] Halevy, A. Y., Franklin, M. J., and Maier, D., *"TRUSTER: TRajectory Data Processtingon ClUSTERs"*. Dasfaa, 2009,Vol 3882, pp. 768-771, doi:10.1007/11733836.

[24] Han, P.I. and Tsai, H.P., *"SST: Privacy Preserving for Semantic Trajectories"*. 2015 16th IEEE International Conference on Mobile Data Management, 2015,Vol 2, pp.80-85, doi:10.1109/MDM.2015.18.

[25] He, X., Cormode, G., Machanavajjhala, A., Procopiuc, C. M., and Srivastava, D., *"DPT: Differentially Private Trajectory Synthesis Using Hierarchical Reference Systems"*. Proceedings of the VLDB Endowment, 2015,Vol 8, pp. 1154-1165, ISSN 21508097, doi:2150-8097/15/07.

[26]] Li, X., Li, W., Anselin, L., Rey, S., and Kochinsky, *"A MapReduce Algorithm to Create Contiguity Weights for Spatial Analysis of Big Data"*. in BigSpatial, 2014.

[27] Monreale, A., Pedreschi, D., Pensa, R. G., and Pinelli, F., *"Anonymity preserving sequential pattern mining"*. volume 22, 2014, ISBN 1050601491546, doi:10.1007/s10506-014-9154-6.

[28] Nergiz, M. E., Atzori, M., Saygin, Y., and Guc, B., *"Towards Trajectory Anonymization: A Generalization Based Approach"*. Transactions on Data Privacy, 2009,2(106), pp.47-75, doi:10.1145/1503402.1503413.

[29] Pensa, R. G., Monreale, A., Pinelli, F., and Pedreschi, D., *"Pattern-preserving k-anonymization of sequences and its application to mobility data mining"*. CEUR Workshop Proceedings, 2008,Vol 397, pp. 44-60, ISSN 16130073.

[30] Poulis, G., Skiadopoulos, S., Loukides, G., and Gkoulala-Divanis, A., *"Select-organize-anonymize: A framework for trajectory data anonymization"*. Proceedings IEEE 13th International Conference on Data Mining Workshops, ICDMW 2013,2013, pp. 867-874, doi:10.1109/ICDMW.2013.136.

[31] Poulis, G., Skiadopoulos, S., Loukides, G., and Gkoulalas, A., *"Apriori-based algorithms for km-anonymizing trajectory data"*. Transactions on Data Privacy, 2014,Vol7, pp. 165-194.

[32] Poulis, G., Skiadopoulos, S., Loukides, G., and Gkoulalas-Divanis, A., *"Distance-based km-anonymization of trajectory data*" Proceedings IEEE International Conference on Mobile Data Management, 2013,Vol 2, pp. 57-62, ISSN 15516245,doi:10.1109/MDM.2013.66

[33] Sankararaman, S., Agarwal, P., Molhave, T., Pan, J., and Boedihardjo, A., *"Model-Driven Matching and Segmentation of Trajectories"*. in SIGSPATIAL. 2013.

[34]] Ward, K., Lin, D., and Madria, S., *"Melt: Mapreduce-based efficient large-scale trajectory anonymization"*. in SSDBM '2017 doi:10.1145/3085504.3085581.

[35] Yarovoy, R., Bonchi, F., Lakshmanan, L. V. S., and Wang, W. H., *"Anonymizing moving objects: How to hide a MOB in a crowd?"*. Proceedings of the 12th International Conference on Extending Database Technology Advances in Database Technology(EDBT'09), 2009, pp. 72-83, doi:10.1145/1516360.1516370.

[36] Zhao, W., Ma, H., and He, Q., *"Parallel K-Means Clustering Based on MapReduce"* Cloud Computing. Springer Berlin Heidelberg, 2009, pp. 674-679.

[37] Executive Summary. *"Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013-2018"*. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html,2014, [Online Accessed 2018].

[38] Zixkhur, K., *"Location-Based Services"*. http://www.pewinternet.org/2013/09/12/location-based-services, 2013, [Online Accessed 2016].

[39] Francis, M., *"Future Telescope Array drives development of exabyte processing"* http://arstechnica.com/science/2012/04/future-telescope-array-drives-development-of-exabyte-processing/, 2014, [On-line accessed 2016]

[40] Deal, M., *"Census Bureau Reports 471,000 Workers Commute into Los Angeles County, Calif., Each Day"*. http://www.census.gov/newsroom/press-releases/2013/cb13-r13.html, 2016, [Online accessed 2018].

[41] Wang, W., Ying, L., and Zhang, J., *"On the Tradeoff between Privacy and Distortion in Differential Privacy"*. in KDD '2014 pp. 517-525.

[42] Zheng, Y., Zhang, L., Xie, X., and Ma, W.-Y., *"Mining interesting locations and travel sequences from gps trajectories"*. in âĂŸACM Press, '2009 pp. 791-800.

II. REMIND: RISK ESTIMATION MECHANISM FOR IMAGES IN NETWORK DISTRIBUTION

Dan Lin

Department of Computer Science

University of Missouri

Columbia, Missouri 65211

Tel: 573-884-6628

Email: lindan@missouri.edu

Doug Steiert

Department of Computer Science

Missouri University of Science and Technology

Rolla, Missouri 65409

Tel: 636-293-0940

Email: djsg38@mst.edu

Katrina Ward

Department of Computer Science

Missouri University of Science and Technology

Rolla, Missouri 65401-0050

Tel: 417-217-4273

Email: kjw26b@mst.edu

ABSTRACT

People constantly share their photos with others through various social media sites. With the aid of the privacy settings provided by social media sites, image owners can designate scope of sharing, e.g., close friends and acquaintances. However, even if the owner of a photo carefully sets the privacy setting to exclude a given individual who is not supposed to see the photo, the photo may still eventually reach a wider audience including those clearly undesired through unanticipated channels of disclosure, causing a privacy breach. Moreover, it is often the case that a given image involves multiple stakeholders who are also depicted in the photo. Due to various personalities, it is even more challenging to reach agreement on privacy settings for these multi-owner photos. In this work, we propose a privacy risk reminder system called REMIND, which estimates the probability that a shared photo may be seen by unwanted people - through the social graph - who are not included in the original sharing list. We tackle this problem from a novel angle by digging into the big data regarding image sharing history. Specifically, the social media providers possess a huge amount of image sharing information (e.g., what photos are shared with whom) of their users. By analyzing and modeling such rich information, we build a sophisticated probability model that efficiently aggregates image disclosure probabilities along different possible image propagation chains and loops. If the computed disclosure probability indicates high risks of privacy breach, a reminder is issued to the image owner to help revise the privacy settings (or at least inform the user about this accidental disclosure risk). The proposed REMIND system also has a nice feature of policy harmonization that helps resolve privacy differences in multi-owner photos. We have carried out a user study to validate the rationale of our proposed solutions and also conducted experimental studies to evaluate the efficiency of the proposed REMIND system.

Keywords: Image privacy, Sharing chain, Risk estimation, Probability model

1. INTRODUCTION

With social media affecting the way millions of people live their lives each day, we have assisted to an explosion of user contributed content online, especially images and media files. Some of the user-contributed photos may be harmless and effective for users' self-recognition and gratification. However, for many of these photos, the portrayed content affects individuals' social circles, as it either explicitly includes multiple users or it relates to users other than the original poster (e.g., a child or a house/location). To further complicate this issue, photos may be leaked or disclosed with an audience larger than expected, for both the image owner and its stakeholders.

Although many social websites provide privacy configurations that allow the users to specify to whom they would like to share the photos with, through the possible re-sharing via the friends to friends, it is often the case that the photos reach a wider audience than that in the original sharing list of the photo owner. Figure 1 illustrates a simple example of the privacy breach caused by such image sharing propagation.

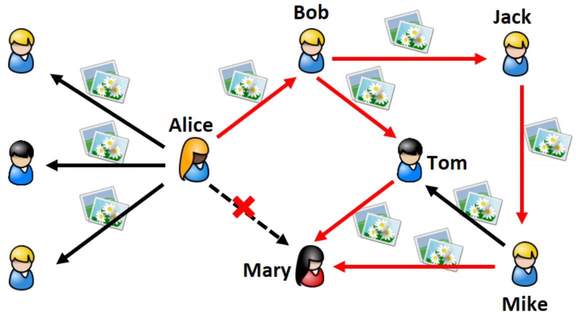


Figure 1. An Example of Privacy Breach Due to Image Propagation

Alice wanted to share her images with some her friends but not Mary. However, one of Alice's friends, Bob, forwarded Alice's images to his friends whose social circles include Mary, and eventually Mary views Alice's images although Mary is not supposed to. Such privacy risks caused by sharing from friends to friends have been aware by many (3; 25; 27; 36). Some propose monitoring approaches to check the privacy violation during each sharing event (3; 27). Most employ (5; 13; 33; 37; 49) 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 as we will elaborate shortly in this work.

Another critical factor that could cause a privacy breach is the difference among privacy preferences of people depicted in the same photo. Due to the variety of personalities, users may drastically disagree on the scope of sharing for a given co-owned image causing some significant conflicts. In some instances, it can be courtesy that these users may personally discuss which photos can be posted and by whom so that there are no conflicts of interest, but that takes time and is not often the route pursued. An increasing number of recent works (50) have analyzed how to address the policy conflict, by considering every user's prior privacy preferences of sharing or through semi-automated resolution mechanisms. These existing works are usually based on fuzzy logic while we aim to provide clear evidence of chances of privacy breach.

In this work, unlike any existing works, we tackle the privacy risk estimation problem from a novel angle by digging into the big data regarding image sharing history. Specifically, the social media providers possess a huge amount of image sharing information (e.g., what photos are shared with whom) of their users. In fact, even some external websites (32) have provided tools to maintain statistics of the sharing propagation throughout the social networks.

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 present the users with direct evidence of potential scope of sharing, i.e., the probability of unwanted people to access one's photos. Specifically, if the computed disclosure probability indicates high risks of privacy breach, a reminder will be issued to the image owner to help revise the privacy settings. Users then have the opportunity to make informed decisions when setting their privacy preferences. For example, our work would remind Alice that sharing with Bob could result in 90% chance that Mary would see the photo as well. Based on such a high disclosure probability, it is very likely that Alice would remove Bob from her initial sharing list, and hence avoid the potential privacy breach.

Carrying the spirits of our goal, the proposed system is named REMIND (Risk Estimation Mechanism for Images in Network Distribution). The REMIND system has the following novel contributions:

- The REMIND system provides users a quantitative and easier way to directly evaluate the potential consequence of sharing. It "nudges" users about the risk of sharing the image with certain people and remind the owners of the photos about users that could be explicitly excluded to avoid over sharing. The goal is to reduce the risk of privacy breach that could result from the image propagation in the social network.
- Underlying the REMIND system, we propose a sophisticated probability model that models the image sharing history. It is very challenging to calculate and aggregate the disclosure probabilities caused by various sharing paths especially loops in convoluted social networks. To overcome this, we design an efficient probability serialization algorithm that ensures each node in the related social circle to be visited and calculated only once.
- The REMIND system also has a nice feature called policy harmonization, which calculates *image disclosure matrix* to help resolve differences in the privacy preferences of people depicted in the same photo.
- We have carried out a user study to validate the rationale of our proposed solutions and also conducted experimental studies in real-life social networks to evaluate the effectiveness and efficiency of the proposed REMIND system.

In addition, it is worth noting that while we present our models with images as a reference content type, any co-owned or co-managed piece of content in an online social setting could take advantage of REMIND, with no significant differences.

The remaining of the paper is organized as follows. Section 2 introduces the problem statement. Section 3 elaborates the proposed REMIND system. Section 4 reports the experimental study and Section 5 concludes this work.

2. PROBLEM STATEMENT

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

Definition 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 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 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 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.

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 2 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 that for an image with multiple users, e.g., u_{o_1}, u_{o_2}, ..., u_{o_n}, 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

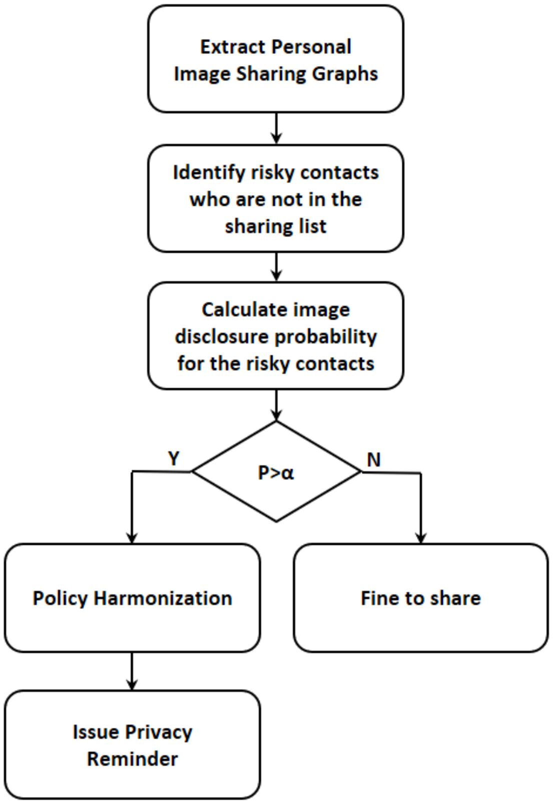

Extract Personal Image Sharing Graphs
Identify risky contacts who are not in the sharing list
Calculate image disclosure probability for the risky contacts
P>α
Y
N
Policy Harmonization
Fine to share
Issue Privacy Reminder

Figure 2. An Overview of REMIND System

through which user who are 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.

3.1. PROPAGATION CHAIN MODELS

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 *(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, $\Psi = \{\psi_{u_o:u_i \to u_j}\}$ where $\psi_{u_o:u_i \to u_j}$ denote the number of images originally owned by u_o and are shared by user u_i with u_j.

Figure 3 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). 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

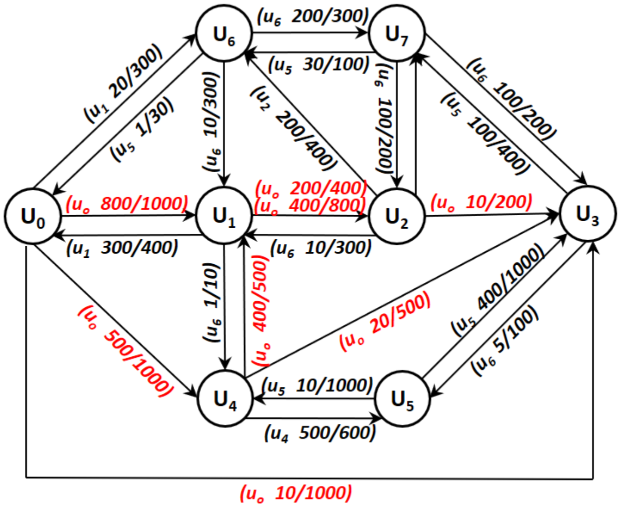


Figure 3. An Example of Image Sharing Graph

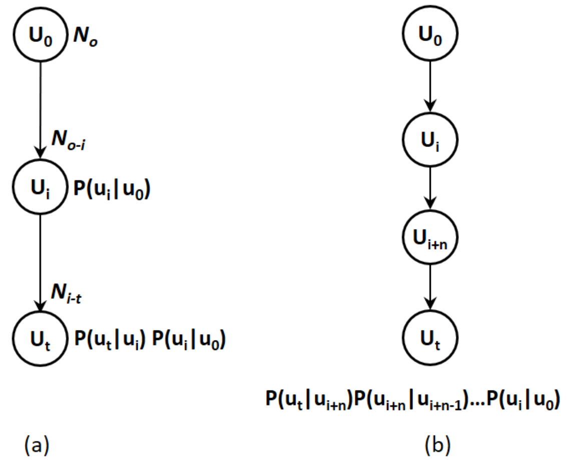


Figure 4. Single Photo Propagation Chains

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(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 1.

$$P_{u_o \Rightarrow u_t} = P_{u_o \Rightarrow u_i} \cdot P_{u_o}(u_t|u_i) \tag{1}$$

In Equation 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(b). The probability that u_o's photos may reach the target user u_t via multiple routes of sharing can be computed by Equation 2.

$$P_{u_o \Rightarrow u_t} = P_{u_o \Rightarrow u_i} \cdot P_{u_o}(u_t|u_{i+n}) \prod_{j=1}^{n} P_{u_o}(u_{i+j}|u_{i+j-1}) \tag{2}$$

At the end, we extend the probability formula to the generic scenarios (as shown in Figure 5) 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 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.

$$\begin{aligned} P_{u_o \Rightarrow u_t} &= 1 - \prod_{k=1}^{m}(1 - P_{c_k}) \\ &= 1 - \prod_{k=1}^{m}(1 - P_{u_o \Rightarrow u_k} \cdot P(u_t|u_k)) \end{aligned} \tag{3}$$

In Equation 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. Correspondingly, $1 - P_{c_k}$ is the probability that u_t will not obtain u_o's photos from the chain c_k. Then,

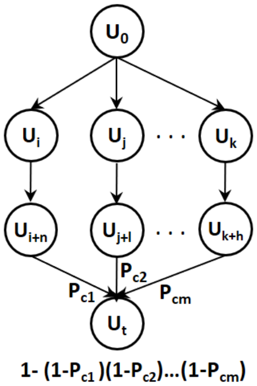


Figure 5. A Generic Photo Propagation Model

$\prod_{k=1}^{m}(1 - P_{c_k})$ 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.

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 practise?" 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 5.

Definition 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) $\forall\ u_j \in \Xi_o, \exists \psi_{u_o:u_i \rightarrow u_j}$.

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 3. We can extract the personal image sharing graph for u_o as shown in Figure 6.

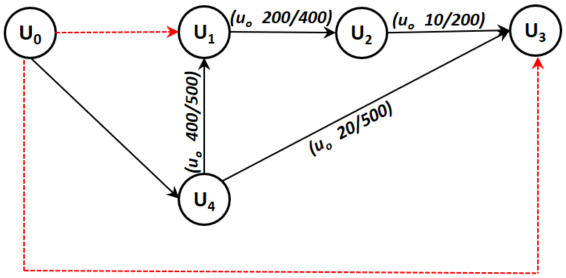


Figure 6. User u_o's Personal Image Sharing Graph

Assume that the image owner u_o shares a new photo with only u_4. The red dotted arrows in Figure 6 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

$P_{u_o \Rightarrow u_2} = P_{u_o \Rightarrow u_1} \times \frac{200}{400}$ = 0.8×0.5=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_1'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 6).

Definition 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 ensure 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.

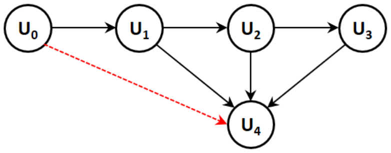


Figure 7. Sharing Scenario Case 1

To conduct the probability serialization, we first analyze various sharing scenarios and classify them into two main categories as shown in Figure 7 and Figure 8, 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 7, 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 $u_1 \succ u_2 \succ u_3 \succ u_4$.

Case 2 depicts the scenario when there is a sharing loop. Specifically, user u_2 forwards u_o's photos to u_3, and u_3 forwards the photos to u_4. Without knowing that u_2 already has seen u_o's photos, u_4 forwards the photos received from u_3 to u_2, thus creating a sharing loop. In this case, even though u_4 is also u_2's immediate parent, u_2's disclosure probability does not depend on u_4 since u_4 is sharing what u_2 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

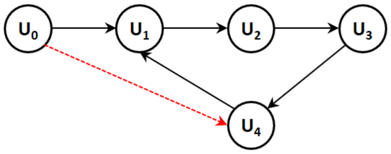


Figure 8. Sharing Scenario Case 2

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 6):

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. $Checkingcurrentnodeinthepriorityqueue$: Then, we examine the node in the priority queue one by one. Let u_i denote the node in the priority queue 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. $Probabilitypropagation$: 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 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 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. $BreakingtheLoopbetween\ u_i and u_c$: 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 9. 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 10 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, the probability that u_1 views the photo is 1. We start evaluating the first node in the priority

ALGORITHM 6: Probability Calculation Algorithm

```
Data: Input:
mage sharing graph Result: Output:
Disclosure probabilities of $u_o$'s friends Extract $u_o$'s personal image sharing
  (PIS) graph
for each user $u_i$ in $u_o$'s sharing list do
    Initialize Prob[$u_i$] ← 1
    Add $u_i$ to priority_queue
end
while priority_queue is not empty and $U_o^-$ is not computed do
    $u_i$ ← priority_queue.pop()
    for each parent $u_j$ of $u_i$ do
        $P_{ij}$ ← chain probability (Equation 1)
        Prob[$u_i$] ← Prob[$u_i$]*(1-$P_{ij}$)
    end
    if all of $u_i$'s parents are computed then
        Prob[$u_i$] ← 1-Prob[$u_i$]
        Remove $u_i$ from priority_queue
    end
    for each $u_i$'s direct friend $u_c$ do
        if $u_c$ is not in priority_queue then
            Add $u_c$ to priority_queue
        end
        else
            Break_Loop between $u_i$ and $u_c$
        end
    end
end
```

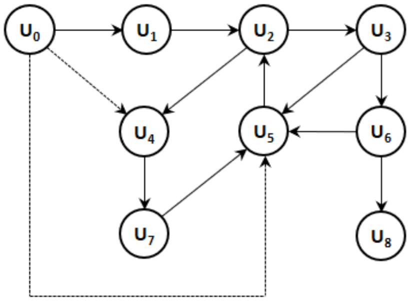


Figure 9. An Example of Sharing Graph

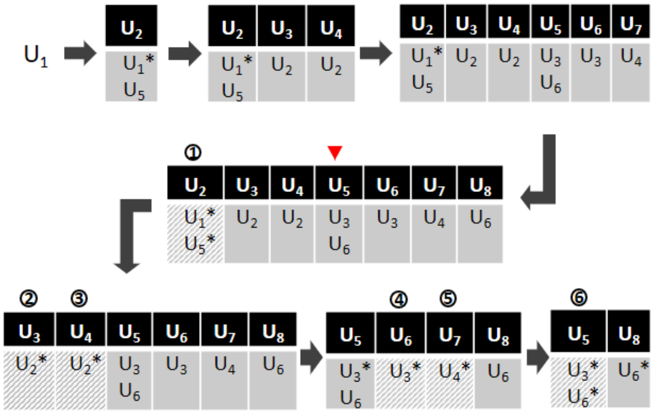


Figure 10. An Example of Probability Serialization

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.

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 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 $u_{i_j} \in \bigcup_{w=1}^{n} U_w^+ / \{u_1, ...u_n\}$.

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

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:

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

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:

$$\text{Pol'}_{u1} = \{u_1, u_2, u_3, u_4\}$$
$$\text{Pol'}_{u2} = \{u_1, u_2, u_3, u_4\}$$
$$\text{Pol'}_{u3} = \{u_1, u_2, u_3, u_7\}$$

3.4. POTENTIAL ENHANCEMENT

Our system can be further enhanced to provide more fine-grained and precise privacy alerts by pre-processing statistic information collected by the social media provider. Specifically, images can be first clustered based on their content (49; 58). Then, the image sharing statistics will be linked to each particular type of images of a user instead of all. 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 handle such additional sharing information only requires a minor modification of the previously discussed probability calculation algorithm. The minor change lies in

Table 1. User Response to Different Scenarios

Privacy Breach Probability	90%	50%	10%
Scenario 1 (Single-owner photo, undesired disclosure to family)	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%

the input parameter to the algorithm. With the fine-grained sharing information, we update Ψ by adding the image type information, i.e., $\Psi = \{\psi_{(u_o,typ):u_i \to u_j}\}$. Next, when it comes to the probability calculation, only those Ψ which contain the same image type as the one under consideration will be considered.

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.

4.1. USER STUDY

This user study aims to investigate how a typical user would react if the user knows that sharing with someone may cause a privacy breach with different disclosure probabilities. 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.

4.1.1. Setup and Survey. The user study involves 104 users, including approximately 30% female and 70% male. Their ages range from 18 to over 50. All of the 104 users have at least one social media account with about 71% of them have more than 2 accounts. When asked how often they shared images on social media, more than half said they share regularly with 6% admitting they share images every day, and over half of the participants have between 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 good percentage (28%) of users who may not be aware of privacy risks which makes it important to have a system that can help remind these users about the potential privacy breach.

Moreover, even users who set up the privacy configurations during the photo sharing, they still do not have the knowledge what would be the final audience of their images as their friends may re-share the received images. To get an idea of how much of an impact that 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.

4.1.2. Scenarios. After surveying the participants' common practices in the social media sites, we next present them three different scenarios to learn how they may react if they know the probabilities of their photos being seen by unwanted people.

Scenario 1: *Suppose that you are going to post a picture of yourself doing extreme sports such as surfing. You know your parents (or close family members) always try to convince you to give up these extreme sports for safety concerns, so you are not planning* *to share the photo with your parents (or close family members), but only with your surfing club members. However, one of the surfing club members, Erica, is actually a colleague of your parents' (or close family members') friends.*

Scenario 2: *Suppose that you are going to post a picture of yourself in a party and share it with some of your close friends. One of your friends in your sharing list, Erica, is also a friend of Bob who you do not wish to see your photo for some reasons.*

Scenario 3: *Suppose that you are going to post a picture of you and your best friend Mary in funny costumes (such as shown in the photos) with your family members. Mary does not wish her manager to see this. However, your cousin, Erica, knows some of Mary's colleagues who may know Mary's manager.*

The first two scenarios are regarding single-owner photos and the last scenario is about multi-owner photo. 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 Erica 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 Erica. Table 1 reports the percentage of participants who responded positively that they would change their initial privacy settings as suggested. From the table, we can clearly observe an increasing trend of privacy concerns when the relationship between the photo owner and the unwanted person becomes loose. Specifically, 61% of participants said they would not share with Erica 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 the photo owner's manager. Such results essentially conform with most people's common practice as we are less concerned about our privacy within the family than in the working environments. The second observation is that the percentage of participants who would like 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 are still a high percentage (68%) of people would like to prevent their manager from seeing the photo even there is only 10% chance of disclosure. This is understandable since the privacy risk with respect to the manager may lead to severe impact on the career development, and hence people would not want to take much risk. Based on these responses, we envision that our proposed REMIND system would have a good number of customers when adopted by the social media providers.

To be even more clear about the 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. We have 73% of participants responded that they are interested in using this kind of system. It is interesting to note that this percentage matches the percentage of participants who answered that they usually set up privacy settings in the social networks. It indicates that users who are concerned about their privacy are very likely to adopt our proposed REMIND system to further enhance their privacy protection. For people who do not care about privacy issues much, we feel that our REMIND system could play an important role in gradually educating them the importance of privacy protection via constant privacy breach alerts. To sum up, there is a large percentage of people who may benefit from our proposed REMIND system to gain better privacy protection.

4.2. EFFICIENCY STUDY

After the user study which validates the usefulness and potential market of our proposed REMIND system, we 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 (28). Since current social mediate sites only release the social network connections as shown in Table 2, but not the image sharing statistics yet, we simulate a variety of scenarios in terms of image sharing on these real social networks as described in Definition 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 group of his/her friends to share certain percentage of the photos. 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

Table 2. 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

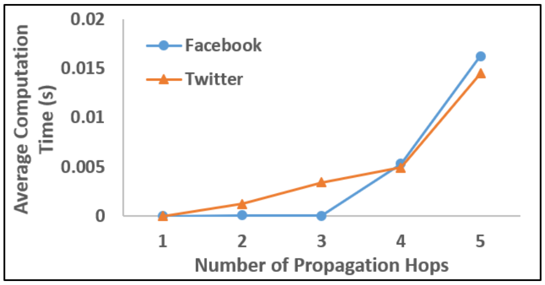


Figure 11. Effect of the Number of the Hops

convergence as a photo may becomes less interesting to people who are farther away from the photo owner. The following subsections elaborate the detailed experimental settings for each round of experiments and report the corresponding results.

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 5 hops is based on the "six degrees of separation" theory (55) 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.

Figure 11 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 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.

4.2.2. 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 12, the average time to calculate the disclosure probability for a user in both datasets can be done in just a few milliseconds. 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.

4.2.3. Effect of the Sharing Convergence Speed. Finally, we 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 third hop. Figure 13 shows the average probability calculation time for each user. Observe that the calculation time decreases when

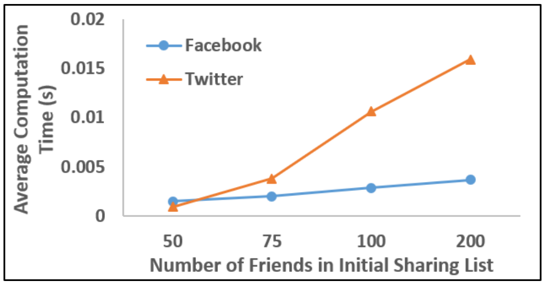


Figure 12. Effect of the Size of the Initial Sharing List

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. Also, the smaller sharing scope, the lower the average disclosure probabilities.

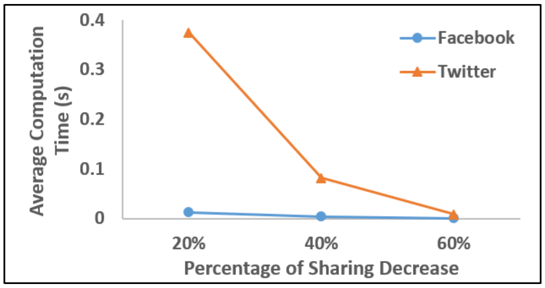


Figure 13. Effect of Sharing Convergence Speed

5. CONCLUSION

In this paper, we present 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 both user studies and performance studies to validate the effectiveness and efficiency of our approach.

REFERENCES

[1] Watts, D. J., *"Six Degrees: The Science of a Connected Age, W. W. Norton & Company"*; Reprint edition, 2004.

[2] Leskovec, J., *"Stanford Large Network Dataset Collection"*. https://snap.stanford.edu/data, 2015, [Online Accessed 2019].

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

[4] Pensa, R. G. and Di Blasi, G., *"A semi-supervised approach to measuring user privacy in online social networks"*, in T. Calders, M. Ceci, and D. Malerba, editors, Discovery Science, Cham, 2016 pp. 392-407.

[5] Squicciarini, A. C., Lin, D., Sundareswaran, S., and Wede, J., *"Privacy policy inference of user-uploaded images on content sharing sites"*, IEEE Transactions on Knowledge and Data Engineering, 2015, Vol 27, pp. 193-206.

[6] Squicciarini, A. C., Sundareswaran, S., Lin, D., and Wede, J., *"A3p: Adaptive policy prediction for shared images over popular content sharing sites"*, in Proceedings of the 22Nd ACM Conference on Hypertext and Hypermedia, 2011. pp. 261-270.

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

[8] Squicciarini, A., Karumanchi, S., Lin, D., and Desisto, N., *"Identifying hidden social circles for advanced privacy configuration"*, Computers and Security, 2014, Vol 41, pp. 40-51.

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

[10] Mariotti, T., *"How can we track when someone shares through the facebook share (not like) button?"*, in https://www.quora.com/How-can-we-track-when-someoneshares-through-the-Facebook-Share-not-Like-button, 2012.

[11] Kafali, O., Gunay, A., and Yolum, P., *"Detecting and predicting privacy violations in online social networks"*, Distributed and Parallel Databases, 2014, Vol 32, pp. 161-190.

[12] Akcora, C. G., Carminati, B., and Ferrari, E., *"Risks of friendships on social networks"*, in âĂŸData Mining (ICDM), 2012 IEEE 12th International Conference on, '2012 pp. 810-815.

[13] N.Kokciyan and Yolum, P., *"Priguard: A semantic approach to detect privacy violations in online social networks"*, IEEE Transactions on Knowledge and Data Engineering, 2016, Vol 28, pp. 2724-2737.

[14] Rao, P., Lin, D., Bertino, E., Li, N., and Lobo, J., *"Fine-grained integration of access control policies"*, Computers and Security, 2011, Vol 30, pp. 91-107, ISSN 0167-4048, doi:https://doi.org/10.1016/j.cose.2010.10.006, special Issue on Access Control Methods and Technologies.

[15] Such, J. M. and Criado, N., *"Resolving multi-party privacy conflicts in social media"*, IEEE Transactions on Knowledge and Data Engineering, 2016, Vol 28, pp. 1851-1863.

[16] Bonneau, J., Anderson, J., and Danezis, G., *"Prying data out of a social network"*, in Social Network Analysis and Mining, 2009. ASONAM '09. International Conference on Advances in, IEEE, 2009 pp. 249-254.

[17] Adu-Oppong, F., Gardiner, C. K., Kapadia, A., and Tsang, P. P., *"Social circles: Tackling privacy in social networks"*, in Symposium on Usable Privacy and Security (SOUPS), 2008.

[18] Mazzia, A., LeFevre, K., and Adar, E., *"The pviz comprehension tool for social network privacy settings"* in Proceedings of the Eighth Symposium on Usable Privacy and Security, ACM, 2012 p. 13.

[19] Klemperer, P., Liang, Y., Mazurek, M., Sleeper, M., Ur, B., Bauer, L., Cranor, L. F., Gupta, N., and Reiter, M., *"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, ACM, 2012 pp. 377-386.

[20] Fang, L. and LeFevre, K., *"Privacy wizards for social networking sites"*, in Proceedings of the 19th international conference on World wide web, ACM, 2010. pp. 351-360.

[21] Spyromitros-Xioufis, E., Papadopoulos, S., Popescu, A., and Kompatsiaris, Y., *"Personalized privacy-aware image classification"*, in Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval, ACM, 2016. pp. 71-78.

SECTION

3. CONCLUSIONS

More than ever before, our growing population is using all sorts of devices to connect and communicate with people from all over the world. They are sharing their locations, their images, and using hand held computers to find the things they are looking for, get directions, and make recommendations. All of this information can easily lead to the loss of privacy for many users as we have seen in recent years with data breach after data breach. Controlling what information is released is already difficult, but also controlling how far it goes once online is impossible.

The potential benefits are impossible to ignore, however. This information could be used to plan critical infrastructures, route emergency vehicles, improve conveniences, and so much more. The volume of data being produced is overwhelming due to current algorithms however, and require a new approach that can make the most use of the data while also maintaining the privacy of the users producing it.

This collection of work has provided improvements that not only can handle large volumes of data quickly, but also maintain the privacy of all users associated with the data. This work is easily deployable to real world environments and brings our social nature and technology closer to both being safe, and helping the greater needs of society. As a whole, it has shown that using data for real world improvements is not only possible, but that it can be done without sacrificing safety.

APPENDIX A.

AUTHOR PUBLICATIONS LIST

The following are all current publications by the author for quick reference.

1. Katrina Ward., Dan Lin, Sanjay Madria. 2019 *MELT: Mapreduce-based Efficient Large-Scale Trajectory Anonymization*. ACM Transactions on Data Science(TDS)' 19.

2. Dan Lin, Douglas Steiert, Katrina Ward, Anna Squicciarini, Jianping Fan. 2019. *Risk Estimation Mechanism for Images in Network Distribution*, Under Submission to ACM Conference on Computer and Communications Security.

3. Katrina Ward, Dan Lin, Sanjay Madria. 2017. *MELT: Mapreduce-based Efficient Large-scale Trajectory Anonymization*. In Proceedings of SSDBM '17. Chicago, IL, USA. June 27-29, 2017, 6 pages. DOI: http://dx.doi.org/10.1145/3085504.3085581

4. Yeung, San. Ward, Katrina. Madria, Sanjay. 2018. *Ridesharing-Inspired Trip Recommendations*. In Proceedings of MDM âĂŹ18. Aalbord, Denmark. 16 July 2018. DOI: 10.1109/MDM.2018.00019

5. Jennifer L. Leopold, Chaman L. Sabharwal, Katrina J. Ward. 2015. *Spatial relations between 3D objects: The association between natural language, topology, and metrics, Journal of Visual Languages & Computing*, Volume 27, April 2015, Pages 29-37, ISSN 1045-926X, Distributed Multimedia Systems

6. Jennifer L. Leopold, Chaman L. Sabharwal, Katrina J. Ward. 2014 *Spatial Relations Between 3D Objects: The Association Between Natural Language, Topology, and Metrics*. DMS 2014: 241-249

APPENDIX B.

MELT COMPLEXITY ANALYSIS EXPANDED

In the MELT paper (the third paper) in this thesis, the author presented a summary of the complexity analysis of the algorithms for the system. In this appendix, the author presents more detail to the complexity analysis to facilitate better understanding.

- $Round1Analysis$: For the first round, recall that we:

  1. Calculate which partition a trajectory belongs to
  2. Output Trajectories and roads with their assigned partition
  3. Cluster identical trajectories and roads

  For this analysis:

  $$c = \text{Average nodes in a trajectory}$$
  $$n = \text{Number of trajectories in data}$$
  $$x = \text{Number of map partitions the data will be divided into}$$

  We can say that the time for round one is:

  $$\frac{cn+n}{x}$$

  where cn represents looking at each node in a trajectory, n represents outputting each trajectory, assuming none are grouped for worst case, and x represents each partition has its own reducer working in parallel.

- $Round2Analysis$: For the second round, we:

  1. Identify infrequent roads
  2. Remove infrequent roads from trajectories
  3. Divide data into branches and output

For this analysis, we add the following terms:

$$m = \text{Number of roads in data}$$

$$i = \text{Number of mappers running in parallel}$$

$$j = \text{Number of branches to divide data into}$$

$$k = k \text{ Anonymity privacy threshold}$$

With this, we add the following to our complexity:

$$\frac{m}{i} + \frac{cn}{i}$$

where $\frac{m}{i}$ represents the time to identify infrequent roads among all the roads and $\frac{cn}{i}$ represents the time to remove those infrequent roads from the trajectories since we look at each road in them. This assumes the worst case of no duplicate trajectories at this point. Also at worst case, we can assume only one reducer is operating and no more than k similar trajectories in any group, or branch. This means that one reducer is processing the maximum number of branches itself.

$$\frac{n * \frac{n}{k}}{j}$$

The above equation represents the time for each trajectory to be compared with each other cluster of similar trajectories and assuming it does not match and creates its own. j is arbitrary and constant, so the equation can be reduced to $\frac{n^2}{k}$ with an I/O cost of $\frac{n}{k}$.

Therefore in round 2, we see:

$$\frac{m}{i} + \frac{cn}{i} + \frac{n^2}{k} + \frac{n}{k}$$

We assume there are n infrequent roads, meaning every trajectory will need to be changed. In the third term, realistically, there is never a single reducer working and the chances of even clusters and every trajectory checking every cluster and reaching the end every time is very small. It would be considered an extreme outlier case. In practice, this term is reduced to $\frac{n}{k}$ to be realistic. Therefore, the actual round two result to be realistic would be:

$$\frac{m}{i} + \frac{cn}{i} + \frac{n}{k} + \frac{n}{k}$$

- $Round3Analysis$: In the third round, we build small cluster trees, locate the cluster each trajectory belongs to, and output the final anonymized dataset. We start with:

$$\frac{nlog(cn)}{jx} + \frac{cn}{x} + \frac{n}{k}$$

where the first term represents finding a cluster for each trajectory to be assigned to. Since we look at individual nodes in the cluster, we assume we look at all nodes for worst case. This task is split up over j branches being processed in batches over x reducers in parallel. The second term shows us the time to calculate the representative trajectory for those clusters looking at the nodes, and the last term represents outputting the anonymized dataset assuming all trajectories are split evenly into k clusters and all clusters meet our privacy threshold so no data is lost.

- $FinalSummary$: From these three rounds, we have:

$$\text{Round } 1(\frac{cn+n}{x}) + Round2(\frac{m}{i} + \frac{cn}{i} + \frac{n}{k} + \frac{n}{k}) + Round3(\frac{nlog(cn)}{jx} + \frac{cn}{x} + \frac{n}{k})$$

When we assume constants are 1 and toss them, we are left with:

$$8n + nlog(n) \rightarrow O(n + nlog(n))$$

REFERENCES

[1] Abul, O., Bonchi, F., and Nanni, M., ‘N ever W alk A lone : Uncertainty for Anonymity in Moving Objects Databases,’ ICDE, 2008.

[2] Adu-Oppong, F., Gardiner, C. K., Kapadia, A., and Tsang, P. P., ‘Social circles: Tackling privacy in social networks,’ in ‘Symposium on Usable Privacy and Security (SOUPS),’ 2008 .

[3] Akcora, C. G., Carminati, B., and Ferrari, E., ‘Risks of friendships on social networks,’ in ‘Data Mining (ICDM), 2012 IEEE 12th International Conference on,’ 2012 pp. 810–815.

[4] Bonchi, F. and Wang, H. W., ‘Trajectory Anonymity in Publishing Personal Mobility Data,’ SIGKDD, 2011, **Vol 13**, pp. 30–42.

[5] Bonneau, J., Anderson, J., and Danezis, G., ‘Prying data out of a social network,’ in ‘Social Network Analysis and Mining, 2009. ASONAM’09. International Conference on Advances in,’ IEEE, 2009 pp. 249–254.

[6] Brodkin, J., ‘Ajit Pai gives carriers free pass on privacy violations during FCC shutdown,’ https://arstechnica.com/tech-policy/2019/01/ajit-pai-gives-carriers-free-pass-on-privacy-violations-during-fcc-shutdown, 2019, [Online accessed July 2019].

[7] Burke, M., ‘Miami teen commits suicide in two-hour long facebook live video, the third in as many weeks,’ http://www.nydailynews.com/news/national/miami-teen-commits-suicide-two-hour-long-facebook-live-video-article-1.2955175, 2018, [Online Accessed 2019].

[8] Chen, R., Fung, B. C. M., Mohammed, N., Desai, B. C., and Wang, K., ‘Privacy-preserving trajectory data publishing by local suppression,’ Information Sciences, 2013, **Vol 231**, pp. 83–97, ISSN 0020-0255, doi:10.1016/j.ins.2011.07.035.

[9] Deal, M., ‘Census Bureau Reports 471,000 Workers Commute into Los Angeles County, Calif., Each Day,’ http://www.census.gov/newsroom/press-releases/2013/cb13-r13.html, 2016, [Online accessed 2018].

[10] DiGiacomo, J., ‘2017 security breaches: Frequency and severity on the rise,’ https://revisionlegal.com/data-breach/2017-security-breaches/, 2018, [Online accessed 2018].

[11] Domingo-Ferrer, J. and Trujillo-Rasua, R., ‘Microaggregation- and permutation-based anonymization of movement data,’ Information Sciences, 2012, **Vol 208**, pp. 55–80, ISSN 00200255, doi:10.1016/j.ins.2012.04.015.

[12] Eldawy, A. and Mokbel, M., ‘A demonstration of SpatialHadoop: an efficient mapreduce framework for spatial data,’ VLDB, 2013, **Vol 6**, pp. 1230–1233.

[13] Fang, L. and LeFevre, K., ‘Privacy wizards for social networking sites,’ in ‘Proceedings of the 19th international conference on World wide web,’ ACM, 2010. pp. 351–360.

[14] Foundation", A. S., ‘What is Apache Hadoop?’ http://hadoop.apache.org/, 2016, [Online Accessed 2017].

[15] Francis, M., ‘Future Telescope Array drives development of exabyte processing,’ http://arstechnica.com/science/2012/04/future-telescope-array-drives-development-of-exabyte-processing/, 2014, [Online accessed 2016].

[16] Gedawy, H. K., ‘Dynamic Path Planning and Traffic Light Coordination for Emergency Vehicle Routing,’ Carnegie Mellon University Thesis, 2009, pp. 1–9.

[17] Ghasemzadeh, M., Fung, B. C. M., Chen, R., and Awasthi, A., ‘Anonymizing trajectory data for passenger flow analysis,’ Transportation Research Part C, 2014, **39**, pp. 63–79, ISSN 0968-090X, doi:10.1016/j.trc.2013.12.003.

[18] Granville, K., ‘Facebook-Cambridge Analytica Explained,’ https://www.nytimes.com/2018/03/19/technology/facebook-cambridge-analytica-explained.html, 2018, [Online Accessed 2017].

[19] Gruteser, M. and Grunwald, D., ‘Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking,’ Proceedings of the 1st international conference on Mobile systems applications and services MobiSys 03, 2003, pp. 31–42, doi:10.1145/1066116.1189037.

[20] Gurung, S., Lin, D., Jiang, W., Hurson, A., and Zhang, R., ‘Traffic Information Publication with Privacy Preservation,’ ACM Trans. Intell. Syst. Technol., 2014, **Vol 5**, pp. 1–26, ISSN 2157-6904, doi:10.1145/2542666.

[21] Halevy, A. Y., Franklin, M. J., and Maier, D., ‘TRUSTER:TRajectory Data Processting on ClUSTERs,’ Dasfaa, 2009, **Vol 3882**, pp. 768–771, doi:10.1007/11733836.

[22] Han, P.-I. and Tsai, H.-P., ‘SST: Privacy Preserving for Semantic Trajectories,’ 2015 16th IEEE International Conference on Mobile Data Management, 2015, **Vol 2**, pp. 80–85, doi:10.1109/MDM.2015.18.

[23] He, X., Cormode, G., Machanavajjhala, A., Procopiuc, C. M., and Srivastava, D., ‘DPT: Differentially Private Trajectory Synthesis Using Hierarchical Reference Systems,’ Proceedings of the VLDB Endowment, 2015, **Vol 8**, pp. 1154–1165, ISSN 21508097, doi:2150-8097/15/07.

[24] Jensen, C. S., Lin, D., and Ooi, B. C., ‘Continuous clustering of moving objects,’ IEEE Transactions on Knowledge and Data Engineering, 2007, **Vol 19**, pp. 1161–1174.

[25] Kafali, O., Gunay, A., and Yolum, P., 'Detecting and predicting privacy violations in online social networks,' Distributed and Parallel Databases, 2014, **Vol 32**, pp. 161–190.

[26] Klemperer, P., Liang, Y., Mazurek, M., Sleeper, M., Ur, B., Bauer, L., Cranor, L. F., Gupta, N., and Reiter, M., '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,' ACM, 2012 pp. 377–386.

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

[28] Leskovec, J., 'Stanford Large Network Dataset Collection,' https://snap.stanford.edu/data, 2015, [Online Accessed 2019].

[29] Li, X., Li, W., Anselin, L., Rey, S., and Kochinsky, 'A MapReduce Algorithm to Create Contiguity Weights for Spatial Analysis of Big Data,' in 'BigSpatial,' 2014 .

[30] Lin, D., Bertino, E., Cheng, R., and Prabhakar, S., 'Location privacy in moving-object environments,' Trans. Data Privacy, 2009, **Vol 2**, pp. 21–46, ISSN 1888-5063.

[31] Lin, D., Steiert, D., Ward, K., Squicciarini, A., and Fan, J., 'REMIND: Risk Estimation Mechanism for Images in Network Distribution,' CCS, 2018.

[32] Mariotti, T., 'How can we track when someone shares through the facebook share (not like) button?' in 'https://www.quora.com/How-can-we-track-when-someone-shares-through-the-Facebook-Share-not-Like-button,' 2012 .

[33] Mazzia, A., LeFevre, K., and Adar, E., 'The pviz comprehension tool for social network privacy settings,' in 'Proceedings of the Eighth Symposium on Usable Privacy and Security,' ACM, 2012 p. 13.

[34] Monreale, A., Pedreschi, D., Pensa, R. G., and Pinelli, F., *Anonymity preserving sequential pattern mining*, volume 22, 2014, ISBN 1050601491546, doi:10.1007/s10506-014-9154-6.

[35] Nergiz, M. E., Atzori, M., Saygin, Y., and Guc, B., 'Towards Trajectory Anonymization A Generalization Based Approach,' Transactions on Data Privacy, 2009, **2**(106), pp. 47–75, doi:10.1145/1503402.1503413.

[36] N.Kokciyan and Yolum, P., 'Priguard: A semantic approach to detect privacy violations in online social networks,' IEEE Transactions on Knowledge and Data Engineering, 2016, **Vol 28**, pp. 2724–2737.

[37] Pensa, R. G. and Di Blasi, G., 'A semi-supervised approach to measuring user privacy in online social networks,' in T. Calders, M. Ceci, and D. Malerba, editors, 'Discovery Science,' Cham, 2016 pp. 392–407.

[38] Pensa, R. G., Monreale, A., Pinelli, F., and Pedreschi, D., ‘Pattern-preserving k-anonymization of sequences and its application to mobility data mining,’ CEUR Workshop Proceedings, 2008, **Vol 397**, pp. 44–60, ISSN 16130073.

[39] Poulis, G., Skiadopoulos, S., Loukides, G., and Gkoulala-Divanis, A., ‘Select-organize-anonymize: A framework for trajectory data anonymization,’ Proceedings - IEEE 13th International Conference on Data Mining Workshops, ICDMW 2013, 2013, pp. 867–874, doi:10.1109/ICDMW.2013.136.

[40] Poulis, G., Skiadopoulos, S., Loukides, G., and Gkoulalas, A., ‘Apriori-based algorithms for k m -anonymizing trajectory data,’ Transactions on Data Privacy, 2014, **Vol 7**, pp. 165–194.

[41] Poulis, G., Skiadopoulos, S., Loukides, G., and Gkoulalas-Divanis, A., ‘Distance-based km-anonymization of trajectory data,’ Proceedings - IEEE International Conference on Mobile Data Management, 2013, **Vol 2**, pp. 57–62, ISSN 15516245, doi:10.1109/MDM.2013.66.

[42] Rao, P., Lin, D., Bertino, E., Li, N., and Lobo, J., ‘Fine-grained integration of access control policies,’ Computers and Security, 2011, **Vol 30**, pp. 91 – 107, ISSN 0167-4048, doi:https://doi.org/10.1016/j.cose.2010.10.006, special Issue on Access Control Methods and Technologies.

[43] Sankararaman, S., Agarwal, P., Molhave, T., Pan, J., and Boedihardjo, A., ‘Model-Driven Matching and Segmentation of Trajectories,’ in ‘SIGSPATIAL,’ 2013 .

[44] Shang, S., Chen, L., Wei, Z., Jensen, C. S., Zheng, K., and Kalnis, P., ‘Trajectory similarity join in spatial networks,’ Proc. VLDB Endow., 2017, **Vol 10**, pp. 1178–1189, ISSN 2150-8097.

[45] Shang, S., Chen, L., Wei, Z., Jensen, C. S., Zheng, K., and Kalnis, P., ‘Parallel trajectory similarity joins in spatial networks,’ The VLDB Journal, 2018, **Vol 27**, pp. 395–420.

[46] Spyromitros-Xioufis, E., Papadopoulos, S., Popescu, A., and Kompatsiaris, Y., ‘Personalized privacy-aware image classification,’ in ‘Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval,’ ACM, 2016. pp. 71–78.

[47] Squicciarini, A., Karumanchi, S., Lin, D., and Desisto, N., ‘Identifying hidden social circles for advanced privacy configuration,’ Computers and Security, 2014, **Vol 41**, pp. 40–51.

[48] Squicciarini, A. C., Lin, D., Sundareswaran, S., and Wede, J., ‘Privacy policy inference of user-uploaded images on content sharing sites,’ IEEE Transactions on Knowledge and Data Engineering, 2015, **Vol 27**, pp. 193–206.

[49] Squicciarini, A. C., Sundareswaran, S., Lin, D., and Wede, J., ‘A3p: Adaptive policy prediction for shared images over popular content sharing sites,’ in ‘Proceedings of the 22Nd ACM Conference on Hypertext and Hypermedia,’ 2011. pp. 261–270.

[50] Such, J. M. and Criado, N., 'Resolving multi-party privacy conflicts in social media,' IEEE Transactions on Knowledge and Data Engineering, 2016, **Vol 28**, pp. 1851–1863.

[51] Summary, E., 'Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013-2018,' http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html, 2014, [Online Accessed 2018].

[52] Wang, W., Ying, L., and Zhang, J., 'On the Tradeoff between Privacy and Distortion in Differential Privacy,' in 'KDD,' 2014 pp. 517–525.

[53] Ward, K., Lin, D., and Madria, S., 'Melt: Mapreduce-based efficient large-scale trajectory anonymization,' in 'SSDBM,' 2017 doi:10.1145/3085504.3085581.

[54] Ward, K., Lin, D., and Madria, S., 'A Parallel Algorithm for Anonymization of Large-Scale Trajectory Data,' TPDS, 2019.

[55] Watts, D. J., *Six Degrees: The Science of a Connected Age*, W. W. Norton & Company; Reprint edition, 2004.

[56] Yarovoy, R., Bonchi, F., Lakshmanan, L. V. S., and Wang, W. H., 'Anonymizing moving objects: How to hide a MOB in a crowd?' Proceedings of the 12th International Conference on Extending Database Technology Advances in Database Technology (EDBT'09), 2009, pp. 72–83, doi:10.1145/1516360.1516370.

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

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

[59] Zhao, W., Ma, H., and He, Q., 'Parallel K-Means Clustering Based on MapReduce,' Cloud Computing. Springer Berlin Heidelberg, 2009, pp. 674–679.

[60] Zheng, Y., Zhang, L., Xie, X., and Ma, W.-Y., 'Mining interesting locations and travel sequences from gps trajectories,' in 'ACM Press,' 2009 pp. 791–800.

[61] Zixkhur, K., 'Location-Based Services,' http://www.pewinternet.org/2013/09/12/location-based-services, 2013, [Online Accessed 2016].

VITA

Katrina was born in Indianapolis, IN. In May 2014, she received her B.S. in Computer Science from Missouri University of Science and Technology, with a focus in Data Mining and Computer Security. She began her research during her undergraduate studies. Immediately following graduation, she began her Ph.D at the same institution, also in Computer Science with the same focus. During her time as a student, she has worked as a graduate teaching and research assistant within her department, a software developer at Brewer Science to automate chemical mixing equipment, and she formally accepted a position at Sandia National Laboratories immediately following her graduation. Katrina had a very strong focus in helping her department and participated in many events and activities to promote STEM fields at local schools. She was an active member of ACM and ACM-W and has worked closely with the University's diversity office to provide engaging activities for potential future students.

She published conference and journal papers as a primary and secondary author, most of which are cited in this research. In May 2019, she received her Ph.D. in Computer Science from Missouri University of Science and Technology.