ENHANCING PRIVACY AND SECURITY WITHIN SOCIAL NETWORKS

A Dissertation

presented to

the Faculty of the Graduate School

at the University of Missouri-Columbia

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

by

JOSHUA DENNIS MORRIS

Dr. Dan Lin, Dissertation Supervisor

December 2021

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

ENHANCING PRIVACY AND SECURITY
WITHIN SOCIAL NETWORKS

presented by Joshua Dennis Morris,
a candidate for the degree of Doctor of Philosophy and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Dan Lin

Dr. Giovanna Guidoboni

Dr. Jian Lin

Dr. Rohit Chadha

ACKNOWLEDGMENTS

I would like to thank Dr. Dan Lin who worked as both my advisor and mentor since 2019. She is the most encouraging and helpful person, and it has been a privilege to be able to work alongside her. After approving me for the Cybercorps: Scholarship for Service, she has helped me to develop and find venues for different types of research. Had it not been for her, I do not believe I would have ever gone into graduate school. I cannot express the amount of respect and admiration I have for Dr. Dan Lin and I hope she finds only success in her future endeavours.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

ABSTRACT

The introduction of online social networking has allowed people across the world to share information with each other. It has risen to be one of the most popular forms of internet usage where almost everyone has created one or multiple types of social network accounts. Along with its great benefits has come many concerns though, mainly that the overabundance of information has created a plethora of privacy issues for not only the users of online social networks, but also for the company hosting the specific social network. Due to the wide range of privacy concerns associated with online social networks, it is incredibly difficult to tackle all the current possible concerns.

In this thesis, we propose works to tackle two privacy issues associated with online social networks. These two privacy issues are: the friend search engine, and image content. Firstly we will introduce a new sub-graph approach to the friend search engine that removes the ability of attackers to gain more information of your friend list than your privacy settings allow. Secondly we will introduce a new privacy setting that allows users to define locations they do not wish their face to be seen in images. If an image is posted with their face in such a location, they will be privatized through facial replacement so that they are unrecognizable. The overall efficiency of these works will be tested so that their enhanced privacy does not cause usability issues if they are adopted by a social network site. These works allow users to remain more private while using social media and also help users to remain confident that their privacy is kept safe. These improvements not only help strengthen the privacy of users, but also help social network sites retain users that are more wary of privacy breaches online.

Chapter 1

INTRODUCTION

Social networks, such as Facebook and Twitter, are a way of connecting countless people through online sharing and media. These types of online social networks have become one of the most popular pastimes for all ages allowing people across the world to make connections and share content with one another. This explosion in popularity and sharing while great, has created an overabundance of information on people. Many people overshare and inadvertently expose more of their private life and information than is secure. Privacy issues related to both people's sharing and the lack of security within mainstream social media platforms have been on the rise.

Privacy issues can arise in a variety of ways. One way is through the sharing of content. Image sharing, especially, is gaining increasing popularity in social networks like Facebook, Instagram and Foursquare [1, 2, 3, 4]. Especially for social image sharing, privacy protection has now become a crucial issue to be addressed since images can intuitively tell *when* and *where* a special moment took place, *who* participated and *what* were their relationships, i.e., sharing images can reveal much of users' personal and social environments along with their private lives [1]. Mainstream news has reported multiple incidents about people being fired due to their private photos being disclosed to undesired audiences [5, 6]. It has become incredibly difficult to not only manage how to intelligently share image to intended audiences, but also remain private in images which you may not have control over.

Another topic of concern is directly related to social media sites. With the complexity and scale of these sites, it can be incredibly hard to design a system which intelligently deals with conflicting policies between different users and stays useful enough to allow large interconnected friend networks. One such issue is the friend search engine. This engine usually allows users to display the friend list of another user in order to foster new connections, and therefore allow more interactions between people that would otherwise not be connected. Although this is a useful feature to some, the friend search engine may help leak unwanted and unnecessary information of one's network, raising privacy concerns. By crawling an online social network through API tools, a few researchers have found that the friend search engine is more vulnerable to privacy breaches than many would expect. This lack of control lowers the confidence of user's and may in turn lead to people leaving the social media site since they are wary of privacy breaches.

Due to the wide array and complexity of privacy issues, it is very hard for social networks to tackle every issue quickly. Unfortunately, there is currently not enough privacy protection mechanisms to safeguard users and their data. In this thesis, we take a closer look at two types of privacy issues and propose corresponding defensive mechanisms. These two privacy issues are: the friend search engine, and image content.

1.1 FRIEND SEARCH ENGINE

Firstly, we'll discuss privacy protection in regards to the friend search engine. This engine usually allows users to display the friend list of another user in order to foster new connections, and therefore allow more interactions between people that would otherwise not be connected.

Firstly, we'll discuss privacy protection in regards to the friend search engine. As stated previously, this engine allows users to display the friends list of another user in order to foster new friendships.

The friend search engine, although useful, has some privacy concerns by its original design. Conflicting privacy policies between users can cause overexposure allowing an attacker to gain more information of someone's connections than their individual settings allow. Most of a user's friend list can be revealed using a series of carefully designed queries that crawl the user's friends of friends [7, 8]. Figure 1.1 shows an example of such a vulnerability. User's queried by the friend search engine will return the nodes they point to. In this case, we will focus on *Alice*. From this example, assume *Alice* only wants to show *Bob*, *Tom*, and *Mary*. Instead, *Gavin* and *Ben* want to show her as a connection. If every single node within this network is queried, *Alice* will have her entire friends list shown including *Gavin* and *Ben*. This could similarly be true if *Alice* wished that her friend list was private. This shows that with incompatible settings or random selections, an entire user's friend list can be inferred through the picks of others.

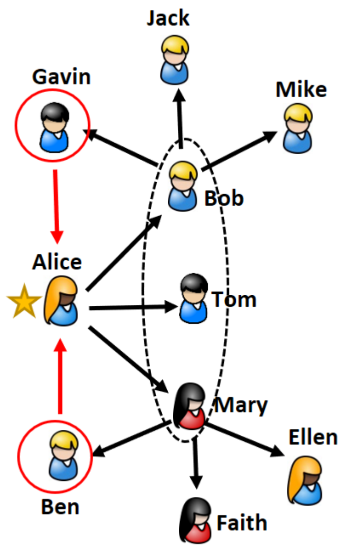


Figure 1.1: An Example of Attack Scenario

This vulnerability enables attackers to gain intelligence of a target's close friend network, and possibly use this information to gain leverage or obtain sensitive information regarding the targeted individuals. For example, a business man may not want to disclose all of its social connections to its competitors since that may contain business intelligence; a politician may not want to reveal his relationships with all his potential supporters. Such attacks clearly violate users' privacy expectations. Unfortunately, this is an under looked issue among social networking users. Someone who has created a private network and friend list should not have to worry about their connections being exposed through a work around in the system. When security controls are bypassed, users may feel hesitant to continue using a platform, which would be a loss for both the product owner and the individual user.

In the recent years, there have been a few approaches that aim to preserve users' private social connections against malicious friend searches [9, 10]. However, the latest research shows that they are still vulnerable under collusion-based attacks [11, 12].

In this work, we will propose a new friend search engine, called FriendGuard. FriendGuard maintains the same key function of the previous defense that allows people to view a maximum k friends of a user. Meanwhile, FriendGuard provides strong privacy guarantees. That is, if a user only allows k of his/her friends to be disclosed, our search engine will ensure that any attempts of discovering more friends of this user through querying the user's other friends will be a failure. For example, the previous attack scenario noted by Figure 1 will be avoided if *Gavin* and *Ben* were not reported as the top k friends of *Alice*.

The key idea underlying our search engine is the construction of a sub social network that is capable of satisfying query needs as well as controlling the degree of friend exposure. Specifically, we extract a minimum sub-network from the existing social network so that any individual or collaborative friend search will not disclose information outside this sub-network. Every node in this sub-network only has a desired amount of connections limited by each user's privacy policy. We have developed efficient algorithms to support friend queries in real time. These algorithms include a greedy algorithm that attempts to pick the top friends of a current user, a depth first search algorithm that prioritizes a single pick for each current user along a path, and an optimized method for the depth first search algorithm that picks root nodes for each path by smallest degree in the original online social network. We have also carried out an extensive experimental study and the results demonstrate both efficiency and effectiveness of our approach.

1.2 IMAGE CONTENT

Secondly, we'll discuss privacy concerns in relation to image content. Recognizing the importance of image privacy, researchers and social media sites have developed various privacy policies and tools to help users specify the group of people for photo sharing. However, most existing image privacy protection approaches [13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23] focus mainly on the privacy of photo owners and at most the photo owners' friends. They lack the consideration of other people who are in the background of the photos and are not related to the photo owners. In fact, when a person is in the background of someone else's photo, he/she may be unintentionally exposed to the public when the photo owner shares the photo online. For example, *Alice* had a bad day and visited a pub at night. Someone took a photo of the pub with *Alice* in the background. *Alice* had no idea about the photo until her supervisor came to show concerns to her because he coincidentally saw her drunk photo online posted by the other person. A recent interview [24] among college students also confirmed such privacy concerns, indicating that more and more undergraduates worry about becoming an Internet meme because their embarrassing moments were photographed by their peers and posted on social media. As an initial effort towards this new privacy problem, Llia et al. [23] suggested the use of face recognition to identify all the people in the photo but their implementation is still limited to identifying photo owner's friends through available image tags and they have not considered the associated location privacy issues as discussed in the following.

With an increasing frequency of images being associated with geo-tags and timestamps, image privacy now comes to the crossroads of location privacy. Such exposure may cause undesired consequences especially when the person being exposed was visiting sensitive locations. For example, a businessman, *Bob*, is meeting an important customer in a restaurant during a business trip while *Jack*, who usually reviews every restaurant he visits, took a photo of the restaurant with *Bob* and his customer in the background. *Jack* published his review along with the photo to a social media site. *Bob's* company's competitor noticed *Bob* was in the photo. The photo may have leaked business intelligence since it tells the competitor when and where *Bob* met the potential business partner whom the two companies are currently competing over. There are many other scenarios, such as visiting a specialty hospital or attending alcoholic counseling, which could cause similar uneasiness to the person if his/her photos at those sensitive locations were posted online by others. Furthermore, attackers may even be able to piece together a person's travel route by analyzing unprotected online photos. Specifically, the photos containing a target victim's face may be identified via face recognition; and the photos locations and timestamps may be revealed through various means such as geo-tags, metadata, or landmarks obtained from the advanced image processing tools. In Section 4.1, we demonstrate an example of such an attack to show its feasibility.

To better understand the aforementioned location related image privacy issues, we have conducted an exploratory user study among more than one hundred people to obtain their privacy opinions over a set of scenarios. The findings from the user study conform with our hypothesis that location sensitive photos could disclose too much of a person's privacy. Unfortunately, there have been very little works on how to help users mitigate such location-dependent image privacy. Thus, we propose a novel image privacy protection system, called LAMP (Location-Aware Multi-party Privacy), which aims to light up the location awareness for people during online image sharing. The LAMP system is based on a newly designed Location-Aware Multi-Party image (LAMPi) access control model that allows individual user to specify sensitive locations and timestamps for any photo in which their faces are identifiable. The proposed access control model goes beyond the traditional owner-centric privacy protection model, and the proposed LAMP system will facilitate social network providers to provide equal protection for any people in the same photo. Specifically, the LAMP system as an add-on to existing social media sites will automatically detect the user's occurrences on photos to be posted online regardless of if the user is the photo owner or not. Once a user is identified and the location of the photo is deemed sensitive according to the user's privacy policy, the user's face will be replaced with a virtually generated human face. As we know, face blurring has been commonly used for privacy protection during photo sharing, while face replacement has been provided by existing apps mainly as a fun pastime activity. We hereby argue that face replacement would be a better way to protect people's privacy as it offers several advantages which cannot be achieved by face blurring. First, it prevents attackers from using the latest image deblurring techniques [25, 26, 27] to uncover the people being protected. Second, the use of face replacement maintains the beauty of the photo and reduces the chance of the photo to become a target of an attack. Considering a photo with a blurred face and a photo with a swapped face, it is obvious that a blurred face has privacy concerns whereas a nicely swapped face may not be noticed by the attacker.

The key challenge during the design of the LAMP system is how to achieve the location-aware privacy protection without affecting users' image sharing experience. First, we need to be able to obtain users' location privacy concerns without adding too much burden to users. For this, we design a graphic-based policy specification tool for users to easily specify sensitive locations at different granularity levels following our proposed LAMPi access control model. Second, we need to ensure that the image uploading time is not significantly delayed due to the privacy policy checking. To protect the privacy for each person in the photo to be uploaded, the first step is to identify each person. However, given the huge number of social network users (e.g., 2.4 billion Facebook users), identifying a person who is not related to the photo owner by comparing the photo against all the social network users would be very time consuming and negatively impact the user's photo sharing experience. In order to overcome this problem, we propose a quick filtering approach that leverages face encoding and location policy indexing to drastically reduce the face comparison to a small group of candidates. We have implemented a prototype of the proposed system, and conducted a second user study. Our experimental results demonstrate the efficiency and effectiveness of our approach. In a summary, the contributions of our work are the following:

- We define a novel fine-grained location-aware multi-party image access control mechanism which breaks the existing limits of privacy protection only for photo owners and their friends by providing equal privacy protection to every identifiable individual in the photo instead of photo owners and their friends. Moreover, we consider the location-dependent privacy issues that are not studied in the past.
- We build a proof-of-concept application, LAMPi, to automate the location-aware multi-party privacy protection process. The algorithms designed for LAMPi are tested to be efficient and scalable to deal with the huge number of photos and users on social media sites.
- We conducted two rounds of user studies involving more than 200 people to obtain valuable user opinions on location-dependent privacy issues and evaluate the effectiveness of privacy protection offered by our approach.

1.3 SUMMARY

In summary, this thesis is composed of two main works involving privacy concerns related to online social networks. The first work goes over privacy issues related to online social network designs. This work, named FriendGuard, defends against overexposure related to friend search engines. The second work, LAMPi or Location-Aware Multi-Party Image Privacy, will focus on privacy concerns related to image content in which a person's face may be exposed unknowingly. This final work will create a system to identify and privatize people within photos based on the photo location regardless of who the owner of the photo is.

The rest of this thesis is organized as follows. Chapter 2 discusses the uniqueness and related works for the two main privacy concerns. Chapter 3 introduces Friend-Guard which aims to alleviate privacy issues connected with the friend search engine. Chapter 4 introduces LAMPi which allows users to utilize a new system to help them eliminate breaches of their privacy through posted images they are depicted within. Chapter 5 summarizes the findings and impact of each work. Chapter 6 concludes the thesis.

Chapter 2

LITERATURE REVIEW

In this chapter, the privacy concerns related to the friend search engine, and image content will be discussed in further detail. The related works associated with each privacy concern will be discussed along with the uniqueness of our proposed solution.

2.1 PRIVACY ISSUES IN FRIEND SEARCH

In this section, we tackle the minimally explored issue of securing privacy breaches associated with the friend search list. With the growing concerns of privacy issues incurred during the growth of online social networking, many researchers and service providers have attempted to propose solutions that help provide better privacy protection for users. Most of the efforts are placed on location privacy and information sharing privacy [28, 29, 30, 31, 32, 33, 34]. Very few studies have been carried out to preserve unexpected social network exposure caused by the online friend search engine.

Several vulnerabilities in the existing friend search engine of Facebook have been pointed out in [7]. Specifically, Bonneau et al. [7] found an attack that is able to discover nearly the entire friend list of a user by conducting an abundant amount of random friend queries using the friend search engine. In [8], Ren et al. point out that attacks that use intersection set computation may reveal more users' friend information if no privacy protection mechanism is in place. Such kind of attacks are especially concerning since users have no control of it and may not even be aware of these vulnerabilities, even though they have properly set up their privacy configurations, e.g., sharing only a small set of their social contacts.

To address the privacy concerns during the friend search, Ren et al. [10] proposes a remediation approach that uses a privacy-aware method to verify that the given set does not leak information. In this solution, user A queries the friend search engine for user B. If a friend of A and B, say C, is found, then the friends of C and B can be intersected to find the overlap. This intersection set of B and C will be recommended to A by the friend search engine, based on the findings that many users are willing to accept friend requests to strangers if they have a common connection [35]. By doing this, common connections are needed to gain an accurate estimate of a user's network. While this is useful, it restricts the amount of connections given through a friend search engine to a node that is not inside their common connections. Our approach aims to give controls to individual users, that can specify the exact amount of friends that they would like to disclose through a friend search engine. This would also help to give picks that are outside of a common connection, allowing more diverse sets to be given than this method allows.

Another solution proposed by Li [9] implements a policy that allows users to set a k-friend limit to the query results returned by a friend search engine. This solution utilizes a stack method that tracks the queries made, and changes results based on prior returned sets. This algorithm checks if current query results would violate the k-friend integrity of the previous searches when performing the current one. If the query violates this, the algorithm will not give the user(s) that would be violated, and instead gives another suitable option. By implementing this method, they are able to give optimal sets as available, but will then switch to more secure results if further queries would cause the k-friend policy limit to be violated.

An example of this defense is shown in Table 2.1 which references Figure 2.1.

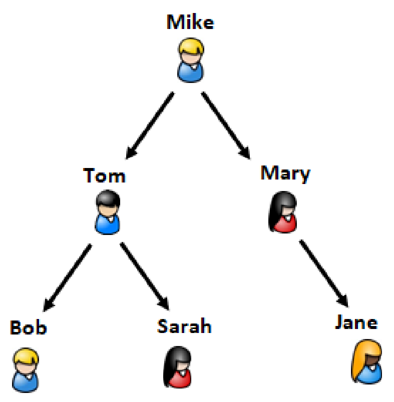


Figure 2.1: Sample OSN

Suppose that an attacker queries *Mike* and *Tom*, when there is no defense. The friend search engine gives the nodes specified by there arrows. By adding both results we can see that Tom has friends *Mike*, *Bob*, and *Sarah* violating K=2. When the defense scheme is in place though, the algorithm changes its behavior after giving the results from Mike. Because it knows that giving two new friends for *Tom* would violate his k friend policy, it instead returns *Mike* and *Bob*. This allows the friend search engine to return a set that does not violate *Tom's* k friend policy of 2.

User Being Queried (k=2)	*Mike*	*Tom*
Query results without defense	*Tom, Mary*	*Bob, Sarah*
Query results with defense	*Tom, Mary*	*Mike, Bob*

Table 2.1: Query Results Comparison

This defense scheme was later discovered by the same researchers to contain a vulnerability. In [11, 12], collusion attacks could be made on this defense scheme to gain more exposure than the k friend limit policy allowed. By pooling information across multiple attackers making queries, queries can bypass the defense method by making queries that gain information of a user's friend list not possible if the queries were made by the same attacker. Because this defense uses a stack like system to log their queries, the stack would not remain for another user, even if they were sharing information, because it was not in the scope of their defense.

User Being Queried (k=2)	*Mike*	*Tom*
Query results obtained by 1st attacker	*Tom, Mary*	*Mike, Bob*
Query results obtained by 2nd attacker	no query	*Bob, Sarah*

Table 2.2: Collusion Attack

The example in Table 2.2 shows how the defense stack is bypassed by a collusion attack. Specifically, when the first attacker queries $Mike's$ friends, the friend search engine returns Tom and $Mary$. Afterwards, if the first attacker queries about Tom, the friend search engine will return $Mike$ and *Bob* but not $Sarah$ in order to prevent the attacker from knowing more than two friends of Tom. However, this defense scheme is not able to prevent two or more attackers who combine their query results. For example, if the second attacker does not query $Mike$ but only query $Tom's$ friends list, it is possible that the friend search engine returns any two friends of Tom which could be *Bob* and $Sarah$. Once these two attackers share their query results, they will know that Tom has three friends: $Mike$, *Bob* and $Sarah$. This violates $Tom's$ privacy policy whereby he only wants to disclose two friends. While this example is simple, there is a lot that goes into making attacks on a network through collusion to explore most of a user's contact list. Such collusion attack violates users' privacy settings and result in the same overexposure as if there was no defense for this network.

Unlike previous works, which rely on runtime privacy checking and query modification, we tackle the problem from a new angle by extracting a sub-network from the online social network to form a robust platform for any friend search engine. We ensure that queries performed on our constructed sub-network will be robust to various kinds of attacks including the newly discovered collusion attacks, and provide a strong guarantee of preserving users' friend exposure degree.

2.2 PRIVACY ISSUES RELATED TO IMAGE CONTENT

In this section we explore privacy protection with existing works on photo content labeling, privatization, and potential overexposure. Our work shares similar goals to previous defenses by privatizing users in photos they do not wish to be a part of. However, our proposed approach shows increased efficiency when dealing with large social networks and utilizes facial replacement rather than blurring to hide people without dramatically altering the initial photo. More details are elaborated in the following.

Photos are an easy way to recognize people, a place, and an event. Due to this it can be very easy to tell what many people within a photo are located and doing, even if they are not the main focus of the image. Being recognized in an image can cause unwanted disclosure, and usually in this case the person who does not wish to be seen is not the photo owner. There is a need for the privacy of each person within an image to be considered regardless of if they are the photo owner or not since they can be identified within an image. In this section, we briefly review the image privacy protection methods for photo owners, their friends, and photo metadata.

(1) Image Privacy Protection for the Photo Owner

There have been a large body of image privacy protection works which focus on protecting the privacy of the photo owners [13, 14, 15, 16, 17]. These types of solutions include frameworks meant for suggesting privacy policies (i.e., which groups of people to share the image) for the photo owners at the time of the image being uploaded. For example, an early work [14] is by Squicciarini et al. who propose a privacy policy prediction system called A3P which considers image content, image metadata as well as the photo owners' historic privacy preferences when generating the policies. In [15], Hu et al. propose an interesting idea of calculating a level of sensitivity for each photo based on both user-defined rules and general rules discovered by machine learning.

Users can then use the sensitivity levels as guidance for their privacy settings. In [13], Yuan et al. employ machine learning algorithms to analyze a social media user's photo sharing behavior, taking into account both the content of the image and the social context of the users who may see the photo. From that information, the system then determines whether or not to share the photo, entirely or partially, with a certain user.

(2) Multi-Party Image Privacy Protection

The protection for a single photo owner has later been extended to co-owners of the photo, i.e., people who took a group photo together. This type of multi-party privacy protection is typically achieved by considering privacy preferences of each party, solve policy conflicts among multiple parties, and then blur the faces with privacy restriction [18, 19, 20, 21, 22, 23]. For example, Hu et al. [22] define an access control model to capture the multiparty authorization requirements, based on which they develop multiparty policy specification scheme and algorithms to solve policy conflicts among multiple parties. llia et al. [23] propose a new way for multi-party privacy protection. They employ face recognition to automatically detect faces on the photo, and present the photo with the restricted faces blurred out. Similarly, Fan et al. [20] also leverage the blurring technique to enforce privacy policies. They cluster semantically similar images and generate common privacy settings for privacy-sensitive classes. Then, the proposed iPrivacy system automatically identifies privacy sensitive objects on images and also help blur the detected sensitive objects.

Unfortunately, very limited efforts have been devoted into privacy protection of people who occur in the background of others' photos like what we discuss in our work. As acknowledged in [23], identifying a person who is not related to the photo owner, i.e., not in the photo owner's contact list, would require a huge amount of computing resources due to the need to scan the whole enormous social network user set. A related work by Henne et al. [36] needed up to 1 hour to check an image and notify the bystanders. It detects only 50%-70% privacy violations in many cases and did not enforce the protection.

(3) Privacy Issues Regarding Photo Metadata

Besides achieving protection through proper policy configurations, recent studies also look into potential privacy breach caused by metadata associated with photos [37]. Metadata like geotags and timestamps can easily disclose a person's location information, and multiple photos with geo-tags and timestamps may be used to track a person. To prevent undesired exposure, researchers [38] have proposed to remove metadata. However, such strategy may not be sufficient since the context of the photo may still reveal the location with the advance of the image processing technology. Our approach takes another route by hiding the person's face so as to avoid any location privacy breach. In another work [39], Chandra et al. developed a mobile app which can detect human subjects and issue a privacy alert if the location is sensitive. Their location privacy protection component is relatively preliminary which simply stores users' sensitive locations as link lists.

The Uniqueness of Our Work

To summarize, our work is different from existing works in terms of the following aspects. First, we aim to protect the privacy of every human subject depicted on a photo regardless he/she is the owner of the photo or happens to appear in the background of the photo. Second, we aim to preserve location privacy during the photo sharing. We not only propose a sophisticated location-aware image privacy policy language, but also design an efficient and scalable approach that minimizes computational overhead incurred by the privacy protection process. Third, we go one step further by providing privacy protection through face replacement instead of only preventing photos from being shared. Face replacement would also provide a better protection than face blurring since images with blurry faces may invite curiosity and become targets.

Chapter 3

PRIVACY PROTECTION FOR THE FRIEND SEARCH ENGINE

This chapter goes over a new method to enhance privacy and security for the friend search engine of online social networks. We introduce a novel friend search engine that will allow each person to list k friends when other view their friend's list. This new approach will guarantee that no more k friends of a user will be exposed even if the entire social network is queried. This method can be easily integrated with most current social networks. This method additionally has low space and runtime overhead so that it does not heavily affect any integration.

The rest of this chapter is organized as follows. Section 3.1 discusses the overall goal of this chapter in relation to the current security vulnerabilities. Section 3.2 talks about each specific algorithm and how they achieve a more secure friend search engine. Section 3.3 overviews our implemented algorithm across a variety of tests. Section 3.4 talks about some of the shortcomings of our solution and other possible problems. Finally, Section 3.5 concludes this chapter.

3.1 PROBLEM STATEMENT

In this work, the online social network is modeled as an undirected graph as defined in Definition 1.

Definition 1. ***(Online Social Network or OSN)*** *An online social network is* *defined as an undirected graph $G(\Xi, R)$, where Ξ is the set of the users in this social network, and R is the set of edges connecting pairs of users who have relationship with each other, i.e., $R = (u_i, u_j)$ where $u_i, u_j \in \Xi$.*

In the online social network, the users can set the number of contacts that they allow the friend search engine to disclose. This privacy setting is called *friend policy* in this chapter.

Definition 2. ***(Friend Policy)*** *A friend policy P(u_i, k_{u_i}) denotes that user u_i allows no more than k of his/her contacts to be disclosed by a friend search engine.*

Definition 3. ***(Friend Query)*** *Given P(u_i, k_{u_i}), a friend query on u_i is denoted as Q(u_i, k_{u_i}) which will return a set of 0-k_{u_i} users from u_i's contact list.*

Our proposed approach aims to honor user-defined friend policies by ensuring the friend exposure degree (Definition 5 under the attack model as defined in Definition 4).

Definition 4. ***(Attack Model)*** *Attackers are assumed to be able to query any user through the friend search engine, and may share their query results to gain more knowledge.*

Definition 5. ***(Friend Exposure Degree)*** *Let u_i denote a user on an online social network. User u_i's friend exposure degree (denoted as ξ_{u_i}) is the number of friends who will be revealed by aggregated query results through any number of queries on the friend search engine.*

3.2 THE FRIENDGUARD SEARCH ENGINE

As users and regulations define more strict privacy control, a fault in the friend search system that allows attackers to violate these privacy preferences is a big problem. Attackers who obtain a large percentage of a targeted users' friend information could develop an intelligent social-graph of that targeted user [7, 8] to gain leverage. This leads to growing privacy concerns from users, and possibly fault on the social network for not upholding their own privacy rules. To address these issues, we propose a new scheme to protect social connections during the friend search, which is called FriendGuard.

Our FriendGuard scheme takes a dramatically different approach compared to prior works [9, 10]. The FriendGuard will be operated by OSN (online social network) service providers which have the full knowledge of their own social networks. Our main idea is to help OSN providers to mark a sub-network for the friend search engine to constrain the friend exposure for all the users. The sub-network can either be built all at one time, or can be built incrementally as queries to the friend search engine are made. It is worth noting that the sub-network does not need extra storage space. Instead, the OSN providers will just need to add a flag to the user links to indicate whether this link belongs to the sub-network and can be used during the friend query.

The FriendGuard scheme can be operated in two modes: (i) unweighted friend selection; (ii) popularity-based friend selection. In the first mode, the FriendGuard will return any k friends of a user being queried without considering how frequently the user interacts with the k friends. In the second mode, the FriendGuard will strive to return the most popular friends while preserving the friend exposure degrees. It is worth noting that it is almost impossible to always return the most popular friends while guaranteeing the friend exposure degree for each user, which is likely to lead to overexposure as shown in the previous examples. In the following subsections, we will discuss the trade-off that needs to be made to ensure the privacy guarantee.

Figure 3.1 gives an example social network that will be used to compare the results from our proposed two types of friend search engine results. Specifically, Figure 3.1 shows the social connections among 7 users, whereby the edges between two users indicate the two users are friends of each other, and the value on each edge

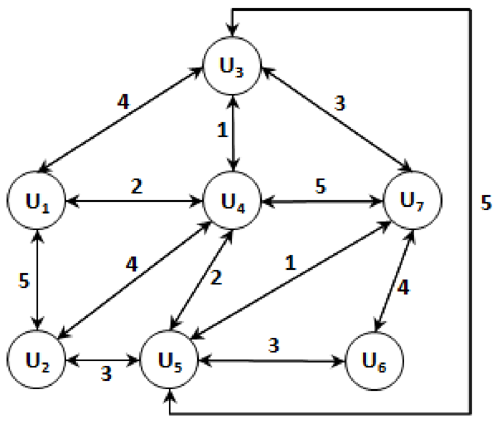


Figure 3.1: The Original Online Social Network

indicates an aggregated popularity score between the two users. Assume that the OSN providers have their own functions for obtaining the original popularity scores. We will model the connection between two users by using their average popularity score. For example, if user U_1 has a popularity score 6 among U_2's friends according to his/her contact frequency with user U_2, and user U_2 scores 4 among U_1's friends, their aggregated mutual popularity will be simply modeled as (6+4)/2=5 as denoted on the graph. This helps us to consider each connection extracted from the original social network to be bidirectional. That means that if user A is returned as a friend of user B during the query, user B will also be in user A's friend query list. In this way, we eliminate the possible collusion attack as defined by definition 4.

3.2.1 Unweighted Friend Selection

To support the unweighted friend selection, we will construct the sub-network as follows. First, we sort all the users (i.e., nodes in the OSN) in an increasing order to denote which users will be selected first. We have explored three kinds of selection types as presented in the following three algorithms: (1) Independent User Selection; (2) Path Traversal User Selection; (3) Degree Based Path Traversal User Selection.

The Independent User Selection algorithm (Algorithm 1) is the least complex al-

Algorithm 1: FriendGuard Independent User Selection

```
/* Ξ is a set of nodes within an OSN. This set can be an individual
   node or the total nodes within the network.  $N_i$ denotes the node
   selected.                                                         */
Data: ( Ξ )
/* net is the main OSN, whereas sub_net is the FSE specific
   sub-network.  Calling sub_net[$N_i$].friends.size() gives the current
   connections of $N_i$.                                             */
1  for ( $N_i \in \Xi$ ) do
2      k = $N_i$.k_friend_policy()
3      friends_left = k - sub_net[$N_i$].friends.size()
4      for ( $N_j \in net$ ) do
5          if ( $friends\_left < k$ ) then
6              if ( net[$N_i$][$N_j$] are friends && $N_j$ != $N_i$ &&
               sub_net[$N_j$].friends.size() $< k$ ) then
7                  if ( $N_j \in sub\_net[N_i].friends$ ) then
8                      continue
9                  else
10                     sub_net[$N_i$].friends.append[$N_j$]
11                     sub_net[$N_j$].friends.append[$N_i$]
12                     friends_left -= 1
13                 end
14             end
15         else
16             break
17         end
18     end
19 end
```

gorithm that could help efficiently build the sub-network, and also allow this network to focus on one user's top picks at a time. This algorithm is based off the assumption that picking each user's top friends will result in the most optimal sub-network for the FSE (friend search engine). This greedy strategy picks the maximum amount of connections for each current node that is selected. Each selection made is a bidirectional pick. This means that if user A chooses user B to be a connection, user B must also choose user A. At each pick, both the source and destination nodes are checked to see if they are at their respective k friend policy limit. If one of them is, the pick is not allowed. This is done until 0 - k picks are found for all users in Ξ.

The Path Traversal User Selection algorithm as shown in Algorithm 2 is a more

Algorithm 2: FriendGuard Path Traversal User Selection

```
/* $N_i$ is the node selected where c denotes the current node.  K is
   the friend policy specified by a user.                                */
Data: ( $N_c$ )
/* net is the main OSN, whereas sub_net is the FSE specific
   sub-network.  Calling sub_net[$N_i$].friends.size() gives the current
   connections of $N_i$.                                                 */
k = $N_c$.k_friend_policy()
friends_left = k - sub_net[$N_c$].friends.size()
for ( $N_i \in net$ ) do
    if ( $friends\_left < k$ ) then
        if ( net[$N_c$][$N_i$] are friends && $N_i$ != $N_c$ &&
          sub_net[$N_i$].friends.size() $< k$ ) then
            if ( $N_i \in$ sub_net[$N_c$].friends ) then
                continue
            else
                sub_net[$N_c$].friends.append[$N_i$]
                sub_net[$N_i$].friends.append[$N_c$]
                friends_left -= 1
                DFS( $N_i$, $N_i$.k_friend_policy() )
            end
        end
    else
        break
    end
end
```

complex defense scheme that uses a depth first search to create the FSE sub-network. This is done by picking a single top friend of the current user, and in iterations moving through the chosen nodes until no more picks can be made. This algorithm differentiates from Algorithm 1 because it attempts to optimize the first picks of every user in a path rather than just the current user. Checking the k friend limit at each selection and forcing bidirectional picks is still part of this algorithm to keep the integrity of our core defense.

Algorithm 3: FriendGuard Degree Based Path Traversal User Selection

```
/* Sorted is a set containing all nodes of the OSN. This list
   contains nodes in increasing order of degree from the original
   network.                                                      */
Data: (Sorted)
/* N_i denotes the node selected.  Calling sub_net[N_i].friends.size()
   gives the current connections of N_i.                         */
1 for ( $N_i \in Sorted$ ) do
2     if ( $sub\_net[N_i].friends.size() < k$ ) then
3         Algorithm 2( $N_i$, $N_i$.k_friend_policy() )
4     else
5         continue
6     end
7 end
```

The Degree Based Path Traversal User Selection algorithm (Algorithm 3) is an optimized depth first search algorithm. In this method, a sort method is conducted prior so that the nodes with lowest degree are picked first. This then utilizes Algorithm 2 to complete the sub-network. Utilizing this method helps to allow low degree nodes to make connections before all their connections within the online social network are selected. The resulting sub-network with this method would hopefully have a reduced number of nodes that return an empty set due to a lack of connections. While other algorithms are able to make connections at runtime by the FSE, the full sub-network would need to be made in this case so that low degree nodes can be made first.

Among the above three algorithms, Algorithm 1 and 2 have a nice feature that

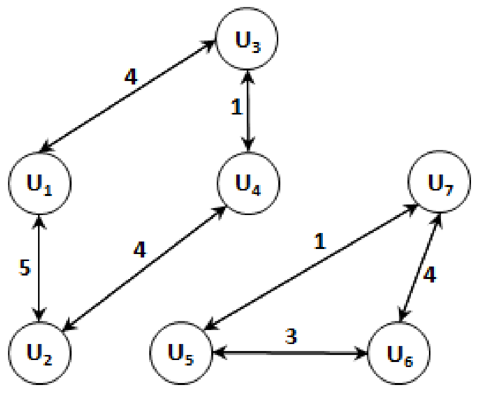


Figure 3.2: Unweighted FSE Sub-Network

the construction of the friend search sub-network can be carried out incrementally. In other words, the two algorithms only need to be called during the first query of each node to create the connections that will be displayed to the user at each later query.

Recall the example in Figure 3.1, the corresponding friend search sub-network obtained by our unweighted algorithm is shown in Figure 3.2.

3.2.2 Popularity-Based Friend Selection

The popularity-based friend selection aims to return a user's popular friends, i.e., active users on OSN, since active users may have more potential to help increase social activities when recommended to new users. It is nearly impossible to provide k top friends for each user while preserving the security of our system. Therefore, we employ the following trade off during the friend search. If a connection with the most popular friend in the extracted sub-network will leak more than k contacts of an individual user, we will substitute this connection with a less popular connection if possible.

Algorithms 1 and 2 can be modified to utilize weights rather than the node order. Adding weights to these algorithms only slightly changes their overall design. As shown in Algorithms 4 and 5, the weighted algorithm adds another step to check each

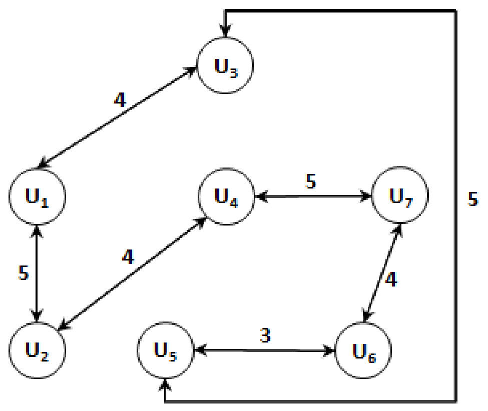


Figure 3.3: Weighted FSE Sub-Network

friend of the current node. This is done in two different ways: a prioritization of the current node's optimal friends, or a compromise between the current and destination node's optimal picks. The two options may lead to similar results in many cases, but there could be substantial differences in some cases. Since all the connections within the FSE sub-network must be bidirectional at the end to prevent additional information leakage, considering both friends' optimal picks may be a more fair choice when the selection process occurs. Figure 3.3 shows the resulting sub-network from Figure 3.1 when utilizing popularity-based (or weighted) selections.

3.2.3 Trade-off between Privacy and Usability

Figure 3.4 shows another example OSN. The FSE sub-network extracted by our unweighted friend search algorithm is shown in Figure 3.6. It is assumed that all users have picked $k = 2$ within this OSN for simple illustration. This example illustrates a trade off to be made between privacy and usability. The impact of this will also be evaluated and discussed in Section 3.3.1.

The FriendGuard Path Traversal User Selection algorithm was used here to make selections. Figure 3.5 shows the picks made by this algorithm. Each circled number indicates a root node that has been chosen. Then, the algorithm follows the sub-network to find other nodes connected to the root nodes until not any new node can

Algorithm 4: Directional Weight Selection

```
/* Ξ is a set of nodes within an OSN. This will be a single node for
   DFS versions of this, and one to the total amount of nodes for the
   Greedy algorithm.                                                  */
Data: (Ξ)
/* net is the main OSN, whereas sub_net is the FSE specific
   sub-network.  Calling sub_net[$N_i$].friends.size() gives the current
   connections of $N_i$.                                              */
for ( $N_i \in \Xi$ ) do
    k = $N_i$.k_friend_policy()
    friends_left = k - sub_net[$N_i$].friends.size()
    while ( $friends\_left < k$ ) do
        highest_weight = 0
        $N_h$ = -1
        for ( $N_j \in net$ ) do
            if ( net[$N_i$][$N_j$] are friends && $N_i$ != $N_j$ &&
              sub_net[$N_j$].friends.size() < k ) then
                if ( net[$N_i$][$N_j$].weight() ¿ highest_weight &&
                  $N_j \notin$ sub_net[$N_i$].friends ) then
                    highest_weight = net[$N_i$][$N_j$].weight() +
                      network[$N_j$][$N_i$].weight()
                    $N_h = N_j$
                end
            end
        end
        if ( $N_j == -1$ ) then
            break
        else
            sub_net[$N_i$].friends.append[$N_j$]
            sub_net[$N_j$].friends.append[$N_i$]
            friends_left -= 1
            if ( Algorithm_Type == DFS ) then
                DFS( $N_j$, $N_j$.k_friend_policy() )
            end
        end
    end
end
```

be added. The nodes with a $*$ next to them denote a connection already made by a previous bidirectional pick. A ! denotes that a node was unable to find any other new connected node. Arrows in the figure represent that the selection algorithm finds another path starting from the root node. This algorithm continues until all nodes

Algorithm 5: Bidirectional Weight Selection

```
Data: ( Ξ )
/* Algorithm 5 follows all the algorithm of Algorithm 4 besides the
   selection of the highest node within the friend connections.  This
   is done through the variation of this selection line within
   Algorithm 5.  All other lines are exactly the same before and
   after this line denoted by ...;                                  */
1 ...
2 if ( $net[N_i][N_j]$ are friends && $N_i$ != $N_j$ && $sub\_net[N_j].friends.size() < k$ ) then
3     $N_h = N_j$
4     if ( ( $net[N_i][N_j].weight()$ + $net[N_j][N_i].weight()$ ) ¿ highest_weight && $N_j \notin sub\_net[N_i].friends$ ) then
5         highest_weight = net[$N_i$][$N_j$].weight() + network[$N_j$][$N_i$].weight()
6         $N_h = N_j$
7     end
8 end
9 ...
```

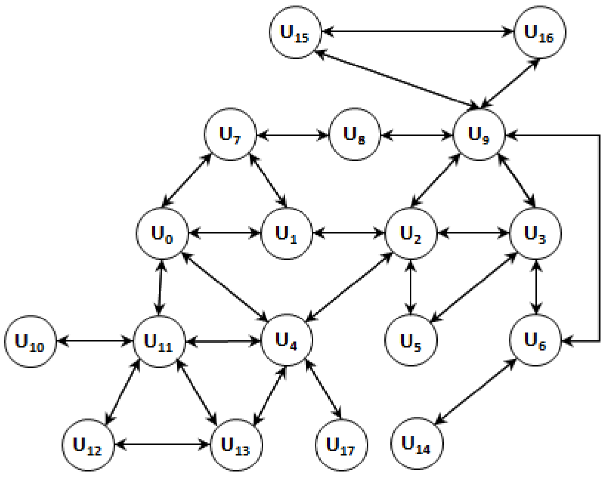


Figure 3.4: Sample OSN

have reached their k connections or have had a chance to make a selection.

Figure 3.6 is the resulting friend search sub-network from the original sample osn. Nodes highlighted in green have maximum k connections of 2, yellow represents having a (0, k) connections of 1, and orange represents a k connection of 0. In this case, users may have differing connection degrees in the FSE sub-network even with a constant k friend policy. In a network that has limited connections or an unoptimized

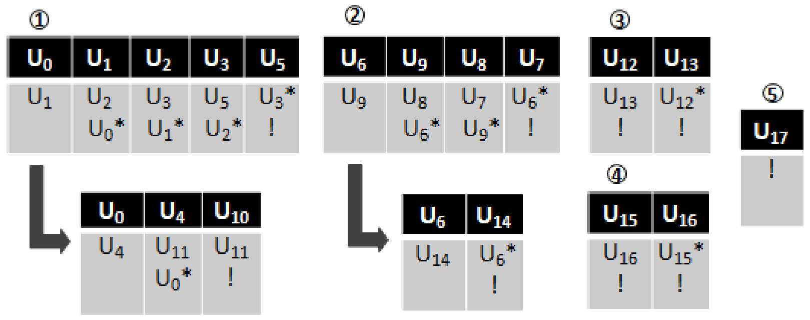


Figure 3.5: FriendGuard Path Traversal User Selection

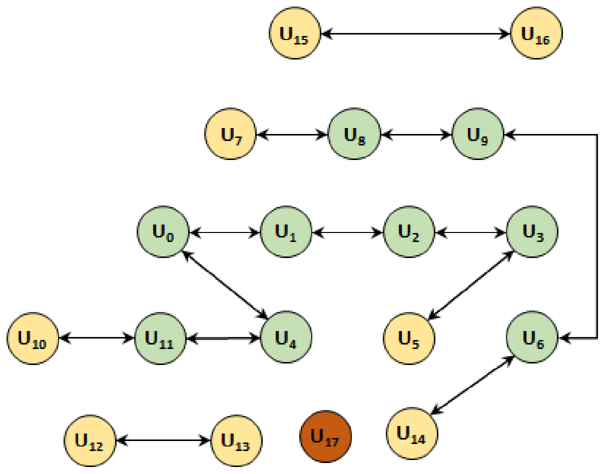


Figure 3.6: FSE Sub-Network

path, some nodes of the network may be unused or have less than k connections to make sure there is no more than the max allotted connections for any user. This trade off allows for a more secure sub-network for the FSE, but limits the degree of some users in the sub-network. While this does limit some FSE queries, these loses will be mitigated by the fact that an OSN is usually heavily connected and that some people will opt out of displaying any friends. This will in turn free up some space for less connected nodes.

3.3 EXPERIMENTAL STUDY

Our FriendGuard scheme was tested using three unweighted datasets from Facebook, Twitter, and Google+ [40], along with one Bitcoin weighted dataset [41, 42]. We tested efficiency of our algorithms on the unweighted data, whereas we used the BitCoin dataset to assess algorithm effectiveness. Effectiveness is defined as the most overall optimized sub-network, and measured by total weight of all connections in the sub-network for the FSE. Testing was done by creating fully populated sub-networks.

These datasets were extracted into two-dimensional vectors [40, 41, 42]. Table 3.1 shows the size of each network. To ensure fair experimental conditions, we kept the same k-friend policy for each user, while in an actual OSN users can pick different policies based on their own privacy preferences.

	Facebook	**Twitter**	**Google+**	**Bitcoin**
Nodes	4,039	81,306	107,614	5,881
Edges	88,234	1,768,149	13,673,453	35,592
Weighted	False	False	False	True

Table 3.1: Social Network Datasets

3.3.1 Sub-Network Node Degree

In the first set of experiments, we evaluate the effectiveness of our algorithms by varying k from 1 to 20. Then, we examine the percentage of nodes of different

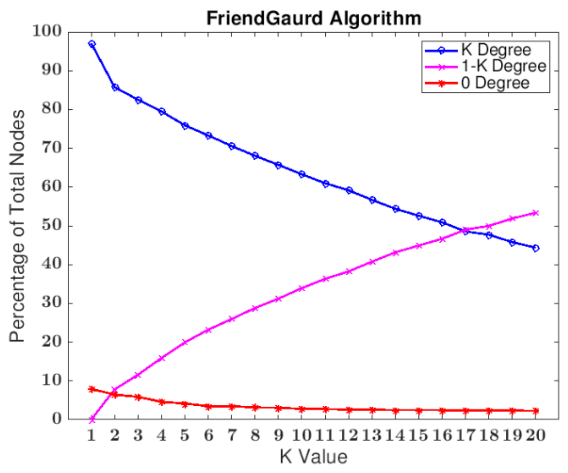
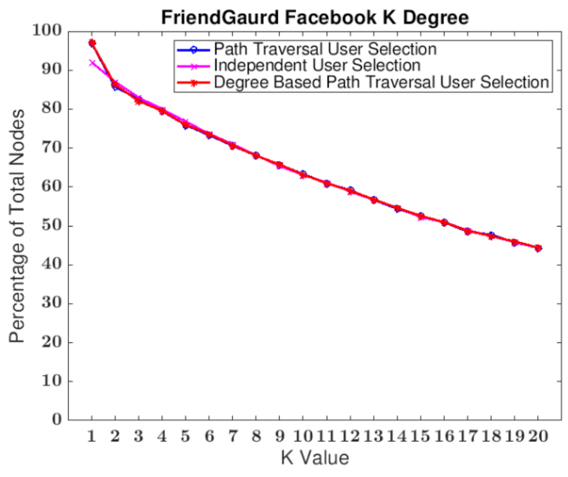
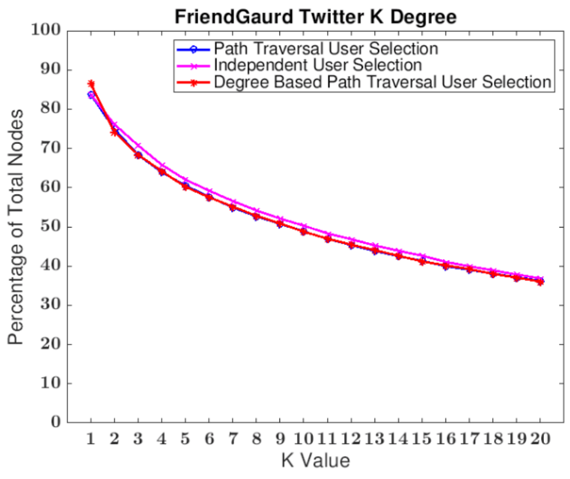


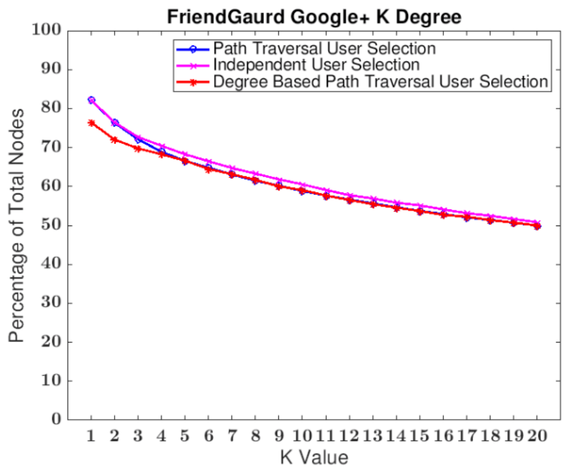
Figure 3.7: Perfomance of Path Traversal User Selection Algorithm in Facebook Dataset



(a) Facebook



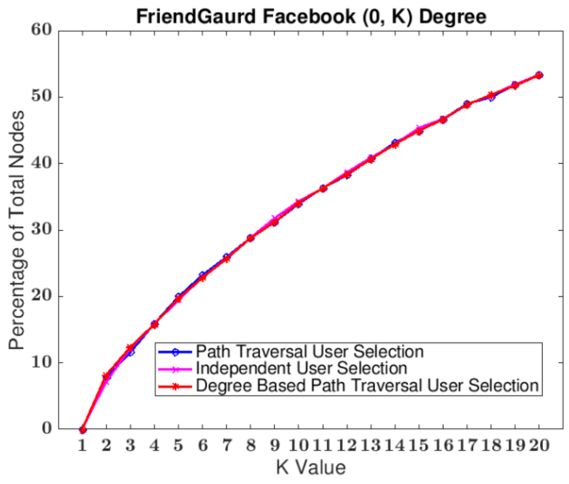
(b) Twitter



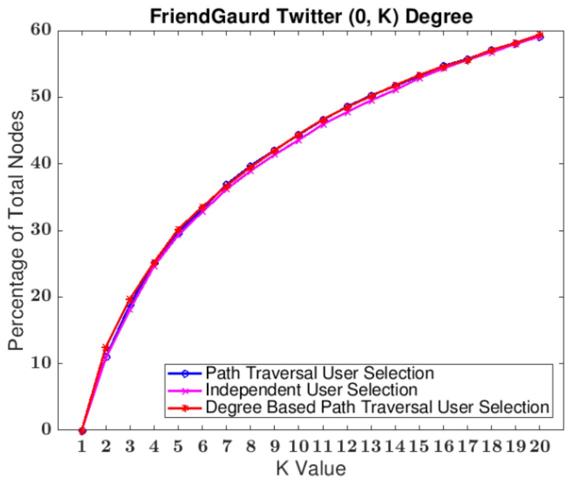
(c) Google+

Figure 3.8: Percentage of Nodes with Degree k

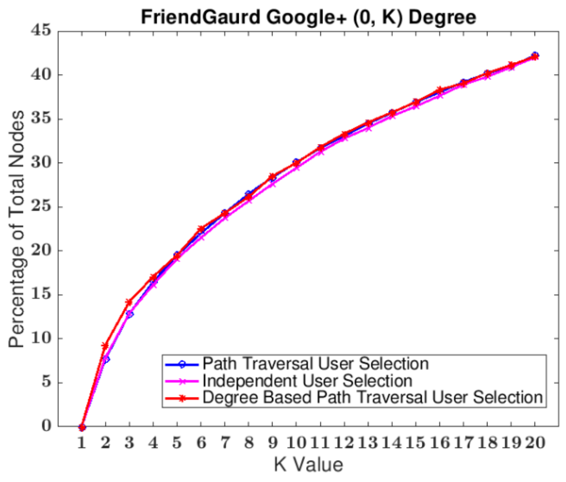
degrees in the obtained sub-network. Figure 3.7 shows the results from the Path Traversal User Selection algorithm. Observe that the percentage of nodes of Degree k decreases with the increase of k. This is expected since the larger the value of k, (i.e.each user needs to disclose more friends), the harder to find k mutual friends in the network that will not cause additional friend information disclosure. Although the extracted sub-network will not return k friends for some queries, we can see that the sub-network still has a high percentage of nodes with degrees ranging between 1 to k, and very low percentage (less than 5%) of nodes with degree 0 (i.e., no friend to report) in most cases, which means the majority of friend queries on the sub-network



(a) Facebook



(b) Twitter

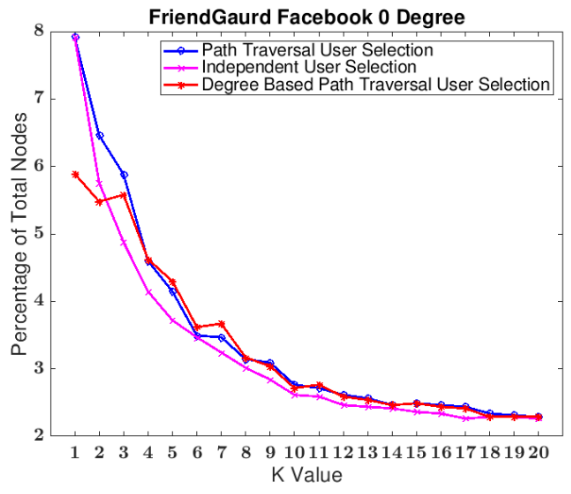


(c) Google+

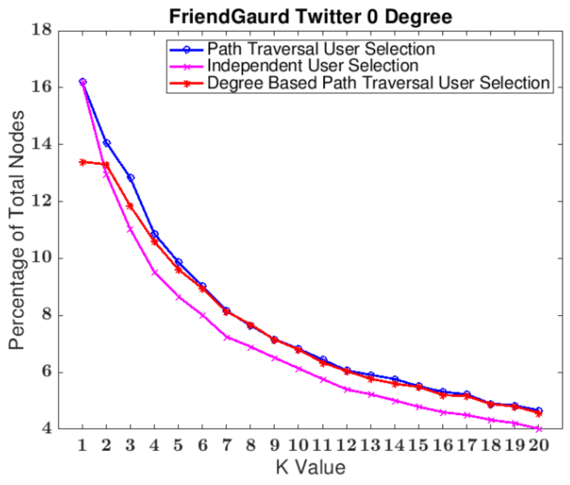
Figure 3.9: Percentage of Nodes with Degrees Between 0 and k

will return non-empty results.

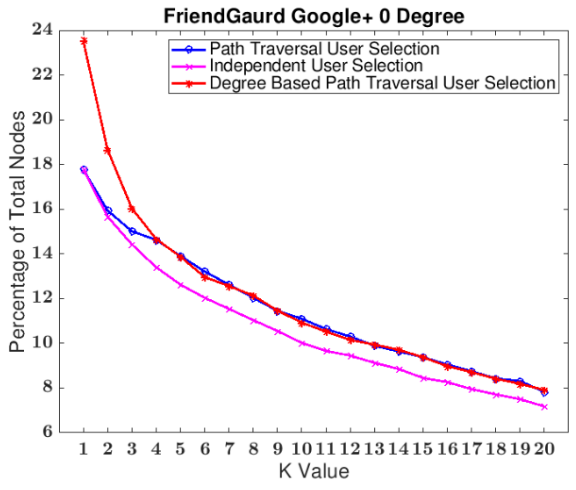
Our two other algorithms, the Degree Based Path Traversal User Selection and the Independent User Selection algorithm demonstrate similar effectiveness as shown in Figures 10, 11 and 12. Recall that the Path Traversal User Selection algorithm attempts to optimize first picks for each user. The Degree Based Path Traversal User Selection that employs the same strategy but picks low degree nodes first, and the Independent User Selection algorithm aims to optimize all of a single user's picks. In the experiments, the Degree Based Path Traversal User Selection algorithm achieved slightly higher k percentages noted in Figure 3.8, and slightly lower 0 percentage



(a) Facebook



(b) Twitter



(c) Google+

Figure 3.10: Percentage of Nodes with Degree 0

nodes in Figure 3.10 than the regular Path Traversal User Selection algorithm for all datasets besides Google+. This is expected because the Degree Based Path Traversal User Selection prioritizes nodes with lower degree. This helps these nodes to get picks before their connections have k connections and are unable to be picked. The Independent User Selection algorithm preformed slightly better than the Path Traversal User Selection algorithm in all cases. It preformed slightly better or around the same for the Degree Based Path Traversal User Selection algorithm as k grew, and a littler worse when k was low. While these algorithms have different methods and overall performance. They all have very similar node degree results as k grows.

As k increases, the FSE sub-network moves from having mostly k degree nodes to mostly (0, k) degree nodes. This could be due to the average degree of nodes being less than the corresponding k values. While this would not be an optimal return set, it still gives most of the desired functionality of the FSE for users. With no nodes having higher than k degree, our approach gives most users full or close to full k sets from the FSE without any vulnerabilities.

Since our system, in theory, may cause some nodes to have no friends to report, we hereby evaluate its effect in the real datasets. With small values of k, the percentage of unusable nodes having 0 connections is between 5 - 24 % in Figure 3.10. This is not the majority of nodes, but still heavily reduces the functionality of the FSE. Because our testing method keeps k constant for all users, having low k values heavily affects nodes that have a low degree in the original OSN. This drawback is substantially reduced as k increases. Across all datasets, the percent of nodes with 0 connections is between 2 - 9 % for $k = 20$. This could be perceived as the expected number of users who have a k friend policy of 0. While these are average results across three different datasets, the percentage of nodes in the FSE sub-networks will heavily depend on the average degree of nodes, and the adopted k friend policies for each user. It is likely that most results will be similar or better as OSNs are usually highly connective. Our average degree per node in the Facebook dataset is around 22 friends whereas the real average is 338 friends [43]. This should help to increase the functionality of our defense scheme when utilized in a very large OSN.

3.3.2 Node Connections Optimized by Weight

The Bitcoin dataset was used to measure the aggregated weight of all connections in the FSE sub-network. This test was preformed using the two methods shown by Algorithms 4 and 5, respectively. Tests were done for both the Path Traversal User

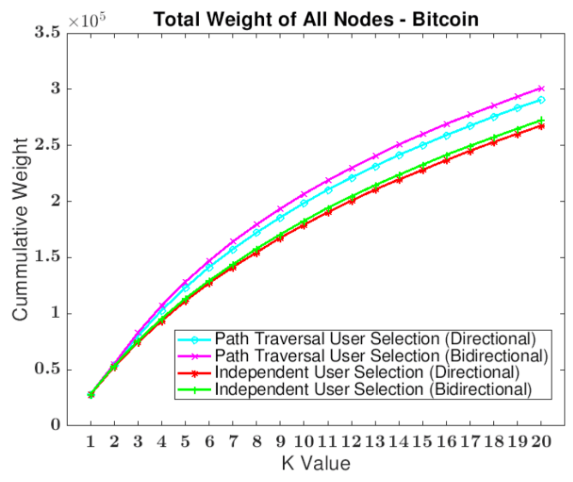


Figure 3.11: Effect of Weighted Nodes

Selection and Independent User Selection versions of the algorithms, varying 1-20 k policies. As in the prior experiment, the policies were kept constant for every node.

Shown in Figure 3.11 is the aggregated weight for both the tested algorithms with two different choices. Directional Weight Selection was done by choosing weights in deceasing order of connections from source to destination node. Bidirectional Weight Selection was also chosen in decreasing order, but selected by sum of weights from source to destination and destination to source.

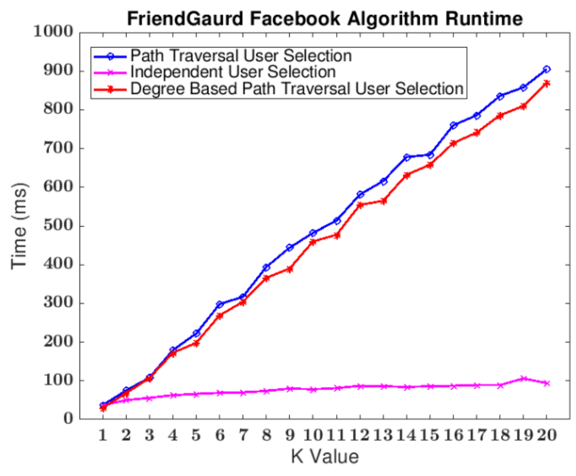
We find these results not surprising. Due to prior testing done in Section 3.3.1, it was expected that weights for both Path Traversal User Selection and Independent User Selection algorithms would be similar due to their near identical node degrees across the datasets. The Independent User Selection algorithm did have a slightly lower total weight across all nodes. This could be caused by the fact that it selects highest weight nodes for one user at a time, whereas the Path Traversal User Selection algorithm selects a single highest degree node at each iteration in the path from the starting node. This could allow for more nodes to get their highest pick which would result in a higher net weight if each method has a similar percentage of degree among the nodes. Bidirectional weight selection was consistently higher than than directional weight selection. This is again caused by the fact that this method optimizes picks based on both the source and destination weights. Because node selections must be bidirectional, a directional weight scheme can lead to a situation where one node has a high weight and its selection a low one. This will usually result in a lower total weight than a compromised selection.

Both weight selection techniques have very similar runtimes compared to one another. In this experiment, bidirectional weight selection makes consistently higher weighted picks in this test network. While the Independent User Selection algorithm performs slightly worse by comparison with the Path Traversal User Selection method, either algorithm can be run with similar results. This test emphasizes that our method can be run under various settings, produce similar results, and still be able to maintain a strong defense against information overexposure.

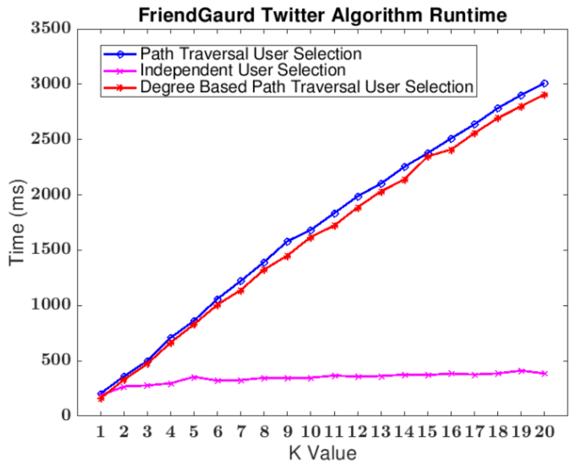
3.3.3 Full Sub-Network Runtime

Lastly, all unweighted datasets were tested to measure time performance. In this test, Twitter and Google+ were capped at 8000 node networks. Results are reported in Figure 3.12. The Independent User Selection algorithm had a much lower runtime than the both the Path Traversal User Selection, and Degree Based Path Traversal User Selection algorithms. While both Path Traversal User Selection algorithms perform similarly, it must be noted that our reported results do not account for the creation of a sorted list needed for the Degree Based Path Traversal User Selection. The sorted list took some time to create putting this algorithm behind the speed of the original Path Traversal User Selection when accounting for the creation of a sorted list. While all algorithms show a fairly linear runtime, the sub-network does not need to be created before any nodes are queried by the FSE.

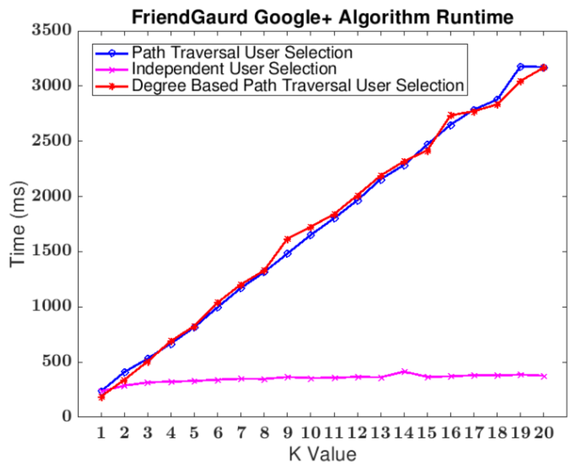
Running each algorithm as a node is queried seems to be the optimal way of making the FSE sub-network. While the Degree Based Path Traversal User Selection algorithm gives slightly better performance in node degree and optimized picks noted



(a) Facebook



(b) Twitter



(c) Google+

Figure 3.12: Runtime

in Section 3.3.1, this algorithm cannot be made at runtime. Our results show that both the Path Traversal User Selection and Independent User Selection algorithm at $k = 8$ (the standard for the original Facebook FSE when vulnerabilities were found [7]) average below 1ms for each node. This would allow our sub-network to be set up in iterations which additionally helps to make connections with more recent data. Due to the low average runtime of each node, this type of setup would have little to no impact on a user making FSE queries.

3.4 DISCUSSION

As security grows, most companies still view optimization of processes as one of their biggest concerns. If a solution can not be optimized in a way where normal actions do not impede the user, it is unlikely the solution will be adopted by the companies. Therefore, our proposed friend search engine strives to achieve high efficiency while adding extra privacy protection. Employing either the Path Traversal User Selection or Independent User Selection algorithm works very quickly when only run on one node. This allows a sub-network to be setup incrementally, enabling quick friend search engine queries, while offering a secure environment in the back end.

It is worth noting that our defense scheme requires updates to the constructed sub-network when someone changes his/her friend policy to a lower k value. This is due to the fact that sets from the friend search engine are given from sub-network connections rather than being created in real time. In this case, parts of the sub-network must be removed so that the security integrity of our approach is kept. This would require a complexity of $O(2kc)$ in the worst case because all connections to the changed node must be removed. This is equally true when the popularity scores of users are changed during the use of the OSN. While this is a low impact on the performance of our solution, it must be kept in mind that the sub-network will be continuously changing in response to user policy changes and social activities.

The change to the sub-network leads to another possible attack that has not been discussed in the existing friend search engine defense schemes [9, 10]. That is attackers may leverage the historical query results to gain more knowledge. For example, consider user A with a friend policy k of 3. Assume that at the beginning user A's top friends are users B, C, and D, but after one year their top friends change to users E, F, and G. If someone had tracked this information over time, it is likely to say that they now know user A is still connected to users B-D, and is also connected to users E-G. This violates user A's friend policy which only allows the disclosure of 3 of his/her friends. This is true for our defense scheme and possibly the defense schemes in [9] and [10] if common and popular connections change over time. While our defense can mitigate this by never changing the original setup of the friend search engine sub-network, this heavily mitigates the overall relevance of the friend search engine. This is just another consideration to be made when setting up defense schemes, and if security should be consistent for its original design or should be consistent across its lifetime. Due to the nature of the friend search engine though, it is hard to say if any defense scheme can be resistant to this attack without heavily reducing the relevance of the friend search engine, which is the trade off that our defense scheme can make.

3.5 RECAP

In this chapter, we have proposed a novel friend search engine called FriendGuard which honors users privacy policies and guarantees friend exposure degree. The FriendGuard system is based on the idea of constructing a reliable sub-network that eliminates vulnerabilities brought by aggregating query results from different rounds to a friend search engine. This sub-network can be created prior to and updated incrementally in response to the changes of social network connections as well as users' social activities. The FriendGuard supports two kinds of friend searches including unweighted and popularity-based friend search, and we have developed several algorithms. In the experiments, we have evaluated and compared our algorithms in different real network datasets. The experimental results demonstrate both efficiency and effectiveness of our approach. It is believed that by using our system, the friend search engine will be a more secure environment at the time of implementation while additionally giving users more flexibility in their privacy controls.

Chapter 4

PRIVACY PROTECTION FOR IMAGE CONTENT

This chapter discusses protecting user privacy in relation to images content where their face may be exposed in a photo without their knowledge. We introduce a novel system that can detect people within photos and obfuscate their appearance based on the posted photos location and time. This allows users to specify privacy levels where regardless of who posted an image, they can manage how their face is seen in an image. This method uses facial swapping so that the obfuscated people within the photo do not dramatically affect the final image's appearance. Integrating our system into existing social networks adds a minor amount of space for a new user privacy setting, but has minimal impact on posting runtime as our solution is efficient when used in big and small social networks.

The rest of the chapter is organized as follows. Section 4.1 presents the privacy risk analysis. Section 4.2 introduces our proposed LAMPi access control model and its implementation. Section 4.3 describes the LAMP system. Section 4.4 reviews privacy evaluations. Section 4.5 reports experimental studies. Finally, Section 4.6 concludes the paper.

4.1 IMAGE SHARING RISK ANALYSIS

4.1.1 Threat Model

As the saying goes, a picture is worth a thousand words. An online photo/image can give out rich information about *who* are doing *what* at *when* and *where*. To better analyze the privacy risks incurred by image sharing, we classify image privacy based on two criteria: human-oriented, and context-oriented.

Human-oriented image privacy can be further classified into three types:

(1) Photo owner's privacy: This type of privacy is currently preserved by allowing the photo owner to specify the groups of people who are permitted to access the shared photo. A majority of previous research work [14, 13, 15, 22] and commercial social media sites provide policy recommendation and configuration tools to achieve this. For example, Facebook users can choose to share the photos only with their friends but not friends of friends.

(2) Photo owner's friends' privacy: This refers to the privacy of the photo owner's friends who took the photo together with the photo owner. For example, Alice plans to post a party photo that includes her friend Kate. Kate is a shy woman who rarely shares photos online. Considering Kate's privacy, Alice may need to communicate with her before publishing the photo. However, such multi-party privacy issues are mainly discussed in academic world [3, 22, 21]. The current social media sites offer very little functionalities that support the multi-party privacy protection.

(3) Unaware people's privacy: This refers to the privacy of the people who are in the photo but are not aware of their photo being taken by others. For example, when someone took a selfie on the street, other pedestrians may be captured in the photo. These pedestrians will not know when and where their photos would appear on the Internet. Recently, an interview-based study [24] among college students found that undergraduates felt a heightened state of being surveilled by their peers when their photos were taken without their permissions and shared on social media by others. Participants in that study stated that they worried about being judged by others in a negative way based on the images which they were not aware of being taken.

Context-oriented image privacy can be further divided into two categories:

(1) Activity-dependant privacy: There are various scenarios when a person does not feel comfortable of sharing that moment with everyone. For example, a person in a funny costume may just want to share the photo with his/her close friends. In another case, a woman was drunk and someone else took her photo [24]. If the photo was posted online, it could lead to misjudgement of the woman and damage her general reputation. News also reported that some people were fired due to online photos. One case is that a fireman took a sick day off for attending an event, and he was later fired because his supervisor saw the event photo and identified him [44].

(2) Location-dependant privacy: A photo can leak location information of a person in many ways. The photo's embedded EXIF (Exchangeable image file format) [45] is a direct source that tells the date and GPS coordinates a photo was taken. Although some social media sites like Facebook and Instagram stripped the metadata when publishing the photos, they store the metadata in a separate database. If a hacker gains the access to these databases, it is even easier for them to track users since they now just need to look at the collection of metadata from all photos without spending much time on extracting metadata or analyzing photos one by one. Besides metadata, the photo itself may tell where the location is. Advanced image processing algorithms can identify the landmarks and the street signs. Yet another way could be the crowd sourcing. People living in the neighborhood of the place where the photo was taken may easily spot familiar buildings on the photo. With this said, *posting photos without metadata is still not sufficient to guarantee the location-dependant* *privacy of people in the photo.*

Most existing works on image privacy mainly focus on protecting photo owner and their friends' privacy (detailed review can be found in Section 2.2). Very limited efforts have been devoted into the other equally important privacy issues. i.e., unaware people's privacy, context-oriented image privacy. Thus, in this work, we aim to design a new access control mechanism to protect people's privacy in the photos which were taken without their knowledge and permissions. Our approach is complementary to existing methods and aims to achieve a full spectrum of image privacy protection. To better motivate our work, we will first present a location tracking attack and an exploratory user study in the following.

4.1.2 Location Tracking Attack Using Photos

In this experiment, we attempt to track a target person through his/her online photos. The goal is to show that it is not a difficult task for an attacker to cyberstalk a person. To avoid legal issues with a randomly chosen normal person, we decided to select a prominent figure whose images are publicly available in different venues: Joe Biden. Also, we do not hack into any social site to obtain metadata, whereas attackers can

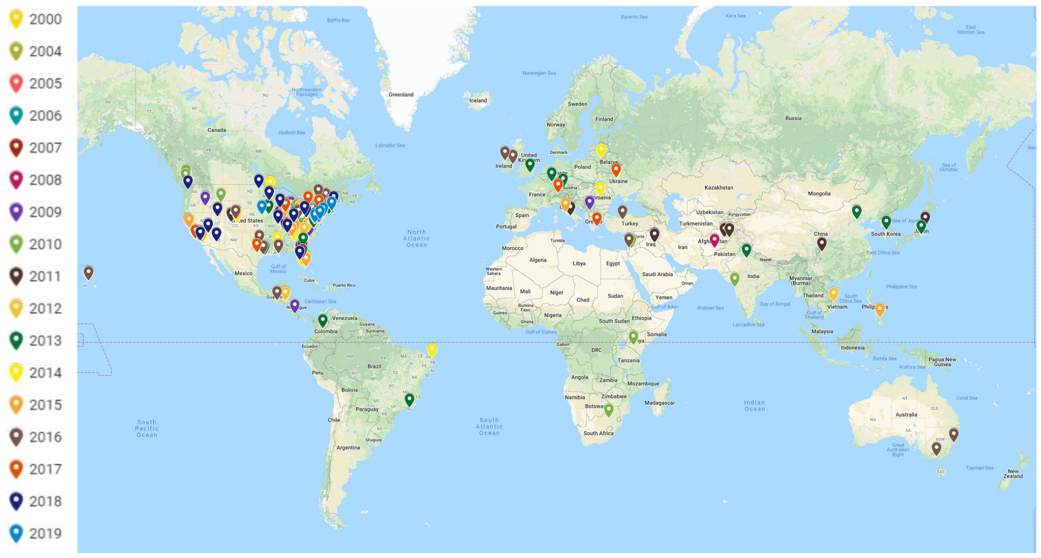


Figure 4.1: User Tracking Through Photo Metadata

certainly gain more information than us by doing so.

We wrote a script to automatically crawl Google images to obtain the target person's photos between 2000 and 2019. We collected around 30,000 photos for the target. From the collected photos, we further analyze the metadata. Although not all the photos contain the metadata, it is still amazing that we were able to found 721 days of location and visiting time for the target. Based on the obtained information, we created a tracking map as shown in Figure 4.1, where each point on the figure shows the location of the target person and the color of the point indicates when the photo was taken.

It is worth noting that Google images may not return photos of a person if he/she is in the background. Even so, the photos obtained from Google images already reveal a lot of location information of a person. When an attacker utilizes advanced image processing tools to look for any occurrences of a target (either foreground or background) and combines the knowledge of the metadata stolen from the social media providers, the target movement may be exposed in a much deeper level due to the prevalence of photographing nowadays. Considering that people who took photos of themselves know about the sensitivity of their current locations, whereas people who were in the background of others' photos have no idea their locations have been recorded, one idea in our proposed work is to replace the faces of people who are in the background of the photos to avoid undesired exposure. In this way, even if the attackers run the image processing tool and have all the metadata of photos, the targets' faces have already changed and would not be identifiable.

4.1.3 A User Study on Unexpected Privacy Disclosure

In order to better understand users' concerns on location-dependent image privacy and gauge their interests in our proposed privacy protection mechanism, we conducted an online user study on Mechanical Turk. The user study is fully anonymous and follows the IRB exempted project guidelines.

We have recruited total 111 participants, including 51 females and 55 males. 15% of them are between 18-25 years old, 41% are between 26-35 years, 23% between 36-45 years, 13% between 46-55, and 8% above 56. The age distribution conforms with the age groups of people who access the social media more often.

At the beginning of the user study, we asked participants if they were aware that online images may tell others where they were and what they were doing, and how much they valued their privacy especially location privacy. From the response, we find that more than 74% of participants were aware of the privacy issues incurred by online images, and more than 76% emphasized that location privacy is important.

From there, we presented 10 different scenarios to the participants and asked them if they would be concerned when their images and their locations are disclosed to unexpected parties. Specifically, each scenario is accompanied with a short paragraph of story and an image. The 10 scenarios were designed with the goal to cover various aspects of our daily life in a nutshell. From the participants' responses, we found that when a photo discloses a critical moment or location of a person without being noticed by the person, more people would hope their identities are protected during the sharing of such kinds of photos. For example, when someone (say Bob) is planning to switch a job and had a job interview at a restaurant, another customer who also dined at the restaurant took a photo of the restaurant with Bob and his interviewers in the background. The customer later shared his photo along with the restaurant review comments in a famous restaurant review website. If Bob's supervisor or colleagues saw the photo and recognized Bob and competitor company's people, that could raise unnecessary tension in Bob's current workplace. Therefore, we see that 92.5% of participants would like their identities be protected in this scenario, which is the scenario with the most concerns. The second mostly concerned privacy breach scenario is when someone's children and home location may be exposed to

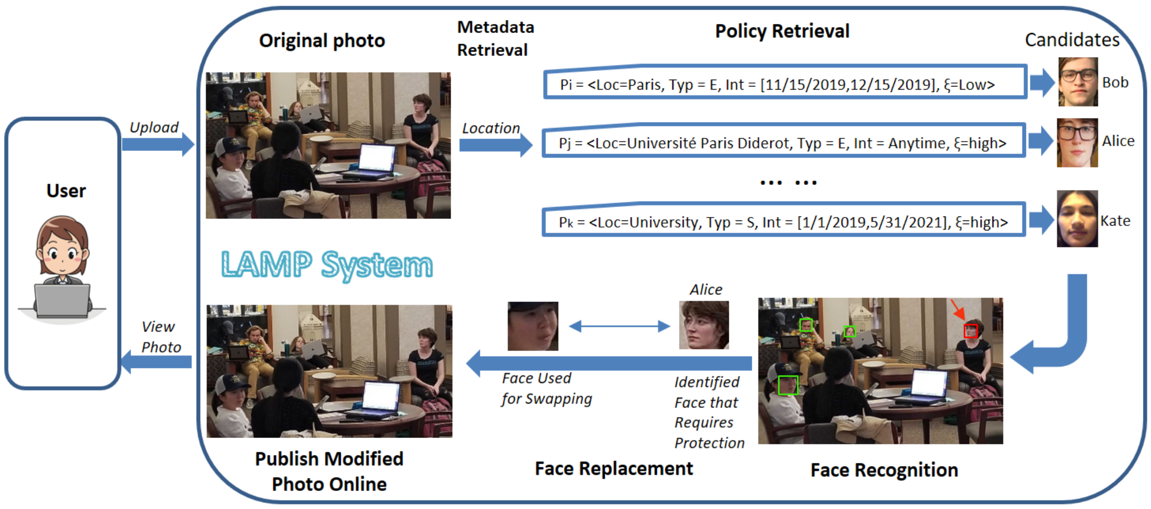

Original photo
Metadata Retrieval
Policy Retrieval
Candidates
User
Upload
Location
Pi = <Loc=Paris, Typ = E, Int = [11/15/2019,12/15/2019], ξ=Low>
Pj = <Loc=Université Paris Diderot, Typ = E, Int = Anytime, ξ=high>
Pk = <Loc=University, Typ = S, Int = [1/1/2019,5/31/2021], ξ=high>
Bob
Alice
Kate
LAMP System
View Photo
Alice
Face Used for Swapping
Identified Face that Requires Protection
Publish Modified Photo Online
Face Replacement
Face Recognition

Figure 4.2: The Privacy Protection Procedure in the LAMP System

strangers. About 91.5% of participants desire an identity protection in this case. On the other hand, some scenarios that may not lead to severe consequences, such as having a personal trip and drinking at a bar, have received privacy concerns from a little fewer people, but still close to 70%. Overall, we can observe that majority of people are concerned when their photos being taken and published by others without their knowledge, especially those photos that disclose sensitive locations and reveal their private issues.

At the end of the survey, we also asked the following general question: "*Suppose you are depicted in a photo published on social media by a stranger while you are in a location where you wish not to be seen. If social media websites provided functionality for hiding your identity (e.g., face swapping) when you are in such photos, would you like to use this function?*" More than 93.4% of the participants said that they would like to use such kind of services, which indicates a promisingly high acceptance rate of our proposed system.

4.2 LAMPI ACCESS CONTROL MECHANISM

In the previous section, we have discussed both human-oriented and context-oriented image privacy, among which context-oriented image privacy of unaware people is least protected in the literature. To fill the gap, we define the LAMPi (Location-Aware Multi-Party image) access control mechanism, complementing the traditional image access control. The LAMPi policies will allow users to specify location sensitivity at different scenarios and at different granularity levels. Its formal description is given below.

Definition 4.2.1. A LAMPi policy P of a user u consists of the following components:
• Location range (*Loc*): The range of locations protected by policy P.
• Location type (Typ): This indicates whether the location is given as a semantic location (denoted as 'S') or an exact address (denoted as 'E').
• Time and date interval (Int): The time and date intervals during which the locations within *Loc* should be protected.
• Sensitiveness (ξ): The sensitiveness of the location *Loc*, which has two levels: "High" or "Low".

Since the LAMPi policies are mainly used to express the users' privacy concerns when they are unintentionally captured in others' photos, there is no need for the LAMPi policy to specify the sharing group like traditional image privacy policies. Instead, the user can specify how much they care about themselves being exposed in the particular location using the sensitiveness level. When the sensitiveness level is set to high, the user's face at that location will be replaced for protection even if the user's face on the photo is less identifiable, i.e., the face matching score is lower than a threshold (say 50%). When the sensitiveness level is low, our system will only replace the user's face when the face matching score is above the threshold. In this way, we minimize unnecessary image modifications. For locations which are not specified in the user's policies, the locations are simply considered not sensitive for that user.

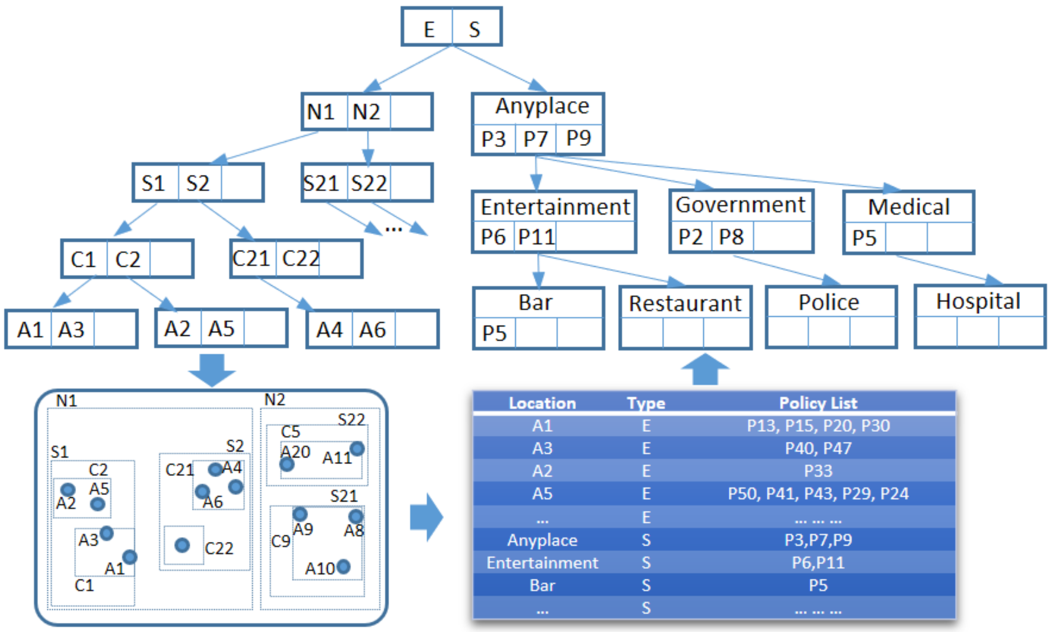
The following are some example LAMPi policies which demonstrate the usage of different policy settings.

Example 4.2.1. *Kate does not want others to post photos which show her doing workout and sweating in a gym near her house. She can set her LAMPi policy as* P_{Kate}*=⟨Loc={gym_address}, Typ=E, Int=anytime,* ξ*=Low⟩, where 'Loc' is set to the gym's address, 'Typ=E' indicates that this is an exact location, 'Int=anytime' means Kate wants the privacy protection whenever she is in this gym, '*ξ*=low' means that as long as she is not easily recognizable in the photo, Kate does not care about the photo being posted.*

Example 4.2.2. *Alice does not wish her face being recognized on any online photo that shows she is visiting a pub. She can thus set her LAMPi policy as* P_{Alice} = *⟨Loc={pub}, Typ=S, Int={8pm-5pm on any day},* ξ*=High⟩. That means if anyone attempts to post a pub photo with Alice in the background to a social network site, the social network site where Alice has registered and set the LAMPi policy will automatically replace Alice's face with a synthetic face to preserve Alice's privacy without affecting the photo owner's sharing experience.*

4.3 THE LAMP SYSTEM

In this section, we discuss how our proposed LAMP system helps preserve privacy of users who have no knowledge of their photos being posted by others. Figure 4.2 illustrates the data flow in the LAMP system. The LAMPi policy configuration function facilitates the users to specify the LAMPi policy through a graphic-based interface developed using Google Maps API. Users' policies will be indexed and stored in a policy database, and users' face features will be encoded to speed up the future face recognition. When someone wants to upload a photo to share, our LAMP system



Location	Type	Policy List
A1	E	P13, P15, P20, P30
A3	E	P40, P47
A2	E	P33
A5	E	P50, P41, P43, P29, P24
...	E	... ...
Anyplace	S	P3,P7,P9
Entertainment	S	P6,P11
Bar	S	P5
...	S	... ...

Figure 4.3: An Overview of the DLP-tree

will first retrieve policies which mark the photo location as sensitive. Among the owners of the retrieved policies, we will further check if their faces depicted on the photo. If so, their faces will be replaced with synthetic faces (or faces having no privacy concerns) to avoid undesired disclosure while maintaining the photo quality. In what follows, we elaborate the user identification and protection algorithms.

4.3.1 Identify Users with Location Privacy Concerns

In order to provide equal privacy protection to every person on the photo, an inevitable step is to know who (especially those in the background) are in the photo. A brute force method to identifying people in the background of the photo will need to compare the person's face on the photo against all the other users' faces in the social network site (e.g., 2.4 billion users in Facebook), which will be extremely computationally expensive and hardly possible to maintain real-time response for the photo uploading request. This has been a very challenging problem in the image privacy protection as also pointed out by Ilia et al. in [23].

To overcome this challenge, we aim to reduce the total number of faces that need to be compared for each uploaded photo. Correspondingly, we propose a hybrid data structure that indexes policies and helps significantly reduce the search space, making the face identification in large-scale user sets a feasible process for real-time applications.

Indexing LAMPi Policies

Given a photo and its location, we aim to quickly locate users who specify this location (based on the address) or this type of location (based on the semantic keywords) as sensitive so that in future occurrences we only need to compare these users' faces with those in the photo. To achieve this, we propose to index LAMPi policies based on locations, and group policies containing the same location together. Specifically, we store the LAMPi policies in a policy database implemented using PostgreSQL, and propose a DLP (Dual-Location-Policy) -tree to speed up the policy retrieval.

PostgreSQL was chosen for a variety of reasons. Most importantly is that PostgreSQL has great read and write performance compared to MySQL. Additionally, the PostGIS extension allows for geospacial data support that is incredibly useful when checking user and image locations. Finally, PostgreSQL easily supports concurrency in reading and writing that is crucial to speed up the face recognition process in our system as discussed in the later part of our approach.

The structure of the DLP-tree is illustrated in Figure 4.3. The DLP-tree consists of two main parts to index exact locations and semantic locations in the LAMPi policies, respectively. The left side of the DLP-tree organizes exact locations in a hierarchical way from nation (N), state (S), city (C) to address (A). Each entry in the leaf node is in the form of $\langle street, city, state, nation, \Gamma, PID\rangle$, where the first four attributes are the address specified in the policy PID, and Γ is the time and date interval when the location is considered sensitive. Nearby locations are grouped together in the same leaf node or sibling leaf nodes. An entry in an internal node is in the form of $\langle region, CPT \rangle$, where *region* described the region that covers all its child node pointed by CPT. In this way, the internal nodes can efficiently facilitate the LAMPi policy search.

Specifically, given a photo's location, we start the search from the root of the DLP-tree and traverse to the left side to find the node whose region encloses the photo's location. Then, we follow its child pointer to conduct the same boundary check in its child node until we reach the leaf level. In the leaf node, we further compare the photo's location against the locations specified in the LAMPi policies and retrieve those policies which mark the photo's location as sensitive. Note that the DLP-tree is different from a map since the DLP does not need to store all the places (e.g., all the addresses, all the cities) if no policies have been specified there yet. Given total n policies to be indexed, the CLP-tree reduces the linear search time $O(n)$ to approximately the height of the tree $O(log(n))$.

The right side of the DLP-tree indexes semantic locations based on the hierarchical relationship among their semantic meanings. In particular, semantic locations are first classified into the basic categories, such as "bar", "hospital", "shopping mall" and "company". Basic categories are further classified into more generic categories which are the upper level of the DLP-tree. For example, basic categories like "bar" and "shopping mall" can be classified as a more generic category: "entertainment", and basic categories like "hospital", "clinic", and "urgent care" can be classified as "medical". Unlike the top-down search in the left side of the DLP-tree, the search in this part of the DLP-tree is from the bottom to the top. This is because users are allowed to specify their sensitive locations using semantic words at different granularity. Some users may specify "entertainment" in their policies while some users may specify only "bar" in their policies. Thus, the user policies are attached to different levels of the DLP-tree correspondingly. An entry in a node of this side of the DLP-tree is in the form of $\langle \varpi, \Xi, PPT \rangle$, where ϖ is the semantic keyword specified in the list of LAMPi policies (denoted as Ξ), and PPT is the pointer to the parent node in the DLP-tree.

The following is an example of how to look up the LAMPi policies that use semantic locations. Given a photo depicting a group of people, assume that its tag indicates it is a bar. In order to check if anyone in this photo considers this place as sensitive, we will search the right side of the DLP-tree. Starting from the leaf level, we find the node that contains the keyword "bar" and retrieve the associated policy IDs. Then, we visit its parent node with the keyword "entertainment", and also retrieve all the policies associated with it. Again, we go up to the parent node with the keyword "any place", and retrieve all the policies.

The owners of the policies retrieved from the DLP-tree will be compared against the people on the photo using face recognition as discussed in the following subsection.

Speed Up Face Recognition

To speed up the individual face comparison, we adopt two strategies. One is to pre-compute the user's facial features when the user set up his/her LAMPi policies, which helps decrease the time taken for facial recognition when a photo is uploaded. The other is to employ multi-thread programming to conduct individual pairs of face recognition simultaneously.

Specifically, we calculate the facial feature using the *load_photo*(*image_path*) and *encode*(*image*) functions in an open source python face recognition tool by Geitgey [46]. As reported, this face recognition tool has achieve 99.38% accuracy. The *load_photo*(*image_path*) function loads an image from an image path using PIL. This image is then converted to a numpy array and returned. Then, the *encode*(*image*) function takes the image numpy array as the input, and utilizes a pre-trained algorithm from dlib's facial recognition library to convert the image numpy array to a

128 dimension facial description. We then store the 128 dimension face features along with the user ID for the future face recognition.

Given a new photo, we first detect faces on the photo using the $locations(image)$ in the Geitgey's face recognition tool. For the detected faces, we calculate their face features in the similar way as aforementioned. Then, we compare the face features of those in the photo with those associated with the retrieved LAMPi policies that specify the photo location as sensitive. The face comparison is conducted using the $compare(source, destination, tolerance)$ function in the face recognition tool, which takes a source facial feature, a destination facial feature, and a tolerance. The tolerance value is set based on the sensitiveness value in the corresponding LAMPi policy. The function calculates the Euclidean distance between the facial feature vectors and checks if the distance is below the tolerance value. If the distance is smaller than the tolerance value, the two faces are considered a match.

It is worth noting that social network sites can also use their existing face recognition tools when adopting our proposed privacy preservation function.

4.3.2 Protect Users with Location Privacy Concerns

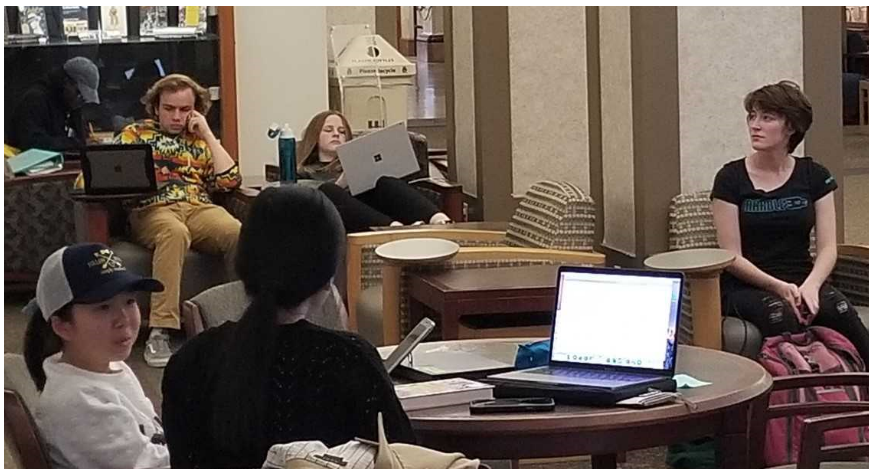
After the face recognition, we will obtain a set of users who are in the photo and concerned about their privacy at the photo location. For these users, we propose to replace their faces so that they will not be recognizable even if the photo is shared publicly. There have been several face replacement algorithms and software [47, 48]. We revised an open source software for the face replacement [48] and integrated it into the LAMP system.

Figure 4.2 illustrates a running example of how the LAMP system protects privacy. Assume that a student reporter took some photos on campus and plans to post them online to show the student life at a university in Paris. When she uploaded the photos to the social media site that deployed the LAMP system, the LAMP system will check each photo and conduct the following privacy preservation procedure.

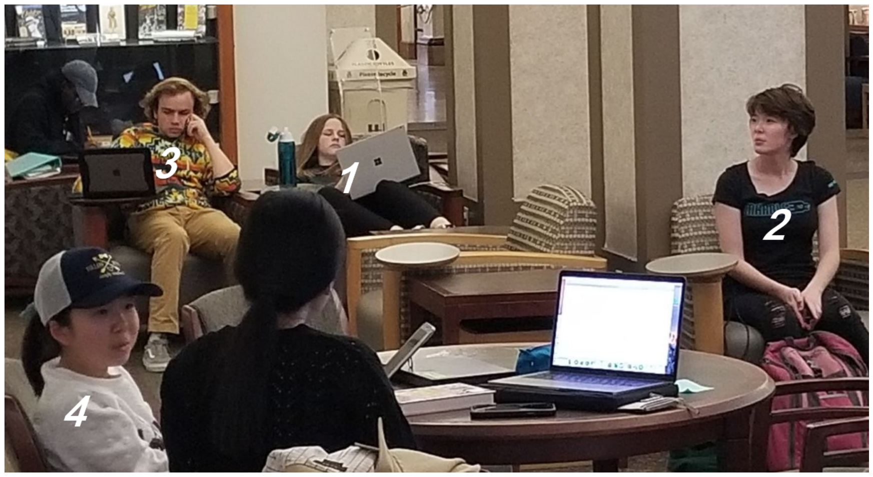
First, the LAMP extracts the photo's metadata to obtain the location information. In the example, the location information includes both the university address and the semantic keyword "university". Next, the LAMP system will search the policy database to find the policies which specify the photo's location as sensitive. For example, Bob was conducting an important business at Paris from 11/15/2019 to 12/15/2019, and he would like to keep his photos in Paris private as indicated by his policy P_i. Alice, a celebrity, was taking a year off to study abroad. Alice does not want to be followed or disturbed by her fans whenever she was at the university, and thus she has set her LAMPi policy as follows: P_j=⟨Loc=Universite Paris Diderot, Typ=E, Int=Anytime, ξ=High⟩. Note that Alice sets the time interval of protection to "Anytime" for convenience instead of using the exact time and date duration. She can simply remove this policy to release the protection after she returns home.

From the set of retrieved policies, the LAMP system next loads the face feature vectors of these policies' owners. These candidate face features will be used for face recognition, i.e., compared with faces on the uploaded photos which are highlighted in boxes in Figure 4.2. In the example, Alice's face was identified (pointed by the red arrow in the figure). Since Alice has wished to remain private in this location, the LAMP system will then help privatize this user through face replacement.

In our current implementation of the face replacement, a reference face needs to be manually selected to replace Alice's face. In the example, we chose another female's face who does not have privacy concerns at this location, and use it to replace Alice' face. Figure 4.4 shows the original photo uploaded by the user and the photo after the face replacement. Specifically, in Figure 4.4 (b), Alice's face (No.2 person) has been replaced with the face of the No. 4 person. Observe that the modified photo looks very natural. Therefore, we expect that the modified face will not raise special attention from users who are viewing the photo.



(a) Original Photo



(b) Twitter

Figure 4.4: Privatized vs Unchanged Photo Comparison

Future work in this would be to utilize an artificial face generation method so that the face replacement can be conducted automatically without selecting a candidate face that does not have privacy concerns. Moreover, besides face replacement, it may be interesting to incorporate techniques that replace other portions of a user such as a user's hair style and attire to prevent someone who is familiar with the user from identifying the person. However, we also argue that since current face replacement results in a natural look, people viewing the photo do not know the photo has been modified or not, and may not try hard to match each person in a photo with someone they know.

4.4 PRIVACY EVALUATION

In order to evaluate the effectiveness of our proposed privacy protection, we conducted another round of user study to see if participants are still able to identify the person who requires privacy protection and has been processed by our system. The idea is to present a set of testing photos and two reference photos to the participants, and ask them to try to identify the female and male references from the photos. Among the testing photos, some contain the referenced female and male without modification which represent the scenarios when people did not mark that location as sensitive; some contain the reference female and male with replaced faces which represent the scenarios when the people require privacy protection at that location; some contain blurred faces which represent the traditional privacy protection approach; and some do not contain any of the referenced people which are used as comparisons. The details of the user study are described as follows.

We have recruited a total of 102 participants on Mechanical Turk. There are 51 females and 51 males. Among them, 22% are 18 to 25 years old, 43% are between 26 and 35, 19% are between 36 and 45, 12% are 46 to 55, and 4% are above 56 years old. The user study is fully anonymous and follows the IRB exempted project guidelines.

Our study starts by telling participants that they will review images with numbered people within them. They will also have a reference photo for a person's face. They are told that if they can identify the reference person in the photo with a large degree of certainty, they just mark down the number of the person on the photo. They are also told that some photos may not include the reference person, and if they can not identify the reference person with a high degree of certainty, they need to input 0.

Each participant was asked to view 10 images. Each image contains 4 to 10 faces in the foreground and background. Half of these photos were asked about a male reference, and the remaining half were asked about a female reference. Photos for both male and female references included 1 photo with the reference not in the photo, 1 photo with the references whose face have been replaced, 1 photo with the references whose faces have been blurred, and 2 photos of the references in clear view. The photos in clear view give us an understanding of if the majority of participants can correctly identify the person within the image without any changes, and then we can compare that with how they react when they see images that have been modified. Lastly, we ask if they noticed any image that has been altered in some way by presenting them unaltered photos and photos with replaced faces. We summarize participants' responses using the misidentification ratio which is the percentage of participants who did not correctly identify the reference person in the photo. From the study, we have the following two major findings.

Finding 1: *Photos with replaced faces have on average the highest misidentification ratio.* Specifically, as shown in Table 4.1, 84% of participants did not recognize the male reference in the photos where his face is replaced. Similarly for the female reference, 77% of participants did not recognize her in the photo with her face swapped. Both ratios are higher than that for the photos with blurred faces. This is possibly because when a face is blurred in the photo, the participants of the study clearly know which face to examine, and they can pay closer attention to the person of the blurred face including checking the hair style and other features. Thus, more participants were able to guess that the blurred faces were the references with high confidence. When it comes to the photos with swapped faces, the participants do not know which face is swapped. There is more work for them to examine all the faces in the photo in great details. Thus, fewer people were able to correctly identify the references. In addition, the misidentification ratios for photos with full face shown are much lower than the misidentification ratio of modified faces.

Finding 2: *Unaltered and face swapped photos are hard to distinguish.* At the end of

Table 4.1: Privacy Evaluation Results

Image Type	**Misidentification Ratio**
Male reference with full face shown.	30%
Female reference with full face shown.	26%
Male reference with 40% of face showing.	76%
Female reference with 50% of face showing.	42%
Male reference not within photo.	44%
Female reference not within photo.	11%
Male reference with face blurred.	79%
Female reference with face blurred.	68%
Male reference with face swapped.	84%
Female reference with face swapped.	77%

the user study, we present an unaltered photo and a photo that contains a swapped face to the participants. For each photo, the participants were asked to check if the photo has been altered. The results are very interesting. Given an unaltered photo, 45% of participants said it had been altered. However, given the photo with a swapped face, only 32% of participants said it was altered. Such results could be because that participants believed that some photos must be altered since the survey asked the question, so they were taking a guess whether a photo had been altered even though they were not 100% sure. This interesting result indicates that it would be really hard for human eyes to distinguish a face swapped photo from unaltered photos.

The result from our study demonstrates the effectiveness of privacy protection of our proposed use of face replacement. Moreover, in the real world scenario, when an attacker suspects a blurred face, he can utilize deblurring technique to further verify. In contrast, the replaced faces may not even arouse any attention from viewers, and hence we expect much higher misidentification ratio in the real social media sites.

4.5 EFFICIENCY EVALUATION

In this section, we aim to examine the efficiency and scalability of our proposed LAMP system. All of our experiments were conducted on custom Intel based computer with an i5-6600k at 3.9 GHz turbo clock, and 24 GB of 2133 Mhz RAM (computer specification).

Users' LAMPi policies are synthetically generated. We randomly select exact locations and semantic locations along with time and date constraints for each user to create the LAMPi policies. In the experiments, we vary the number of sensitive locations (i.e., the number of policies) specified per user to test the efficiency of our algorithms.

We collected 5000 facial images from "Labeled Faces in the Wild" database [49]. These images are used to simulate uploaded images that we need to check the privacy compliance. Since not all the images that we collected associate with location information and our goal is to evaluate only the efficiency of our algorithms, we randomly generate location tags for each image. Each image is associated with both a coordinate location and 1 to 5 keywords indicating the semantic meanings of the location. The keywords for semantic locations are generated based on a four-level hierarchy similar to the one shown in Figure 4.3. In the experiments, we vary the total number of distinct locations to test their impact on our system performance.

(1) Varying the Total Number of Users: In the first round of experiments, we aim to evaluate how fast our system can retrieve users whose LAMPi policies specify the location of the uploaded photo as sensitive. We vary the total number of users from 100K to 1M. There are total 100K distinct locations. Each location has a unique address and 1 to 5 keywords. Each location may be marked as sensitive by 1000 users, thus resulting 100M LAMPi policies in total. There are 50% policies on exact locations and 50% on semantic-based locations.

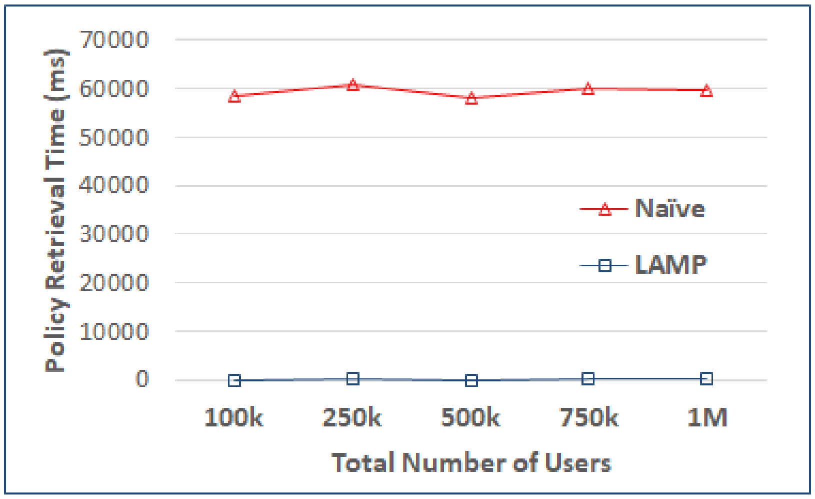


Figure 4.5: Effect of Varying the Total Number of Users

We compare the performance of our system with a naive solution which scans all the users' policies to identify the match. Figure 4.5 shows the experimental results. As we can see that, our LAMP system is thousands of times faster than the naive approach. Specifically, it took the naive algorithm about 1 minute to locate a policy, while the LAMP system needs only 50 milliseconds. This is because our LAMP system indexes policies based on their locations using the DLP-tree. Considering 100 entries per node in the DLP-tree, 5 levels of the tree will be able to index about 1 billion policies. In other words, given a location either an address or a semantic keyword, we only need to check a few nodes (a few hundred entries out of 1 billion) in the DLP-tree to locate the group of policies that specify this location as sensitive. The naive approach will have to check every policy to see if the policy specifies the photo's location as sensitive, and hence it is extremely slow.

From figure 4.5, we also observe that both approaches are not affected much by the total number of users. This is because the policy retrieval time is dominated by the total number of policies rather than the total number of users. In this experiment, the total number of policies is 100M regardless of the increase of the users. The differences among the test datasets mainly reside in the policy owners who are selected from a 100K user pool or a 1M user pool.

(2) Varying the Number of Distinct Locations: In this round of experiments, we evaluate the effect of the total number of distinct locations. We vary the number of distinct locations from 10K to 100K. We set the total number of users to 1M and the number of policies per location to 1000. Under this setting, the total number of policies ranges from 10M to 100M, and each user has 10 to 100 policies, i.e., each user specifies 10 to 100 locations as sensitive.

Figure 4.6 compares the performance of our approach and the naive approach. Observe that the time taken to retrieve the policies for a given photo by the naive approach increases dramatically with the growth of the number of distinct locations. In contrast, our LAMP approach achieves constant and efficient performance in all the cases. Specifically, the naive approach requires 10 to 50 seconds when the number of locations increases from 10K to 100K, whereas our approach only needs 19 milliseconds to 55 milliseconds. This again demonstrates the effectiveness of the policy organization by the LAMP approach. Unlike the naive approach, which needs to check all the policies to find those specifying the photo location as sensitive, the search strategy adopted by our LAMP system significantly reduces the search scope. The size of the DLP-tree is only logarithmic to the number of distinct locations. Therefore, the processing time of the LAMP system only increases a little.

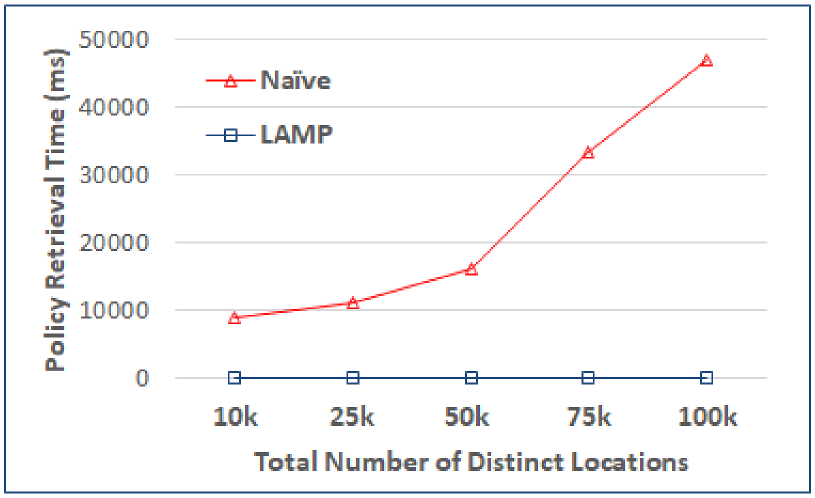


Figure 4.6: Effect of Varying the Total Number of Distinct Locations

(3) Varying the Number of Distinct Semantic Keywords: When testing semantic-based policies, we also set the total number of users to 1M and the total number of exact locations to 100K. Each location is associated with 5 semantic keywords. We vary the total number of distinct keywords from 250 to 5000. 5000 categories of places are considered an extreme case in the real world scenario, and hence we think it is sufficient to be used to test the scalability of our approach.

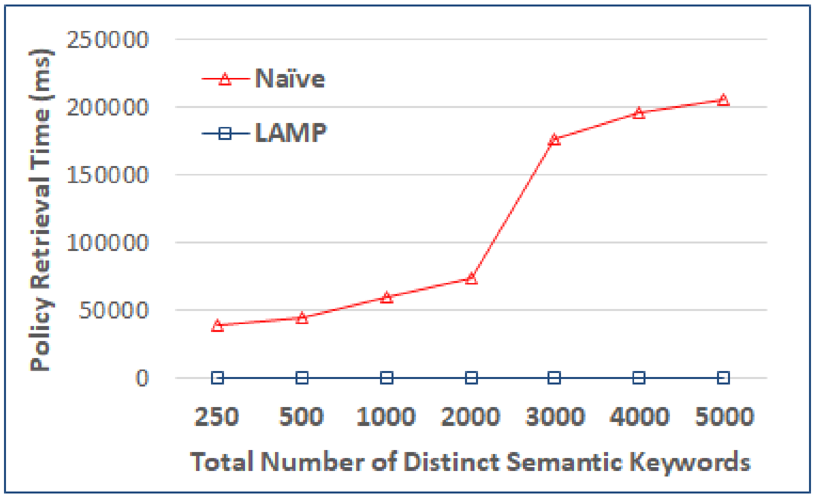


Figure 4.7: Effect of Varying the Total Number of Distinct Semantic Keywords

Figure 4.7 illustrates the experimental results which demonstrate a similar trend as that in the previous experiment regarding the policies on exact locations. Specifically, our LAMP system requires only 49 milliseconds to retrieve policies in the dataset with 250 distinct semantic keywords and 55 milliseconds for the dataset with 5000 distinct keywords. The naive approach took much longer time (more than 200 seconds) especially when the number of distinct keywords increases. This is because the naive approach needs to look through each different semantic keyword until find the matching policies.

(4) Varying the Number of Policies per Exact Locations: Next, we examine the effect of the number of policies per exact location. We fix the total number of users to be 1M and the total number of distinct locations to be 100K. We vary the number of policies per location from 0.1% to 1% of the total number of users. That means the number of policies per location ranges from 1M×0.1%=1000 to 1M×1%=10,000. The total number of policies ranges from 1M to 1 billion. This corresponds to 1 to 1000 policies per user.

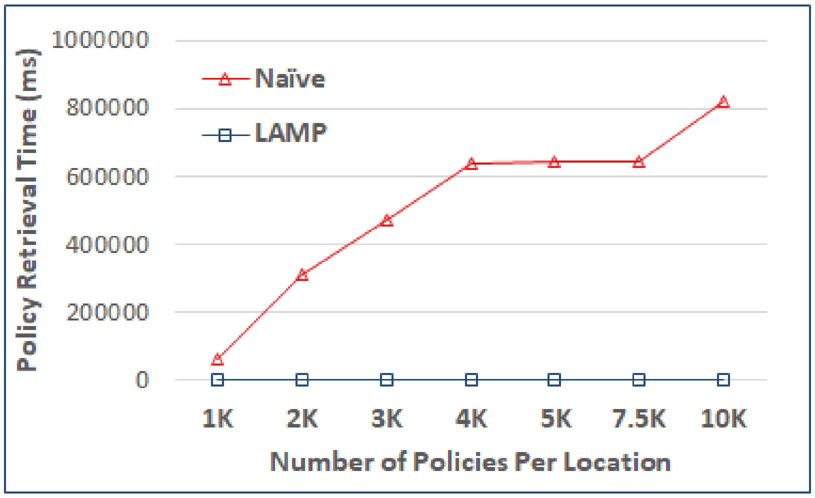


Figure 4.8: Effect of Varying the Number of Policies per Exact Location

The performance comparison between our LAMP system and the naive approach is shown in Figure 4.8. Observe that the LAMP system significantly outperforms the naive approach. It took a similar amount of time for the naive approach regardless the number of policies per location since the naive approach always needs to check all the policies in all locations for a photo. As for the LAMP system, it only checks the policies that contain the photo's location. Thus, when there are more policies per location to be retrieved, the cost of the LAMP system increases slightly. It is worth noting that our system is still considerably fast as it needs only 128 milliseconds to find the candidate policies among 1 billion policies.

(5) Performance of Face Recognition and Replacement: After locating candidate users who specify the photo's location as sensitive, the next step is to check if the user actually appears on the photo through face recognition. Note that in our system, we only need to compare faces in the photo with candidate users who specify the photo's location as sensitive, rather than the users in the whole social network. Therefore, we vary the number of candidate users from 100 to 5000 whereby the value 5000 conforms with our previous setting of 5000 policies per location. We select a group photo that contains 18 faces as the photo to be uploaded. That means, there will be up to 5000*18=90,000 face comparisons. We tested both linear and parallel face recognition performance. As shown in Figure 4.9, the linear algorithm starts to struggle with the increase of the number of candidate users. Our parallel algorithm performs relatively constantly and the face recognition time stays below 100 milliseconds in all cases. We expect the face recognition to be even faster at the real server which has better hardware and can spawn more concurrent threads.

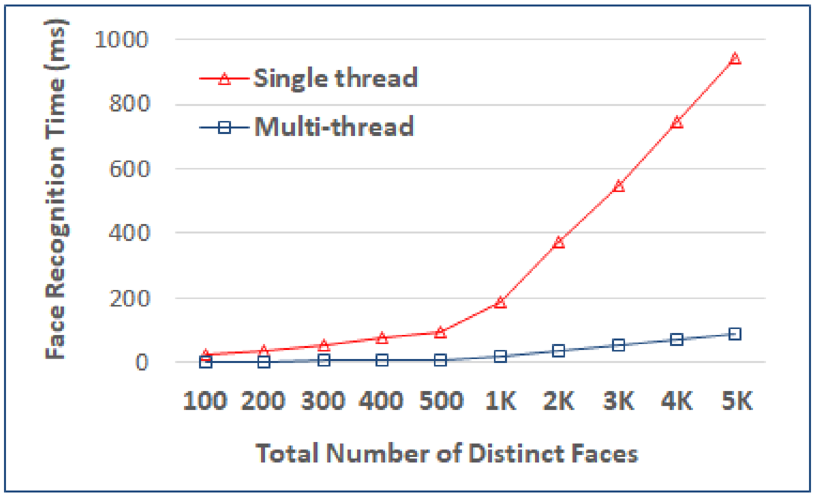


Figure 4.9: Face Recognition Time (18 faces in the uploaded photo)

We also check how long it takes to replace a user's face. Since face replacement is not the research focus of our work, we adopt an existing open source software [48] to gain the idea of the time needed for face swapping. We observe that it took about 8 seconds to replace a face. We expect the face replacement to be faster at the server side, and a few seconds of delay before the photo go live online would not be very noticeable by users considering there are also network and webpage refresh delays.

4.6 RECAP

In this chapter, we propose a novel idea to address the increasing concerns of location tracking of an individual through online images posted by others. Specifically, we define a new access control model, namely Location-Aware Multi-Party image (LAMPi) access control, which goes beyond the traditional access control that offers protection to only the owners of the image. Our proposed LAMPi access control mechanism provides equal privacy protection to every human subject on an online photo, no matter the human subject is the owner of the image or not, is at the foreground or background of the image. We also design an efficient policy management system that leverages policy indexing techniques and uses face replacement as policy enforcement, and we achieve the privacy protection in real time of photo uploading process. Our user studies and experimental results on the system prototype demonstrate both effectiveness and efficiency of our approach.

Chapter 5

SUMMARY

In this dissertation, we introduce two novel privacy protection methods related online social networks. These novel protection methods focus on the friend search engine, and image content. This covers a broad view of social network privacy violations which can occur from the design of the social network, and posts people may not be able to control.

Firstly we introduced a method to secure the friend search engine. Currently due to mismatched privacy settings it is possible to get an accurate idea of someone's friend list even if they have set this feature to private. While a previous work proposed a solution to this [9], they found their defense to be insecure against collusion attacks [11, 12]. We instead construct a sub-graph of the original network and use this for the friend search engine. By forcing mutual connections and restricting connections to each user's friend list limit, it is not possible for attackers to gain more information than is shared by the user's settings. This method is robust against attacks, and minimally impacts an online social network allowing it to be easily adopted.

Secondly we a novel image content privacy protection method. With the explosion of high resolution cameras and the ability to share content through online social networks, these images can reach a large audience. To protect people identifiable within a photo, we propose a location based privacy setting called LAMPi. This system allow user's to set locations and times in which they do not want their face to be seen. As a photo is posted, users depicted in the image with conflicting policies to the photos location and time will have their face replaced. This method lessens the impact on the image, and allows users to remain private. This system not only secures the photo owner or the photo owners friends, but is incredibly helpful in protecting the privacy of people identifiable in the background of an image. Through surveys, many people can not identify the people after facial replacement. Additionally compared to other approaches, our system takes less than a few seconds to protect the privacy of incredibly large networks. This allows people to remain private in photos which they cannot control, while retaining efficient runtime if adopted.

These methods aim to enhance an online social network's ability to keep users safe. If a user sets their privacy settings, these should protect them in all cases. With the proposed methods, users can remain more confident in a social network and feel secure that their settings will be enforced. This will not only help users but will additionally help online social networks to retain users who wish to have privacy online.

Chapter 6

CONCLUSION

This thesis examines two privacy issues which are the friend search engine, and image content. These two works respectively protect the design of a social network, and content a user may wish to privatize. Firstly we introduce FriendGuard, which protects users against friend list overexposure by creating a sub-graph which gives no more information than a user's own privacy settings. Secondly we introduce LAMPi, which privatizes a user's face within an image based on their settings and the metadata contained within a posted photo. Each of these works help to further the privacy that users have online and will allow them to be more confident that their privacy settings will work as designed. All works introduced in this thesis have minimal overhead, allowing them to be adopted by existing social networks without sacrificing usability.

BIBLIOGRAPHY

[1] A. Besmer **and** H. Richter Lipford. “Privacy Perceptions of Photo Sharing in Facebook”. **in**: (**january** 2008).

[2] *Facebook*. https://www.facebook.com/. **november** 2019.

[3] A. Besmer **and** H. Richter Lipford. “Moving Beyond Untagging: Photo Privacy in a Tagged World”. **in**: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’10. Atlanta, Georgia, USA: ACM, 2010, **pages** 1563–1572. ISBN: 978-1-60558-929-9. URL: `http://doi.acm.org/10.1145/1753326.1753560`.

[4] *Social, Digital Video Drive Further Growth in Time Spent Online*. **may** 2013. URL: `https://www.emarketer.com/Article/Social-Digital-Video-Drive-Further-Growth-Time-Spent-Online/1009872`.

[5] P. Morrissey. *6 People Who Were Fired for Social Media Posts*. Website. **november** 2019.

[6] J. Boone. *Florida Teacher Olivia Sprauer Fired for Bikini Modeling Pics*. Website. **october** 2013.

[7] J. Bonneau, J. Anderson, R. Anderson **and** F. Stajano. “Eight friends are enough: Social graph approximation via public listings”. **in**: *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems* (2009), **pages** 13–18.

[8] A. Yamada, T. Kim **and** A. Perrig. “Exploiting privacy policy conflicts in online social networks”. **in**: (2012), **pages** 1–9.

[9] N. Li. “Privacy-aware display strategy in friend search”. **in**: *IEEE International Conference on Communications (ICC)* (2014), **pages** 945–950.

[10] W. Ren, S. Huang, Y. Ren **and** C. Kim-Kwang. “LiPISC: A Lightweight and Flexible Method for Privacy-Aware Intersection Set Computation”. **in**: (2016).

[11] Y. Liu **and** N. Li. “Retrieving Hidden Friends: A Collusion Privacy Attack Against Online Friend Search Engine”. **in**: *IEEE Transactions on Information Forensics and Security* 14.4 (2019), **pages** 833–847.

[12] Y. Liu **and** N. Li. “An Advanced Collusion Attack against User Friendship Privacy in OSNs”. **in**: *IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)* 1 (2016), **pages** 465–470.

[13] L. Yuan, J. Theytaz **and** T. Ebrahimi. “Context-Dependent Privacy-Aware Photo Sharing Based on Machine Learning”. **in**: *IFIP Advances in Information and Communication Technology.* 2017, **pages** 93–107.

[14] A. C. Squicciarini, S. Sundareswaran, D. Lin **and** J. Wede. “A3P: Adaptive Policy Prediction for Shared Images over Popular Content Sharing Sites”. **in**: *Proceedings of the 22Nd ACM Conference on Hypertext and Hypermedia.* HT ’11. Eindhoven, The Netherlands, 2011, **pages** 261–270. ISBN: 978-1-4503-0256-2.

[15] D. Hu, F. Chen, X. Wu **and** Z. Zhao. “A Framework of Privacy Decision Recommendation for Image Sharing in Online Social Networks”. **in**: **june** 2016, **pages** 243–251.

[16] G. Sun, Y. Xie, D. Liao, Y. Hongfang **and** V. Chang. “User-Defined Privacy Location-Sharing System in Mobile Online Social Networks”. **in**: *Journal of Network and Computer Applications* 86 (**november** 2016).

[17] X. Xiao, C. Chen, A. Sangaiah, G. Hu, R. Ye **and** Y. Jiang. "CenLocShare: A centralized privacy-preserving location-sharing system for mobile online social networks". **in**: *Future Generation Computer Systems* 86 (**february** 2017).

[18] H. Hu, G. Ahn **and** J. Jorgensen. "Enabling Collaborative data sharing in Google+". **in**: *2012 IEEE Global Communications Conference (GLOBECOM)*. 2012, **pages** 720–725.

[19] F. Li, Z. Sun, A. Li, B. Niu, H. Li **and** G. Cao. "HideMe: Privacy-Preserving Photo Sharing on Social Networks". **in**: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. 2019, **pages** 154–162.

[20] J. Yu, B. Zhang, Z. Kuang, D. Lin **and** J. Fan. "iPrivacy: Image Privacy Protection by Identifying Sensitive Objects via Deep Multi-Task Learning". **in**: *IEEE Transactions on Information Forensics and Security* 12.5 (2017), **pages** 1005–1016.

[21] H. Hu **and** G.-J. Ahn. "Multiparty Authorization Framework for Data Sharing in Online Social Networks". **in**: *Proceedings of the 25th Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy*. DBSec'11. Richmond, VA, 2011, **pages** 29–43. ISBN: 978-3-642-22347-1.

[22] H. Hu, G. Ahn **and** J. Jorgensen. "Multiparty Access Control for Online Social Networks: Model and Mechanisms". **in**: *IEEE Transactions on Knowledge and Data Engineering* 25.7 (2013), **pages** 1614–1627.

[23] P. Ilia, I. Polakis, E. Athanasopoulos, F. Maggi **and** S. Ioannidis. "Face/Off: Preventing Privacy Leakage From Photos in Social Networks". **in**: *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*. CCS '15. Denver, Colorado, USA: ACM, 2015, **pages** 781–792. ISBN: 978-1-4503-3832-5. URL: `http://doi.acm.org/10.1145/2810103.2813603`.

[24] Y. Rashidi, T. Ahmed, F. Patel, E. Fath, A. Kapadia, C. Nippert-Eng **and** N. M. Su. “”You don’t want to be the next meme”: College Students’ Workarounds to Manage Privacy in the Era of Pervasive Photography”. **in**: *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*. Baltimore, MD, 2018, **pages** 143–157. ISBN: 978-1-939133-10-6.

[25] L. Zhu, L. Jin, J. Zhu, Z. Li, Z. Tian **and** H. Lu. “Blind Image Deblurring Based on Local Rank”. **in**: *Mobile Networks and Applications* (**september** 2019). ISSN: 1572-8153. URL: `https://doi.org/10.1007/s11036-019-01375-8`.

[26] L. Pan, R. Hartley, M. Liu **and** Y. Dai. “Phase-Only Image Based Kernel Estimation for Single Image Blind Deblurring”. **in**: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. **june** 2019.

[27] L. Li, J. Pan, W.-S. Lai, C. Gao, N. Sang **and** M.-H. Yang. “Blind Image Deblurring via Deep Discriminative Priors”. **in**: *International Journal of Computer Vision* 127.8 (**august** 2019), **pages** 1025–1043. ISSN: 1573-1405. URL: `https://doi.org/10.1007/s11263-018-01146-0`.

[28] A. C. Squicciarini, S. Sundareswaran, D. Lin **and** J. Wede. “A3P: adaptive policy prediction for shared images over popular content sharing sites”. **in**: *HT’11, Proceedings of the 22nd ACM Conference on Hypertext and Hypermedia, Eindhoven, The Netherlands, June 6-9, 2011*. 2011, **pages** 261–270.

[29] A. C. Squicciarini, D. Lin, S. Sundareswaran **and** J. Wede. “Privacy Policy Inference of User-Uploaded Images on Content Sharing Sites”. **in**: *IEEE Transactions on Knowledge and Data Engineering* 27.1 (2015), **pages** 193–206.

[30] A. Squicciarini, D. Lin, S. Karumanchi **and** N. DeSisto. “Automatic social group organization and privacy management”. **in**: *8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*. 2012, **pages** 89–96.

[31] D. Lin, E. Bertino, R. Cheng **and** S. Prabhakar. "Location Privacy in Moving-Object Environments". **in**: *Trans. Data Privacy* 2.1 (**april** 2009), **pages** 21–46. ISSN: 1888-5063.

[32] L. Lan **and** L. Tian. "Preserving Social Network Privacy Using Edge Vector Perturbation". **in**: *International Conference on Information Science and Cloud Computing Companion* (2013), **pages** 188–193.

[33] S. Bourahla **and** Y. Challal. "Social Networks Privacy Preserving Data Publishing". **in**: *International Conference on Computational Intelligence and Security (CIS)* (2017), **pages** 258–262.

[34] A. Squicciarini, S. Karumanchi, D. Lin **and** N. Desisto. "Identifying Hidden Social Circles for Advanced Privacy Configuration". **in**: *Computer and Security* 41 (2014), **pages** 40–51.

[35] F. Nagle **and** L. Singh. "Can Friends Be Trusted? Exploring Privacy in Online Social Networks". **in**: *International Conference on Advances in Social Network Analysis and Mining* (2009), **pages** 312–315.

[36] B. Henne, C. Szongott **and** M. Smith. "SnapMe if you can: Privacy threats of other peoples' geo-tagged media and what we can do about it". **in**: **april** 2013, **pages** 95–106.

[37] K. Albrecht **and** L. McIntyre. "Psst...Your Location Is Showing!: Metadata in digital photos and posts could be revealing more than you realize". **in**: *IEEE Consumer Electronics Magazine* 4.1 (2015), **pages** 94–96.

[38] K. Ghazinour **and** J. Ponchak. "Hidden Privacy Risks in Sharing Pictures on Social Media". **in**: *Procedia Computer Science* 113 (**december** 2017), **pages** 267–272.

[39] D. Keerthi Chandra, W. Chowgule, Y. Fu **and** D. Lin. "RIPA: Real-Time Image Privacy Alert System". **in**: *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*. 2018, **pages** 136–145.

[40] J. McAuley **and** J. Leskovec. "Learning to Discover Social Circles in Ego Networks". **in**: (2012).

[41] S. Kumar, F. Spezzano, V. Subrahamanian **and** C. Faloutsos. "Edge Weight Prediction in Weighted Signed Networks". **in**: *IEEE 16th International Conference on Data Mining (ICDM)* (2016), **pages** 221–230.

[42] S. Kumar, B. Hooi, D. Makhija, M. kumar, V. Subrahamanian **and** C. Faloutsos. "REV2: Fraudulent User Prediction in Rating Platforms". **in**: *11th ACM International Conference on Web Search and Data Mining (WSDM)* (2018).

[43] A. Smith. *What people like and dislike about Facebook.* **february** 2014. URL: `http://www.pewresearch.org/fact-tank/2014/02/03/what-people-like-dislike-about-facebook/`.

[44] L. Price. *20 Tales of Employees Who Were Fired Because of Social Media Posts.* `https://people.com/celebrity/employees-who-were-fired-because-of-social-media-posts/`. 2016.

[45] T. Germain. *How a Photo's Hidden 'Exif' Data Exposes Your Personal Information.* `https://www.consumerreports.org/privacy/what-can-you-tell-from-photo-exif-data/`. 2019.

[46] A. Geitgey. *Facial Recognition.* `https://github.com/ageitgey/face\_recognition/blob/master/LICENSE`. MIT License. 2017.

[47] H. Guo, D. Niu, X. Kong **and** X. Zhao. "Face Replacement Based on 2D Dense Mapping". **in**: *Proceedings of the 2Nd International Conference on Image and Graphics Processing.* ICIGP '19. Singapore, Singapore: ACM, 2019, **pages** 23–28. ISBN: 978-1-4503-6092-0. URL: `http://doi.acm.org/10.1145/3313950.3313964`.

[48] wuhuikai, XiangFugui **and** niczem. *FaceSwap.* `https://github.com/wuhuikai/FaceSwap`. 2018.

[49] G. B. Huang, M. Ramesh, T. Berg **and** E. Learned-Miller. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments.* techreport 07-49. University of Massachusetts, Amherst, **october** 2007.

VITA

Joshua Dennis Morris was born in Chicago, Illinois. He graduated Magna Cum Laude from Missouri University of Science and Technology in 2018 with a Bachelor of Science in Computer Science. Joshua received the Scholarship for Service to obtain his Ph.D. He completed his Ph.D from University of Missouri - Columbia in December 2021 under Dr. Dan Lin. Joshua has accepted a Research and Development position at Sandia National Laboratories.