

Secure and Efficient Nearest Neighbour Search in High Dimensional Space

by

Kazi Wasif Ahmed

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
December 2017

© Copyright 2017 by Kazi Wasif Ahmed

Thesis advisor

Noman Mohammed

Author

Kazi Wasif Ahmed

Secure and Efficient Nearest Neighbour Search in High Dimensional Space

Abstract

The attractive features of cloud platforms such as low cost, high availability and scalability are encouraging social networks, health and other service providers to outsource their client data to the cloud. Though there are many advantages of using cloud-based solutions, the privacy of the outsourced data is a major concern. Compromised cloud servers can leak sensitive information about users such as the incident of the iCloud celebrity data leakage. One practical solution to mitigate these concerns is to encrypt or anonymize the data before outsourcing to the cloud. Although encryption protects the data from unauthorized access, it increases the computational complexity to execute the required functions (e.g., similarity or nearest neighbour search), which is the key requirement for different social discovery applications. On the other hand, anonymization supports privacy-preserving fast computation but inefficient anonymization may result in huge data utility loss. In this thesis, I have designed an efficient approach to perform the secure nearest neighbour search in high dimensional space. The proposed framework utilizes the advantages of Intel® Software Guard Extensions (Intel® SGX) architecture and efficient anonymization methods to perform the secure nearest neighbour search.

Contents

Abstract	ii
Table of Contents	v
List of Figures	vi
List of Tables	vii
Acknowledgments	1
1 Introduction	2
1.1 Motivation	2
1.2 Contributions	4
1.3 Organization	8
2 Nearest Neighbour Search under Anonymity Constraint	9
2.1 Introduction	9
2.2 System Overview	11
2.2.1 Architecture and Entities	11
2.2.2 System Service Flow	12
Storing Data in the Cloud Server	12
Social Discovery Request	13
2.2.3 Threat Model	13
2.3 Problem Formulation	14
2.3.1 Image Data	14
2.3.2 Privacy Requirement	15
2.3.3 Utility Requirement	16
2.4 Proposed Approach	17
2.4.1 Preprocessing Steps at the Service Provider End	17
2.4.2 Processing Steps at Cloud Server End	19
2.5 Experiment and Result	25
2.5.1 Implementation Detail	25
2.5.2 Image Profile Evaluation	25
2.5.3 Accuracy Analysis	26
2.6 Related Work	27
2.6.1 Cryptographic Approach	28
2.6.2 Anonymization Based Approach	28

3	Nearest Neighbour Search over Encrypted Data using SGX	30
3.1	Introduction	30
3.1.1	Challenges	30
3.1.2	Contributions	31
3.2	System Overview	33
3.2.1	Architecture	34
3.2.2	System Service Flow	34
3.2.3	Assumptions and Adversary Model	35
3.3	Background	36
3.3.1	Intel SGX Architecture	36
	Memory Protection	36
	Enclave Access	36
	Sealing and Attestation	37
3.3.2	Nearest Neighbour Search	37
3.4	Problem Definition	40
3.4.1	Source Data	40
3.4.2	Security Requirement	41
3.4.3	Utility Requirement	41
3.5	Secure Nearest Neighbour Search Using SGX	42
3.5.1	Service Provider	42
	Feature Extraction	43
	Search Index Construction	44
	Encryption Phase	49
3.5.2	Cloud Server	50
	Decryption Phase	51
	Similarity Computation	51
3.6	Analysis	52
3.7	Experimental Evaluation	56
3.7.1	Implementation Details	56
3.7.2	NNS Performance Evaluation	58
	Symmetric Cryptosystem	59
	Asymmetric Cryptosystem	60
3.7.3	Accuracy Analysis	60
3.8	Related Work	63
3.8.1	Cryptographic Approach	63
3.8.2	Secure Hardware-based Solutions	64
4	Obfuscated Image Classification and Nearest Neighbour Search for Friend Recommendation	65
4.1	Introduction	65
4.1.1	Privacy Implications	67
4.1.2	Privacy Protection	68
4.1.3	Contributions	69
4.2	System Overview	70
4.2.1	Architecture and Entities	71

4.2.2	System Service Flow	72
4.2.3	Problem Definition	72
4.2.4	Threat Model	74
4.3	Methodology	75
4.3.1	Image Obfuscation Techniques	75
4.3.2	Feature Extraction	77
4.4	Proposed Framework	80
4.4.1	Image Obfuscation	80
4.4.2	Obfuscated Image Classification	81
4.4.3	User Profile Generation and Friend Recommendation	83
4.4.4	Security Analysis	85
4.5	Experiment and Result	85
4.5.1	Implementation Details	85
4.5.2	Datasets	86
4.5.3	Accuracy Analysis	87
4.6	Related Work	89
5	Conclusion	91
5.1	Summary	91
5.2	Future Work	92
	Bibliography	107

List of Figures

1.1	Research Problems Addressed in this Thesis	5
2.1	Proposed architecture of the system	11
2.2	Evaluation of image profiles consisting of 50 images using the proposed approach	24
2.3	Accuracy comparison varying anonymization factor(k)	27
3.1	Architecture	33
3.2	Top-5 query on a image-centric social discovery	42
3.3	Computation time for variable dataset size in X-axis considering 17 features for 4 different similarity metrics in plain-text for a top-10 query (Naive Approach)	56
3.4	Computation time for variable dataset size (30k, 40k, 50k) in X-axis considering 17 features for 4 different similarity metrics for a top-10 query	57
3.5	Computation time for variable feature set size (10, 20, 30) in X-axis considering fixed data set size 20k for 4 different similarity metrics for a top-10 query	58
3.6	Computation time for variable records size (in X-axis) considering 17 features for 4 different similarity metrics using asymmetric cryptosystem for a top-10 query (Naive Approach)	61
3.7	Accuracy for NNS on different top- k -values using LSH with 4 different Similarity metrics for data set size 50k	62
4.1	Proposed architecture of the system	71
4.2	Finding top-5 similar users based on image-centric interest	73
4.3	Different Image Obfuscation Techniques	76
4.4	Proposed Architecture of Obfuscated Image Classification	77
4.5	Image Obfuscation (MirFlickr-25k Dataset)	84

List of Tables

2.1	Sample dataset with binary feature attributes	17
2.2	Hash functions computed for the user profile matrix	21
2.3	Computing minhash signatures (Step-1)	22
2.4	Computing minhash signatures (Step-2)	22
2.5	Computing minhash signatures (Step-3)	22
2.6	Computing minhash signatures (Final Step)	22
3.1	Comparisons among previous schemes for the nearest neighbour search problem. Our proposal is the only technique that enables cloud to efficiently compute the final result without any post-processing by a trusted entity.	32
3.2	Sample dataset with binary feature attributes	41
3.3	Hash functions computed for the user profile matrix	46
3.4	Computing MinHash signatures (Step-1)	46
3.5	Computing MinHash signatures (Step-2)	47
3.6	Computing MinHash signatures (Step-3)	47
3.7	Computing MinHash signatures (Final Step)	48
4.1	Comparisons among previous schemes for obfuscated image classification problem. Our proposal is the only technique that provides friend recommendation service using obfuscated image classification	69
4.2	Sample user profiles with binary feature attributes	74
4.3	Classification Accuracy on MNIST Dataset.	88
4.4	Classification Accuracy on CIFAR-10 Dataset	88
4.5	Classification Accuracy on Mirflickr-25k Dataset	88

Acknowledgments

I would like to express my gratitude to Almighty God for granting me the opportunity to write this thesis.

I would like to thank my supervisor, Dr Noman Mohammed for giving me the opportunity to work under his supervision. I am very grateful for his guidance, suggestions, and feedback throughout the preparation of this thesis. I am thankful to the other members of my committee, Drs. Yang Wang, Parimala Thulasiraman and Dima Alhadidi for the comment and advice they provided.

I am also grateful to all my collaborators, Md Momin Al Aziz, Md Zahidul Hasan, Omit Chanda (in no particular order) for their guidance. I am also thankful to my colleagues Nazmus Sadat, Safiur Rahman Mahdi, Reza Ghesemi, Md Waliullah and Toufique Morshed for their valuable consults and time. I am equally grateful to all the benevolent faculties of University of Manitoba, Computer Science and Islamic University of Technology (IUT) who shaped my research and academic standing.

Finally, I would like to thank *my family* for their countless sacrifices and inspirations throughout the years.

Chapter 1

Introduction

1.1 Motivation

The advent of Ubiquitous Computing and Internet of Things (IoT) has given the scientific community a vast amount of data such as media (images, videos), network information (internet, network monitoring), genomic data, medical data (personal biometrics), etc [1]. For example, according to the user statistics published by Instagram in May 2014, it had 200 million registered users with 20 billion photographs shared up to that date and the number of shared images per day via Instagram were on average 60 million [2]. Now that we have accumulated this volume of personal data, terabytes, and petabytes of it, what should be the necessary steps to securely store and utilize these databases?

One of the potential solutions to mitigate the storage constraint problem is outsourcing the data to a cheaper third-party cloud server. The popularity of using cloud services as a back-end data center has increased due to high availability, low cost, and maintenance of cloud storage lately. Consequently, some of the pop-

ular social media sites have adopted cloud as a back-end data center. For example, Instagram was using the Amazon cloud before Facebook acquired them in 2014 [3]. However, some recent attacks on the stored data in the cloud (e.g., iCloudleak 2014) depicts the scenario that cloud servers are not to be entirely trusted with private information. Therefore, these data should go through a privacy-preserving mechanism before outsourcing to the cloud server. However, it is difficult to perform any useful computation on the secured (i.e., encrypted or anonymized) data.

Broadly, the goal is to extract useful information from the securely stored data. For example, a social networking service provider is recommending friends to its users based on the profile similarity or preferences. The similarity among users of the social network is measured considering different parameters like social relationships or graphs. Images can be considered as one of the most dominating parameters as it can provide a deep insight of an individual's preferences [4; 5]. In particular, similar people are more likely to interact with each other and have a connection among them. Therefore, if the contents of the uploaded images of a user in social networking sites are similar to another user, they might have a common interest [4; 5; 6].

The basic idea is to build user profiles aligned to the preferences of a user and perform a similarity computation over the user profiles. One such similarity computation operation is *Nearest Neighbour Search* which is one of the principal functionality of any social discovery or recommendation applications. Social discovery or recommendation based on nearest neighbour search is one of the trending research problems in recent times [6; 4]. The nearest neighbour search problem consists of a set of data elements with defined attributes (age, likes, affiliations, etc.) and a query that has specified values known as the target attributes.

The distance between the query and the data elements is measured by a specified metric. The objective is to find out the closest or most similar elements whose similarity score is greater than a predefined threshold value. Unfortunately, the task of measuring similarity among millions of records in real time is computationally expensive. Moreover, computational complexity increases dramatically when the data is securely stored (encrypted). On the other hand, data utility may be lost due to inappropriate anonymization technique. Hence, developing efficient solutions for performing secure nearest neighbour search operation is an important research problem [7; 8; 9].

1.2 Contributions

This thesis addresses three research problems (depicted in Figure 1.1) about secure nearest neighbour search over securely stored data in the cloud. The security guarantee is gradually improved by adopting different methodologies during nearest neighbour search as shown in Figure 1.1. Following we detail the technical contributions of this thesis.

Nearest Neighbour Search under Anonymity Constraint

Image sharing is one of the most popular features facilitated by different social media sites such as Facebook, Flickr, Pinterest and Instagram. The contribution of this chapter is designing a practical solution for image centric social discovery based on privacy-preserving nearest neighbour search under anonymity constraint. The proposed method extracts the content-rich visual features of images and applies an efficient anonymization scheme based on optimum feature selec-

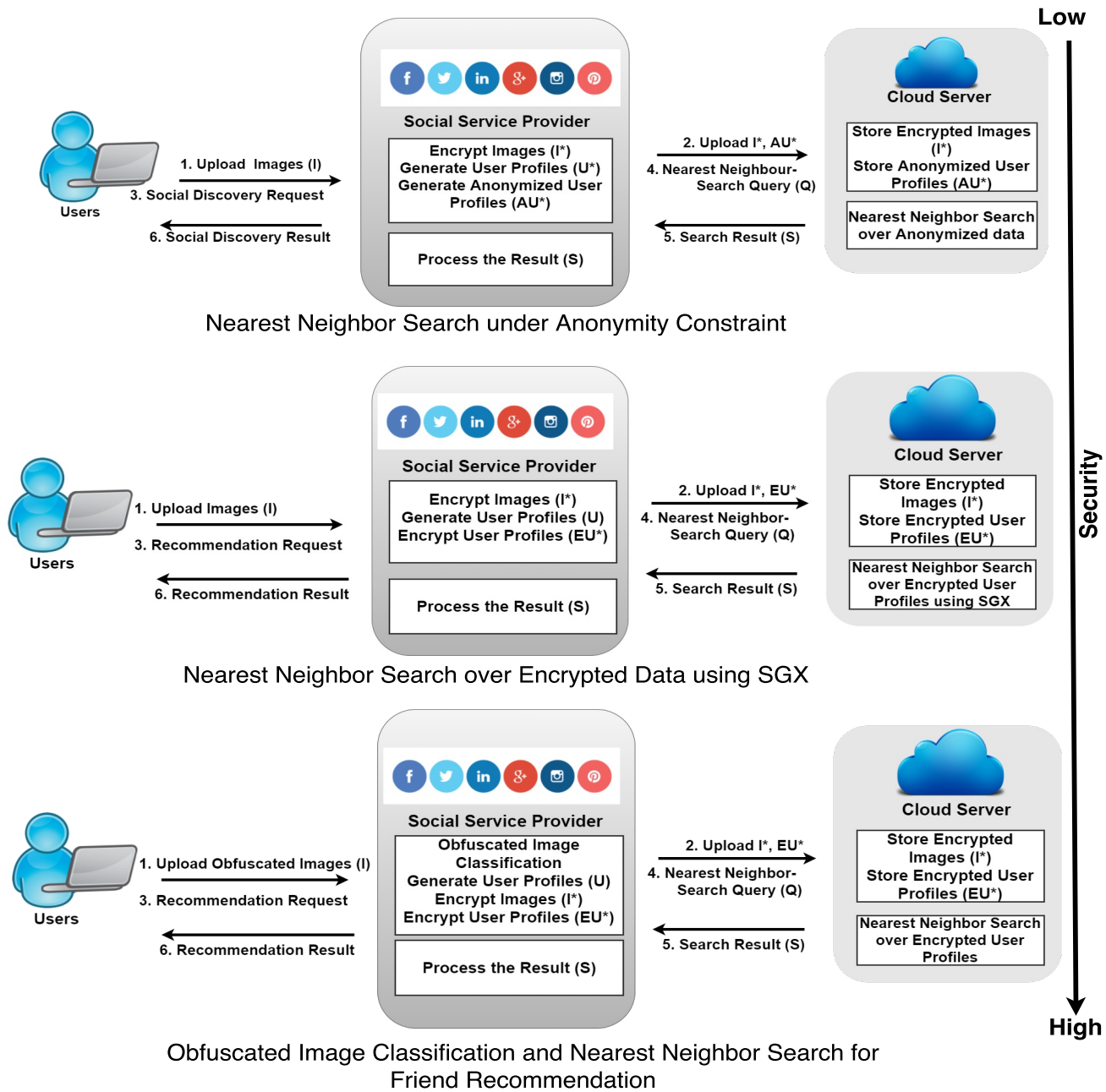


Figure 1.1: Research Problems Addressed in this Thesis

tion technique. The proposed anonymization scheme considers both high dimensionality of data and privacy implications. The anonymized data is then shared with the cloud and search operation is performed for social discovery over the

anonymized data utilizing the efficiency of Locality Sensitive Hashing. To the best of our knowledge our work is the first to deal with nearest neighbour search over anonymized data. The results of this chapter are published in the IEEE International Conference on Cloud Engineering (IC2E 2017) [6].

Nearest Neighbour Search over Encrypted Data using SGX

The attractive features of cloud platforms such as low storage cost and high availability are encouraging social networking sites to outsource their client data to the cloud. However, after reports of citizens surveillance by government agencies the privacy of the outsourced data is a rising concern. Although encryption protects the data from unauthorized access, it increases the computational complexity of executing arbitrary functions (e.g., nearest neighbour search) on the data. Motivated by this scenario, in this chapter, we propose an efficient scheme for secure nearest neighbour search over encrypted high-dimensional data.

The proposed design utilizes the advantages of the Intel® Software Guard Extensions (Intel® SGX) architecture, the Convolutional Neural Network (CNN), and Locality Sensitive Hashing (LSH) for secure similarity computation and the nearest neighbour search. The Intel® SGX achieves both security and efficiency during the nearest neighbour search. To the best of our knowledge, this is the first attempt to instrument SGX to securely search for nearest neighbours in a cloud architecture. The proposed framework is more than two times faster than the secure naive nearest search approach. The results of this chapter are submitted in the Elsevier Future Generation Computer Systems Journal [10].

Obfuscated Image Classification and Nearest Neighbour Search for Friend Recommendation

In the previous two proposed models, we consider primary attacks from the third party cloud server. The social service provider and all users in the system are assumed to be trustworthy. The assumption is justifiable because deviating from the protocol and performing illegal operations on user's shared data will lead to negative user experience as well as potential revenue loss of the service provider [11]. However, the service providers may perform some operations for better user experience and to increase the utility of service. This may lead to unintentional privacy breach and users may feel insecure to share the actual data rich in contents.

Obfuscating images before sharing can be considered as a potential solution; however, de-obfuscating and then classifying millions of images at the service provider's end does not provide privacy guarantee, and it is computationally expensive. Motivated by this scenario, in this chapter, we propose a image-centric friend recommendation framework which protects the privacy of images through obfuscation, classifies obfuscated images using the deep neural network to build user profiles and adopt one of the efficient nearest neighbour search scheme to generate friend recommendation. The proposed model provides better security guarantee as obfuscated version of images are shared for getting recommendation service. To the best-of our knowledge, our work is the first to deal with obfuscated image classification and nearest neighbour search for providing image-centric friend recommendation service. The results of this chapter are published in the Elsevier Sustainable Cities and Society Journal [12].

1.3 Organization

This thesis is organized as follows:

- Chapter 2 shows how to utilize the anonymization technique to perform secure nearest neighbour search and provide social discovery.
- Chapter 3 discusses a how to utilize the Intel SGX architecture to perform secure nearest neighbour search over encrypted data in the cloud.
- Chapter 4 discusses how to utilize nearest neighbour search functionality in image-centric friend recommendation based on obfuscated image classification.
- Chapter 5 concludes the thesis.

Chapter 2

Nearest Neighbour Search under Anonymity Constraint

2.1 Introduction

One of the most common activities on social networking sites is to meet people with similar interests. Different parameters play significant roles in defining the similarity between users in social networking sites. Images can be considered as one of the most dominating parameters as people mostly upload images to represent the things they are interested in and sometimes to share the fragments of their life experiences [5]. To protect the semantic richness of image content, it is necessary to store the images uploaded by the users virtually. This situation presents the problem of the requirement of a huge amount of storage spaces. Storing visual data directly into the cloud can be considered as a solution for this storage constraint problem. However, Incorporating cloud storage for storing user's uploaded images possesses severe security threat as those images may contain much sen-

sitive information related to a particular user. A user can be uniquely re-identified using the information gained through his shared visual contents.

Generally, in recommendation systems, a user profile is built for every user capturing the user preferences. When a user request for recommendation to the service providers, a nearest neighbour search operation is performed based on his/her user profile and results are recommended by the service providers. Apart from the sensitivity of images shared by the users, user profiles are also sensitive.

To minimize the disclosure risk, the data should be anonymized or encrypted. If we choose the data to be encrypted, we need to use a cryptographic scheme, which allows search on encrypted data, such as *Homomorphic Encryption*, *Order Preserving Encryption* or another state of the art cryptographic protocols. But we have avoided the use of such methods in our proposed approach, as these schemes are computationally expensive for wide adoption [13].

Indeed, ensuring the privacy of user's data is difficult. A widely used privacy metric that quantifies the disclosure risk of a given data instance is *k-anonymity* [14]. It requires that for any data instance in a dataset, there are at least $k-1$ other distinct data sharing the same attribute ensuring that unwanted personal information cannot be disclosed merely through the released attributes. However, for high dimensional data, anonymization is hard to achieve even for smaller k -value [15; 16]. As the dimensionality of data increases, there is much information loss during the process of k -anonymization which limits the utility of data to be used for recommendation.

There are many existing works related to privacy preserving data mining and data release. So far, not much is found that deals with the problem of finding the nearest neighbours in anonymized dataset with utility preservation. Ensuring

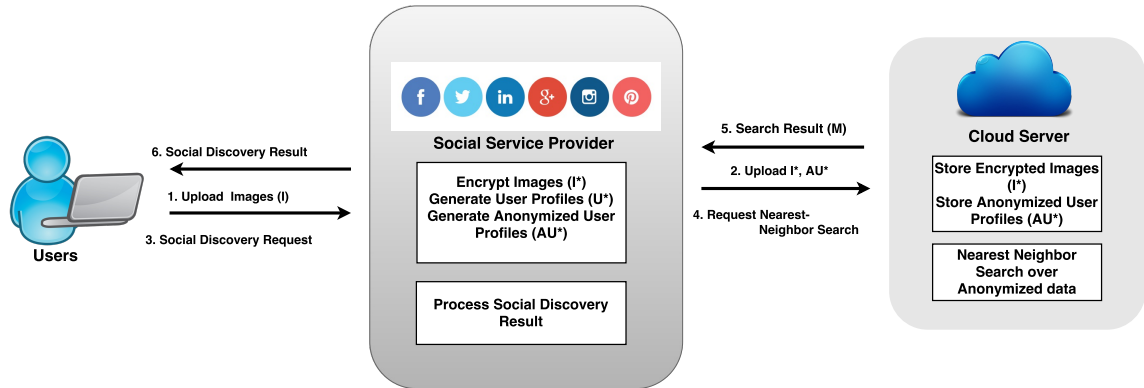


Figure 2.1: Proposed architecture of the system

a right balance between user's privacy and the utility of high-dimensional user profiles data is the primary focus of this chapter. To the best of our knowledge, our work is the first to deal with nearest neighbour search over high dimensional anonymized data.

2.2 System Overview

Analyzing the properties of the recommendation architectures existing in the social networking applications, the matter of concern is how the social service providers store or use the shared sensitive information.

2.2.1 Architecture and Entities

Fig. 2.1 presents an overview of our proposed system. It has three main entities: *users*, *social networking service providers*, and *cloud server*. The role of each entity is illustrated below:

- *Users*: Users are the people who uses the social networking applications.
- *Social Networking Service Providers*: This set consists of the entities who of-

fers social networking services to the users. Facebook, Flickr, Google Plus, Instagram, etc. falls into this category. In our model, they also facilitate the communication between *users* and *third party cloud server*. In the rest of the chapter, we will use the term *service provider* to denote the entity *social networking service provider*.

- *Cloud Server*: Social media sites outsource their large volume of images to a third party cloud server i.e. Amazon cloud. This third party server is semi-honest.

2.2.2 System Service Flow

From Fig. 2.1, it is clear that the information flow in the proposed framework occurs in two stages: *first*, while storing user's uploaded encrypted images and anonymized image profiles in the third party cloud server and *second*, during the processing of similarity search operation on the anonymized image profiles for social discovery requests.

Storing Data in the Cloud Server

The steps involved in storing data in the *cloud server* can be enumerated as —

- When the *users* upload their images using different social networking sites, the images are received by the *service provider*.
- After receiving images from the *users*, the *service provider* first extract the visual features from the images using the convolutional neural network and encrypt those images. The *service provider* then generates a binary valued image profile from these extracted features for every user who have uploaded

those images. Next, the *service provider* generates the user profiles using the optimum feature selection scheme operated over binary attributes of user profiles and anonymizes it.

- After the generation and anonymization of user profiles, the *service provider* stores the encrypted images and the anonymized image profiles in the *cloud server*.

Social Discovery Request

When a social discovery request is generated by a user, the following steps are required —

- The request is issued by the *users*. The *users* send this request to the *service providers*.
- The *service provider* forwards the request to the *cloud server*.
- Upon receiving the social discovery request, the *cloud server* performs a nearest neighbour search operation on the anonymized image profiles.
- The *cloud server* sends the search result to the *service provider*.
- The *service provider* then uses the search result to recommend the similar users, groups or events to the *users* who initiated the social discovery request.

2.2.3 Threat Model

In our proposed model, we consider the primary attack on the third party cloud server which is assumed, to be honest, but curious about learning the contents of user's shared images and user's interests. We focus on preserving the privacy of

those shared images and user profiles outsourced in the cloud and provide secure social discovery. The service provider and all the user's in the system are assumed to be trustworthy. We do not consider other possible attacks, such as malicious user attack at this time.

2.3 Problem Formulation

Suppose, a *social network service provider* (e.g. flicker) wants to outsource users' profile data and shared images to the *cloud server* and provide social discovery service to the interested *users*. The objective of this chapter is to enable *cloud server* to perform social discovery operation in a privacy-preserving manner.

2.3.1 Image Data

For every user, the *service provider* builds a binary $user \times feature$ matrix using the convolutional neural network. Suppose a user has uploaded 50 images. Then the image profile of that user consists of 50 images. We term the image profile of the target user who wants the recommendation as query image profile. Figure-2.2a shows a sample query image profile containing a subset of 5 images of actual image profile.

Convolutional Neural Network. A Convolutional Neural Network (CNN) is the combination of one or more convolutional layers and then subsequently followed by some fully connected layers in a multilayer neural network.

Table-2.1 shows a transformation of user image profiles presented in Figure-2.2a into binary $user \times feature$ matrix. The service provider extracts visual features (e.g. dog, flowers, portraits) from the images uploaded by the *users* using CNN

and then generates feature vectors. Then all the feature vectors are combined and normalized to generate user image profile. As an example, if we visualize a sample user profile, $u = s_1, s_2, \dots, s_n$; then a 1 in position s_i denotes that user has preference on the i^{th} visual class (e.g. dog, flowers, birds) and n is the total number of class active in the output layers of convolutional neural network.

2.3.2 Privacy Requirement

In the online social network, the variation and massive size of user-shared data and interactions have raised new concerns about recommender systems, among which privacy preservation of user-preferences is a challenging task [17]. Anonymization based data release make it computationally difficult for the adversary to re-identify a person who is in the dataset. Although various definitions of privacy exist in practice and literature, we have used a well-known privacy metric k -anonymity.

Data Anonymization. Data anonymization is a process of transferring the data into a new form to produce anonymous data and prevent information leakage. Although during the anonymization process original data is modified, the data is still useful for processing and mining useful knowledge.

Definition 2.3.1. (*k*-Anonymity). *A dataset D satisfies k -anonymity if for any entity in D , there exist at least $k - 1$ other entities having the exactly similar indistinguishable attributes.*

In our approach, we define a database $D(U, F)$ as a binary relation between a set of users (U) and a set of visual object features (F); thus $D \subseteq U \times F$, where $U = \{u_1, u_2, \dots, u_n\}$ and $F = \{f_1, f_2, \dots, f_n\}$; n and c are respectively the number of users and the size of the visual objects feature sets.

It is less expected that the dataset is k -anonymous by default. For example,

if we choose $k = 2$, the sample dataset is not k -anonymous. In fact, the research problem here is the selection of optimum feature set to transform data set D into an anonymous version D' preserving utility.

Definition 2.3.2. (*k-Anonymity based on Feature Selection*). In a dataset D and for a given positive integer k , a user entity $u \in D$ satisfies k -anonymity based on feature selection if there exist at least $k - 1$ other entities in D having the same feature set and indistinguishable from u . We use the term $AFS(u)$ to denote the largest k for which the entity u is k -anonymous by feature selection.

Definition 2.3.3. (*k-Anonymous based on Feature Selection Dataset*). A dataset is k -anonymous, if every user entity $u \in D$ satisfy k -anonymity based on feature selection. We use the term $AFS(D)$ to denote the largest k for which the dataset D is k -anonymous by feature selection.

Example: For the sample dataset D in Table-2.1, if we select the feature set $F = \{f_1, f_2, f_3\}$; then the user entity u_1 satisfy 3-anonymity based on feature selection. But considering this feature set for entity u_3 , the feature attributes are unique, it does not satisfy any anonymity for the selected feature set. Considering the feature set more tactically, if we choose, $F = \{f_1, f_2, f_4\}$, we can see that all the user entities satisfy 3-anonymity. In this case, the data set D is 3-anonymous by optimal feature selection.

2.3.3 Utility Requirement

Indeed, we have to choose the optimum feature set in such a way that for some given positive integer k , k -anonymity is satisfied for all the entities in the dataset and the data utility is also preserved for nearest neighbour search.

Table 2.1: Sample dataset with binary feature attributes

U_{ser}	Dog (f_1)	Cat (f_2)	Sky (f_3)	Portrait (f_4)	Flower (f_5)
u_1	0	1	0	1	1
u_2	0	1	0	1	1
u_3	1	0	0	1	1
u_4	1	0	1	1	1
u_5	0	1	0	1	0
u_6	1	0	1	1	1

2.4 Proposed Approach

The approach begins when a user uploading images to the social networking sites to find a match with other users who have common visual interests. After uploading preferred images, the *users* may request for a social discovery to the social network service provider.

2.4.1 Preprocessing Steps at the Service Provider End

To enable the *cloud server* to perform privacy-preserving social discovery in the outsourced environment, there are some preprocessing steps taken by the *service provider*. After generation of binary valued user image profiles, the profiles are anonymized using the maximal optimum feature selection technique.

Optimum Feature Set Generation. Let's represent the collection of optimum feature sets which satisfy the k -anonymity for all the dataset entities as F_k . While working with large datasets the optimum feature sets size may be huge. One of the possible solutions might be increasing the value of anonymization factor (k) to

control the size. But increasing the value of k can negatively impact the utility of recommendation.

However, brute-force approach can be very slow and take a long time to go through all the features. To make the system practical, one alternative is considering maximal optimum feature set M_k instead of generating F_k and look for the best feature subset in M_k . A subset of features $F \in F_k$ is maximal if it has no supersets of features which is optimum.

The idea of considering maximal feature set comes from the assumption that with more features the data utility is more preserved reducing the computational cost. Just like the frequent item set mining, maximal frequent itemset mining algorithm can be used for finding M_k . We have used the LCM-miner package designed by Uno et al. [18] which is at present the fastest method for finding maximal frequent item set mining.

Utility Function. Considering only the maximal feature sets has some limitations as ties are very common with many feature sets of the same size and it does not consider the utility preservation of recommendation data to search the nearest neighbour later on. We have to look for some utility measure while choosing the maximal optimum feature set.

In our work we have used F_1 -score measure based on the precision and recall value as the utility function. The recall value is calculated depending on the number of relevant records retrieved during top- k nearest neighbour search after anonymization to the total number of relevant records in the original dataset. Precision is measured based on the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved after anonymization.

After that F_1 -score is calculated based on precision and recall as utility mea-

Algorithm 1: Optimum Feature Set Generation

Input: D, k **Output:** f'

- 1: Calculate the maximal optimum feature set M_k containing the feature set that satisfy k -anonymity based on feature selection for the given positive integer k
 - 2: Find the best optimum feature set f' , where $f' \in M_k$ based on utility measurement using F_1 score criteria
 - 3: **return** f'
-

surement of feature sets.

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

The feature set with highest F_1 -score, f' is selected for anonymization of data which ensures better utility preservation for performing nearest neighbour search operation. The steps regarding optimum feature set generation with utility preservation is summarized in Algorithm 4.

2.4.2 Processing Steps at Cloud Server End

To provide privacy-preserving image centric social discovery there are some processing steps executed by the cloud server. The similarity search query is executed over anonymized data. The similarity search over anonymized data is performed utilizing fast nearest neighbour search in high dimensional spaces with the combination of *Locality Sensitive Hashing* (LSH) and *Jaccard Similarity*.

Combining LSH with Jaccard Similarity. During the similarity comparison, objects are classified by a set of relevant attributes and are represented as vectors or points in a high dimensional space. If we are given a collection of vectors, similarity can be measured using different approaches measuring the distance

or closeness among the vectors. For example, using *Euclidean distance measure*, *Jaccard Similarity*, *Cosine similarity* etc. Performing pairwise comparisons in a set is time-consuming because the number of comparisons grows geometrically with the size of the set. Most of those comparisons, furthermore, are unnecessary because they do not result in matches. The combination of minhash and LSH seeks to solve these problems.

Minhashing is the process of converting large sets to short signatures while preserving the similarity. They make it possible to compute possible matches only once for each element so that the cost of computation grows linearly rather than exponentially. Random permutations are necessary for computing minhash signatures. However, random permutations can be replaced by random hash functions mapping a number of rows to the equal number of buckets [19]. We can assume that the corresponding hash function permutes the corresponding row. We denote it by $h(r)$. Here, $sig_n(i, c)$ is termed as the signature for the i^{th} hash function and column c . At first, we set $sig_n(i, c)$ to ∞ for all i and corresponding column c . The column j of row i is denoted as $c_{i,j}$. Minhash signatures for each user entity u is computed following the steps of Algorithm-5 which is motivated by the algorithm proposed in [19].

Suppose, the each column in Table-2.2 represents user profiles and we have two hash functions defined as: $h_1(x) = x + 1 \pmod{5}$ and $h_2(x) = 3x + 1 \pmod{5}$. If we apply the hashing in the order of the row number we get the value shown in the last two columns of Table-2.2.

Now, we can run through the steps of the algorithm for computing the signature matrix. At the beginning, the signature matrix consists of all ∞ as shown in Table-2.3.

Algorithm 2: Computing Minhash Signatures**Input:** User profile matrix**Output:** Minhash Signatures

- 1: Calculate hash of $h_1(r)$ to $h_n(r)$.
- 2: **for each** column c **do**
- 3: **if** $c_{i,j} = 1$ **then**
- 4: **for** $i = 1$ to n **do**
- 5: Compare $sig_n(i, c)$, $h_i(r)$ and Update $sig_n(i, c)$ to the smaller value among them.
- 6: **end for**
- 7: **end if**
- 8: **end for**
- 9: **return** $sig_n(i, c)$

Table 2.2: Hash functions computed for the user profile matrix

Row	u_1	u_2	u_3	u_4	u_5	$x + 1 \bmod 5$	$3x + 1 \bmod 5$
0	0	0	1	1	0	1	1
1	1	1	0	0	1	2	4
2	0	0	0	1	0	3	2
3	1	1	1	1	1	4	0
4	0	0	0	0	0	0	3

At first, we operate on row marked as 0 in Table-2.2. It appears that the values of $h_1(0)$ and $h_2(0)$ are 1. The row 0 has 1's in the columns for user profiles u_3 and u_4 . According to the algorithm, these columns of the signature matrix can be updated. As 1 is less than ∞ , we update both values in the columns for u_3 and u_4 .

Table 2.3: Computing minhash signatures (Step-1)

	u_1	u_2	u_3	u_4	u_5
h_1	∞	∞	∞	∞	∞
h_2	∞	∞	∞	∞	∞

The present state of the signature matrix is shown in Table-2.4.

Table 2.4: Computing minhash signatures (Step-2)

	u_1	u_2	u_3	u_4	u_5
h_1	∞	∞	1	1	∞
h_2	∞	∞	1	1	∞

Next, we operate on row marked as 1 in Table-2.2. This row contains 1 in u_1 , u_2 , u_5 , and $h_1(1) = 2$ and $h_2(1) = 4$. The new signature matrix is shown in Table-2.5.

Table 2.5: Computing minhash signatures (Step-3)

	u_1	u_2	u_3	u_4	u_5
h_1	2	2	1	1	2
h_2	4	4	1	1	4

Table 2.6: Computing minhash signatures (Final Step)

	u_1	u_2	u_3	u_4	u_5
h_1	2	2	1	1	2
h_2	0	0	0	0	0

If we continue in this way, finally we get the signature matrix as Table-2.6. It is clear from this table that signature matrix of u_1 , u_2 and u_5 are identical. The



signature matrix of u_3 and u_4 are also identical. So, their original user profiles should be similar. We can measure the similarity using the *Jaccard Similarity*.

Definition 2.4.1. (*Jaccard Similarity*). *The Jaccard similarity between two user u_1 and u_2 can be defined as: $|u_1 \cap u_2|/|u_1 \cup u_2|$, that is the ratio of the size of the intersection of u_1 and u_2 to the size of their union.*


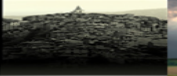
















The *Jaccard Similarity* between user profile u_1 and u_2 is 1 and the *Jaccard Similarity* between their signature matrix is also 1. So, they are similar. The *Jaccard Similarity* between user profile u_1 and u_3 is $\frac{1}{3}$. Eventually, they are not similar and the similarity of their signature matrix is also $\frac{1}{3}$. Indeed, we can reduce the dimension of the user profile using minhashing technique and then measure similarity.

But computing minhash signatures and the linear comparison is time-consuming for big datasets. So, we combine LSH with minhashing. LSH is an efficient algorithm to perform fast nearest neighbour search in high dimensional spaces [20]. One of the effective processing steps of LSH is to hash items several times technically so that similar items are hashed to the same bucket than dissimilar items [19].

The query retrieves the similarity match from the *cloud server* and returns the matching results to the *service provider*. After receiving the matched results, the service provider process it and rank according to the distance of target user query image profile. The ranking accuracy can be more utilized by integrating with other efficient social discovery approaches combining different features to make the social discovery framework more acceptable and accurate. For example, one can combine our proposed approach with [21] to further improve ranking result and accuracy.

User	Preferred Images				
Query					
Top1					
Top2					
Top3					
Top4					
Top5					

(a) Before anonymization

User	Preferred Images				
Query					
Top1					
Top2					
Top3					
Top4					
Top5					

(b) After anonymization

Figure 2.2: Evaluation of image profiles consisting of 50 images using the proposed approach

2.5 Experiment and Result

2.5.1 Implementation Detail

The implementation of the prototype of the framework as described in Fig. 2.1 was done in two phases. In the first phase, we used the convolutional neural network for extracting features from the images. For this purpose, we used a PC with Intel core i5-4590 CPU and 8GB RAM. For training the network we used a MATLAB toolbox called MatConvNet [22]. MatConvNet toolbox implements *CNN* and this network is very effective for extracting visual features from the images. We used a pre-trained model called imagenet available in the toolbox. After training our model, we extracted the image features using the trained model.

In the second phase, we used the extracted features to simulate the interaction between the service front end and the *cloud server*. We used the extracted features to generate binary valued user image profile vectors. Then we anonymized those image profile vectors by applying maximal optimum feature selection technique using LCM-miner package [23]. For the implementation of LSH with *Jaccard Similarity*, we used a Java library called java-LSH [24]. This library efficiently implements Locality Sensitive Hashing (LSH) with minhashing. For the extraction of the features from the images and the generation of image profiles we used MirFlickr-25k dataset [25].

2.5.2 Image Profile Evaluation

We simulated image profile evaluation for social discovery operation using our model in two ways.

First, we created 2000 users and then randomly assigned 50 images to each

of the users. Then we generated binary-valued image profile for each user. Our query image profile also consisted of 50 random images. Then for the query image profile, we found out the nearest top 5 user image profiles. The result is shown in Figure-2.2a. In this figure, the query image profile represents the subsets of visual object preferred by the target user. It is clear from the figure that all the nearest neighbour image profile's contents are similar visual objects as the query image profile.

Second, we run the same experiment on the anonymized user profiles. The user profiles are anonymized using the proposed scheme of optimum feature set selection. Then in the same way we search for top 5 nearest image profiles for the query image profile. The result is shown in Figure-2.2b. The experimental results show that the user image profiles which are mostly similar with the query image profile are eventually recommended. This clearly demonstrates that data utility is preserved after anonymization.

2.5.3 Accuracy Analysis

We trained a CNN to classify the 25K images in the MirFlickr-25K dataset into the 17 different classes based on most popular image tags on Flickr. On the test data, we achieved top-1 to top-5 error rates on the range of approximately 11% which is considerably better than the state-of-the-art of bag-of-words model.

We compare the accuracy of our model for finding the similar image profiles. First, we created 2000 users and then randomly assigned 50 images to each of the users. Then we generated binary-valued image profile for each user. Our query image profile also consisted of 50 random images. It is more practical to construct image profile comprised of 50 images rather than only 5 images as done by Yuan

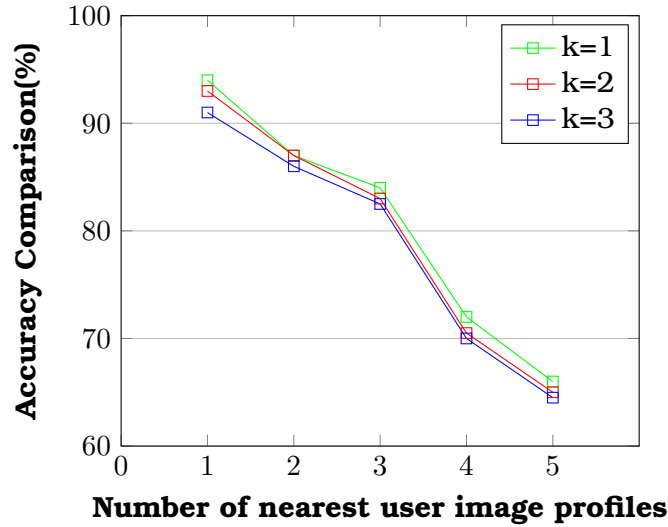


Figure 2.3: Accuracy comparison varying anonymization factor(k)

et al. [4] because only 5 images fail to capture user preferences relevantly.

The accuracy of finding similar image profile is shown in Figure-2.3. The accuracy analysis and the image profile evaluation in Figure-2.2b confirms that the model recommends the top similar user correctly after anonymization also. So, the data utility is preserved after anonymization. We have run our experiment varying the anonymization factor k up to 3 with the maximal feature set of size 12. More anonymization requires choosing of small feature sets on small dataset significantly reduce the utility of data which validates the privacy-utility trade-off. In both cases, the accuracy is measured by taking the average value of independent five experiments.

2.6 Related Work

There are many different privacy preserving techniques to obscure information that can uniquely identify individuals in released dataset during social discovery.

These methods can be broadly divided as cryptographic and anonymization based approaches. The related works are summarized as follows.

2.6.1 Cryptographic Approach

There are some existing works on social discovery which solely depends on content-based user profile matching [26], [27] ensuring the security of user contents from the unauthorized attacker. Their works are mostly based on private set intersection cardinality [28]. There are some other works considering user profiles as vectors [29]. Two users are considered as similar when their secure scalar product is under certain defined threshold. Moreover, we can relate privacy-preserving social discovery to the existing works on search over encrypted data in the cloud [7], [30] and secure ranked search over high dimensional encrypted data [31]. Yuan et al. [4] proposed a privacy-preserving image-centric social discovery system where the images shared by the users were encrypted before storing them in the public cloud.

2.6.2 Anonymization Based Approach

Unfortunately, current cryptographic solutions are computationally expensive and impractical for a large number of users [32]. The alternative possible solutions are using differential privacy or anonymity based approaches. Differential privacy adds noise for privacy preservation which may result in huge utility loss of recommendation data [33]. On the other side k -anonymity based approaches are simple and flexible to apply for any real life privacy preserving needs, but it may reduce the utility of data due to inefficient anonymization scheme.

Only a few work exists which considers anonymization based privacy model

with data utility. Hence, most of them work fine for relational datasets. An anonymization based collaborative filtering approach is designed by Richard et al. [32] to recommend movies for Netflix users applying *locality sensitive hashing* and adding fake ratings to user profiles but their approach does not preserve the utility of high dimensional data. The feature selection techniques to improve the utility of privacy preserving data publishing has been proposed in [34], [35]. However, they used feature selection as an add-on tool to anonymize data without considering privacy during feature selection process and their approach is more suitable for preserving the utility of data during classification.

Chapter 3

Nearest Neighbour Search over Encrypted Data using SGX

3.1 Introduction

Nearest neighbour Search (NNS) is an optimization problem for finding closest or most similar points with respect to a given query point. Social discovery based on NNS is one of the most prevailing services facilitated by different social networking sites such as Facebook, Twitter, Linked-In, Google+, etc. On top of that, the availability of a large number of images gives the opportunity to the social networking service providers to generate recommendations using the visual features of the user uploaded images.

3.1.1 Challenges

However, the task of measuring similarity among millions of records in real time is a computationally expensive task. Although anonymization solves the problem of

NNS in a reasonable time, there is utility loss during anonymization which affects NNS performance [6]. On the other hand, the complexity of searching increases significantly when the underlying data is encrypted. If data is encrypted, we need to adopt the cryptographic schemes which provide a strong security guarantee such as the homomorphic cryptosystems and garbled circuits. Unfortunately, fully homomorphic encryption based techniques are computationally expensive for real-life applications. While the use of garbled circuits requires huge interaction overhead which is inefficient for supporting a huge number of queries in search applications [9]. These operations alone are not efficient for performing an NNS operation on an encrypted large dataset.

3.1.2 Contributions

- We exploit the efficiency of Convolutional Neural Network (CNN) [38] and the computation architecture of Intel® SGX [39]. Because of the resemblance with the human brain, CNN can classify objects more accurately than the existing models (e.g., the bag of words and feature based methods) and helps service providers to build relevant user profiles. On top of that, Intel® SGX achieves both security and efficiency during the NNS. To the best of our knowledge, this is the first attempt to instrument SGX to securely search for nearest neighbours in a cloud architecture.
- We optimize the similarity search operation by combining Locality Sensitive Hashing (LSH) [20] along with the MinHashing technique [19]. As user profiles, generated by CNN, are high-dimensional and contain different attributes, we depend on the MinHashing technique to group similar users in one bucket. We depend on LSH to execute the MinHashing technique more

Table 3.1: Comparisons among previous schemes for the nearest neighbour search problem. Our proposal is the only technique that enables cloud to efficiently compute the final result without any post-processing by a trusted entity.

Existing Works	Year	Data Size (n×m)	Time (sec)	Security Primitive	Cloud Computes Final Results	Faster than Linear
Kuzu et al. [7]	2012	$3e10^3 \times 3e10^3$	1.40	Paillier	✗	✓
Yao et al. [36]	2013	$1.2e10^6 \times 30$	100	OPE	✗	✓
Elmehdwi et al. [8]	2014	$1e10^4 \times 18$	600	Paillier	✓	✗
Yuan et al. [4]	2014	$1e10^6 \times 64$	2.43	Paillier	✗	✓
Samanthula et al.[37]	2015	$1.72e10^3 \times 6$	721.2	Paillier	✓	✗
Wang et al. [9]	2016	$2.6e10^5 \times 1$	344	OPE	✗	✓
Our work	2017	$5e10^4 \times 17$	0.297	SGX	✓	✓

than one time (stages) to increase the probability that similar users will be grouped in one bucket. Assigning each user with similar other users in one bucket optimizes the similarity search operation.

- We provide a comprehensive security and accuracy analysis of the proposed approach. The similarity is measured using four different metrics: cosine similarity, jaccard similarity, hamming and euclidean distance. We experimentally show that the proposed approach is more than two times faster than the naive approach (presented in Algorithm 3). Actually, the proposed approach finds nearest neighbours in less than 300 milliseconds for 50k users. It is only 5 times slower than computing nearest neighbours in plain-text

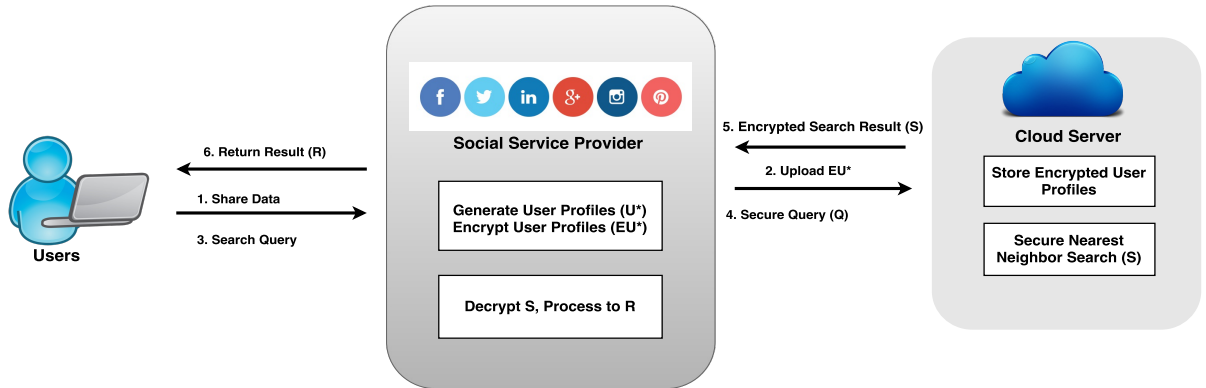


Figure 3.1: Architecture

which is quite reasonable with just one round of interaction with the server.

- We design an approach which performs faster than linear to find the nearest neighbours in milliseconds and to compute the final results in the cloud without further post-processing steps. The post-processing steps include actual similarity measurement or ranking the results at the service provider's end. From Table 3.1, it can be noted that other efficient approaches do not generate final results at the cloud server's end. As a result, they need further processing steps at the service provider's end.

3.2 System Overview

To enable social discovery while preserving the personal profile information, nearest neighbours should be computed in a privacy-preserving manner. In this section, we will detail the architecture, the service flow, and the adversary model of the proposed framework.

3.2.1 Architecture

The architecture of the proposed framework is shown in Figure 3.1. It consists of three main entities: users, a service provider, and a third-party cloud server. Users are individuals who use social networking applications. The service provider is the party that offers services to users (e.g., Facebook and Flickr). In our model, it facilitates the communication between users and the third-party cloud server. The service provider outsources its large volume of data to a third party cloud server (e.g., Amazon cloud).

3.2.2 System Service Flow

The information flow in the proposed framework, as shown in Figure 3.1 begins when the users uploading their images to the social service provider. After receiving the images, the social service provider executes the following steps:

- It builds a user profile (U^*) using CNN.
- It constructs the search index for each user using LSH and the MinHashing technique.
- It encrypts all the resulted user profiles (EU^*) and search indices and uploads them to the cloud.

When a social discovery request is generated by a specific user and sent to the social service provider, the service provider encrypts the request in the form of a query and forwards the query to the third party cloud server. After receiving the encrypted query, the cloud server executes the following steps:

- It performs a NNS operation on the stored encrypted profiles. This search operation is performed via the Intel® SGX.
- It returns the encrypted search result (S) to the social service provider.

Upon receiving the search result, the service provider decrypts it and provide social discovery service to the user (R). To make the system more practical and user-friendly, the encryption is done by the trusted service provider rather than by the user. If the encryption is done by the user, this requires much computation overhead and bandwidth [40].

3.2.3 Assumptions and Adversary Model

In our proposed model, we assume that an adversary is unable to manipulate the protected memory of the Intel® processor that resides in the cloud service provider's data centre. It is a widely accepted assumption based on the security guarantee provided by Intel® utilized by many recent frameworks [41; 42; 43; 44]. We also consider the cloud server as a semi-honest (also known as honest but curious) entity. A semi-honest party follows the protocol but it might be curious about learning the contents of the user's shared data and user's interests [45]. We focus on the security of the user's shared data outsourced to the cloud and enable a secure social discovery service. The social service provider and all users in the system are assumed to be trustworthy. We are aware of some recent side channel and hardware attacks against secure hardware-based techniques [46]. In the context of the semi-honest cloud server, such side channels are not applicable, and hence, we do not address these issues in this article.

3.3 Background

In this section, we present an overview of the building blocks that are utilized in the proposed solution.

3.3.1 Intel SGX Architecture

SGX is a set of x86-64 ISA extensions that enables execution of code in protected environment termed as enclave without trusting anything other than the processor and the code placed inside the enclave [39]. The main features of SGX securing the computation are highlighted below.

Memory Protection

An enclave can be visualized as a protected container that is located in the application's address space. The processor protects the enclave and provides permission to access the enclave memory only when it is necessary. Any instruction tries in vain to read or write the memory of a running enclave [47]. Furthermore, SGX architecture ensures privacy and integrity of the enclave pages. The cache-resident and enclave data are protected by CPU access controls. The integrity of data is protected by encryption while writing into memory.

Enclave Access

In addition to the memory protection and integrity, SGX also monitors the transfer of function calls inside and outside of the enclave. In this way, SGX protects the enclave's register file from operating systems' exception handlers [48]. The functions written inside the enclave can be called by an untrusted application

through a process transferring the call to a user-defined entry point inside the enclave [42] only. There are some exceptions which may interrupt the enclave execution such as system calls. In this situation, the processor saves the on-going state of the register to the enclave memory and resume execution outside of the enclave.

Sealing and Attestation

SGX also supports sealed storage to protect the integrity of data and attestation to prove to a remote computer that it is communicating with a specific secure container hosted by a trusted platform [39]. During the initial stage of creating an enclave, a secure hash termed as a measurement is established which may be used for identity verification to the processor.

3.3.2 Nearest Neighbour Search

Nearest Neighbor Search (NNS) is an optimization problem for finding closest or most similar points with respect to a given query point (q), which can be high dimensional containing more than one attributes. Algorithm 3 shows the steps of the naive nearest neighbour search.

Definition 3.3.1. *Given a dataset of u_1, u_2, \dots, u_n users having m_1, m_2, \dots, m_d attributes (total d attributes) and query attributes $q = \{q_1, q_2, \dots, q_d\}$, NNS will output the most similar users according to the query attributes and a specific similarity metric.*

Proposition 3.3.1. (Complexity) *The computation cost of Algorithm 3 is bounded by $O(nd)$ where n is the number of users and d is the number of attributes.*

Algorithm 3: Naive Nearest Neighbour Search**Input:** d query attributes (q), dataset of u_1, u_2, \dots, u_n users**Output:** search results (R)

- 1: **for** $i = 1$ to n **do**
- 2: distance=ComputeDistance($u_i[1, 2, \dots, d]$, $q[1, 2, \dots, d]$);
- 3: $R = \text{RankResult}(u_i, \text{distance})$; rank results in ascending order according to the distance from the query and store them in R
- 4: **end for**
- 5: **return** R

In this work, the similarity among the user profiles is computed inside the enclave of the SGX in a secured manner. If we consider two user profile vectors $A = \{0, 1, 0, 1, 1\}$ and $B = \{1, 0, 0, 1, 1\}$ and we consider $d = 5$ as the total size of the feature set, then the similarity between the two user profiles (A, B) can be measured using different metrics. We considered four different similarity metrics described below:

Cosine Similarity. Cosine similarity is a measure of similarity between two non zero vectors regarding of inner product space. The cosine similarity between two vectors A and B can be calculated as:

$$\begin{aligned}
 \text{CosineSimilarity}(A, B) &= \frac{\sum_{i=1}^d A_i * B_i}{\sqrt{\sum_{i=1}^d A_i^2 * \sum_{i=1}^d B_i^2}} \\
 &= \frac{(0 * 1 + 1 * 0 + 0 * 0 + 1 * 1 + 1 * 1)}{\sqrt{(0^2 + 1^2 + 0^2 + 1^2 + 1^2) * (1^2 + 0^2 + 0^2 + 1^2 + 1^2)}} \\
 &= \frac{2}{\sqrt{3 * 3}} = \frac{2}{3}
 \end{aligned}$$

For cosine similarity we can say that the similarity score between A and B is $\frac{2}{3}$.

The maximum cosine similarity score between two vectors can be 1.

Jaccard Similarity. The Jaccard similarity between two sets can be defined as the ratio of the size of their intersection divided by the size of their union. The Jaccard similarity between two vectors A and B can be calculated as,

$$JaccardSimilarity(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{2}{4}$$

For jaccard similarity, the similarity score between A and B is $\frac{2}{4}$. The maximum jaccard similarity score between two vectors can be 1.

Hamming Distance. Hamming distance between two vectors of equal length is the number of positions at which the corresponding elements are different. The hamming distance between two vectors A and B can be calculated as,

$$HammingDistance(A, B) = \sum_{i=1}^d (A_i - B_i) = 2$$

Therefore, the hamming distance between A and B is 2. The maximum hamming distance between two vectors can be d . The lower the distance the higher the similarity.

Euclidean Distance. Euclidean distance is the straight-line distance between two vectors in Euclidean space. The euclidean distance between two vectors A and B can be calculated as,

$$\begin{aligned} EuclideanDistance(A, B) &= \sqrt{\sum_{i=1}^d (A_i - B_i)^2} \\ &= \sqrt{(0 - 1)^2 + (1 - 0)^2 + (0 - 0)^2 + (1 - 1)^2 + (1 - 1)^2} \\ &= \sqrt{2} \end{aligned}$$

Therefore, the Euclidean distance between A and B is $\sqrt{2}$. The maximum euclidean distance between two vectors can be \sqrt{d} . The lower the distance the higher the similarity.

3.4 Problem Definition

Suppose, a social service provider (e.g., flicker) wants to outsource users' profile data and shares images to the semi-honest cloud server and at the same time provides social discovery service to interested users. The objective of this chapter is to enable semi-honest cloud server to perform social discovery operation in a privacy-preserving manner.

Example 1. *Figure 3.2 shows a query consists of multiple images (row 1). The goal is to find those people who have uploaded similar images; hence showing similar interests. Figure 3.2 shows the top-5 results of such a query.*

3.4.1 Source Data

In this work, we have run the experiment using Flickr image dataset (MirFlickr-25K [25]) that contains 25K images. Some of the images are depicted in Figure 3.2 where images contain different targets, i.e., people, trees, dogs, etc. These targets can be represented as different attributes as one image can contain different attributes. For example, in Figure 3.2, our query individual has interest in dogs, nature and human portraits. We can generate a specific feature set from these images using CNN (discussed in Section 3.5) and represent it as shown in Table 3.2.

In Table 3.2 each row represents a user, and each column except the first one represents a visual object class. If we call the table UP and $UP_{i,j} = 1$, then the visual object in the j^{th} column is associated with the preference of the user of the i^{th} row.

Table 3.2: Sample dataset with binary feature attributes

User	Dog (f_1)	Cat (f_2)	Sky (f_3)	Portrait (f_4)
u_1	0	1	0	1
u_2	0	1	0	1
u_3	1	0	0	1
u_4	1	0	1	1

3.4.2 Security Requirement

The diversity and massive size of user-shared content and inter-user content interactions have raised new concerns for social discovery and other online applications, among which privacy preservation of user-preferences which is a major challenge [17]. In this work, we preserve the security of user’s shared contents by encryption before outsourcing the data into the cloud server. We considered two different encryption methods, symmetric and asymmetric as they have different properties discussed in Section 3.6. The required computation regarding NNS (i.e., using different similarity metrics among user profiles) is to be performed over encrypted data using the Intel® SGX architecture.

3.4.3 Utility Requirement

Finding nearest neighbours for a target user by measuring similarity among millions of users and their attributes in a brute force way is time exhaustive and often impractical. Therefore, we have to adopt a scheme to reduce the number of candidates who will be qualified for the similarity measurement during the NNS. However, this reduction of search space may introduce errors resulting utility loss.



User	Preferred Images				
Query					
Top1					
Top2					
Top3					
Top4					
Top5					

Figure 3.2: Top-5 query on a image-centric social discovery

Hence, our objective is to minimize this error with respect to the original method (as described in Section 3.3.2).

3.5 Secure Nearest Neighbour Search Using SGX

The approach begins with user uploading images to social networking sites to find a match with other users who have common visual interests. After uploading preferred images, users may send a request for a social discovery to the social network service provider. In the following, we detail the steps conducted by the service provider and the cloud server to securely reply to the user's request.

3.5.1 Service Provider

To enable the cloud to remotely perform a secure NNS for providing social discovery service there are some steps that should be conducted by the service provider.

Feature Extraction

The service provider builds a binary $user \times feature$ matrix by extracting features from the images using a Convolutional Neural Networks (CNN). CNN is a relatively new and popular approach to deep learning. A network of such kind is comprised of one or more convolutional layers and then followed by one or more fully connected layers as in a standard multilayer neural network [38]. The network layers act as a detection filter and try to predict specific features or patterns present in the original data.

The adopted CNN in this chapter was taken from imagenet model [38] and further tuned with 25K images provided by the MirFlickr-25K dataset [25]. The MirFlickr-25K dataset contains 17 different classes based on most popular image tags on Flickr. Suppose a user has uploaded 50 images which are provided to the convolutional network. The convolutional network outputs the corresponding classes presented in those images. This classification is named as an image profile of that specific user.

For training purposes, we split the 25k images into training and testing parts. On the testing part, we achieved top-1 to top-5 error rates on the range of approximately 11%. Table 3.2 shows a transformation of user image profiles into a binary $user \times feature$ matrix. It is noteworthy that the bag of words model [4] has been proposed to generate such profiles which certainly has less accuracy than our CNN architecture. If we visualize a sample user profile (m_1, m_2, \dots, m_d) , then having '1' in the m_i position denotes that the user has a preference on the i^{th} visual class (e.g., dog, flowers, birds) and d is the total number of classes that we are interested in. In Table 3.2 each row represents a user, and each column except the first one represents a visual object class. Since, an image profile only contains a sequence

of '0' and '1' values to represent user preferences, we termed it as a binary valued user image profile.

In summary, the service provider extracts visual features (e.g., dog, flowers, portraits) from uploaded images by users using CNN and then generates feature vectors. Then all the feature vectors are combined and normalized to generate user image profiles. This normalization helps to build efficient user profiles by focusing on the preferences of users and eliminating noise that has a negative impact on the social discovery result.

Search Index Construction

From the feature extraction using CNN, we end up with a feature set which can be represented as a collection of vectors. Hence, similarity can be computed by measuring the distance or closeness among these vectors using different similarity metrics. The similarity metric can be Euclidean distance measure, Jaccard similarity, or Cosine similarity as detailed in Section 3.4. Performing pairwise comparisons between sets is time exhaustive because the number of comparisons grows *exponentially* with the size of the sets. Furthermore, most of those comparisons are unnecessary because they do not result in matches. In this chapter, we depend on MinHashing and LSH to solve these problems.

MinHashing is the process of converting large sets to short signatures, while preserving similarity [19]. MinHashing makes it possible to compute possible matches only once for each element such that the cost of the computation grows linearly rather than exponentially. Computing of MinHash signatures requires random permutations. It is possible to perform random permutations by using random hash functions. The size of the MinHash signature is same as the num-

Algorithm 4: Computing MinHash Signatures**Input:** User profile matrix $UP[d, n]$, Number of hash functions k **Output:** MinHash Signatures $SIG[k, n]$

```

1: for  $i = 1$  to  $k$  do
2:   for  $j = 1$  to  $n$  do
3:      $SIG(i, j) = \infty$ 
4:   end for
5: end for
6: for  $i = 1$  to  $k$  do
7:   Compute  $h_i, h_{2i}, \dots, h_{ni}$  for the corresponding rows ( $UP_{1*}, UP_{2*}, \dots, UP_{n*}$ )
8: end for
9: for  $l = 1$  to  $k$  do
10:  for  $j = 1$  to  $d$  do
11:    for  $i = 1$  to  $n$  do
12:      if  $UP_{i,j} = 1$  then
13:        Set  $SIG(j, l)$  to the smaller of the current value of  $SIG(j, l)$  and  $h_{jl}$ .
14:      end if
15:    end for
16:  end for
17: end for
18: return  $SIG[k, n]$ 

```

ber of hash functions used to compute signatures. The signature size can be initialized at the beginning. In the example shown below we use two random hash functions and accordingly the size of the MinHash signature is two. MinHash signatures for each user entity u is computed following the steps of Algorithm 4.

To facilitate understanding Algorithm 4, suppose that each column in Table 3.3 represents a user profile and we have two random hash functions: $h_1(x) = x + 1 \pmod{5}$ and $h_2(x) = 3x + 1 \pmod{5}$. The values of the two hash functions are calculated from the row numbers given in the first column of Table 3.3. Here, the value of x comes from the row number (e.g., $0, 1, 2, \dots, d$ where d represents the number of the classes). For example, if we are operating on row marked as 3, then $x = 3$ and the value of the first hash function is $3 + 1 \pmod{5} = 4$.

Table 3.3: Hash functions computed for the user profile matrix

Row	u_1	u_2	u_3	u_4	u_5	$x + 1$ mod 5	$3x + 1$ mod 5
0	0	0	1	1	0	1	1
1	1	1	0	0	1	2	4
2	0	0	0	1	0	3	2
3	1	1	1	1	1	4	0
4	0	0	0	0	0	0	3

Table 3.4: Computing MinHash signatures (Step-1)

	u_1	u_2	u_3	u_4	u_5
h_1	∞	∞	∞	∞	∞
h_2	∞	∞	∞	∞	∞

Let us simulate Algorithm 4 for computing the signature matrix. The signature matrix is initiated to ∞ as shown in Table 3.4.

At first, we consider row 0 of Table 3.3 where the values of $h_1(0)$ and $h_2(0)$ are both 1. The row numbered 0 has 1's in the columns for users u_3 and u_4 and this why only these columns of the signature matrix can change. As 1 is less than ∞ , we change both values in the columns u_3 and u_4 . The current estimate of the signature matrix is shown in Table 3.5.

Table 3.5: Computing MinHash signatures (Step-2)

	u_1	u_2	u_3	u_4	u_5
h_1	∞	∞	1	1	∞
h_2	∞	∞	1	1	∞

Now, we operate on the row numbered 1 in Table 3.3. This row has 1 in u_1 , u_2 and u_5 , and its hash values are $h_1(1) = 2$ and $h_2(1) = 4$. The new signature matrix is shown in Table 3.6. If we continue in this way, we will end up with the final signature matrix shown in Table 3.7. It is clear from the table that the signatures of u_1 , u_2 and u_5 are identical (when comparing column-wise). The signatures of u_3 and u_4 are also identical. Accordingly, their original user profiles should be similar as well.

Table 3.6: Computing MinHash signatures (Step-3)

	u_1	u_2	u_3	u_4	u_5
h_1	2	2	1	1	2
h_2	4	4	1	1	4

Computing signatures once with random hash functions may affect the NNS accuracy. So, we combine Locality Sensitive Hashing (LSH) with MinHashing. LSH

Table 3.7: Computing MinHash signatures (Final Step)

	u_1	u_2	u_3	u_4	u_5
h_1	2	2	1	1	2
h_2	0	0	0	0	0

is an efficient algorithm to run a faster NNS in high dimensional spaces [20]. The methodology of LSH is to hash items several times so that the items which appear to be similar are more likely to be the member of the same bucket than dissimilar items. In this work, we apply LSH with MinHashing by repeating MinHashing for several stages. Actually, the steps that we have just explained will be repeated more than one time. If the signatures of user profiles are matched in any of the stages then the user profiles are considered as candidates for similarity comparison.

The method works in such a way that the items which are not similar will never hash to the same bucket, and as a result, will never be checked [19]. The size of the MinHash signature changes at every stage depending on the similarity threshold and stage number. Similarity threshold can be defined as the similarity score up to which candidates will be considered as a similar pair. For example, if the similarity threshold is set to 0.7, then the vectors whose similarity score is 0.7 or higher will be considered as a candidate pair. If the similarity threshold is defined as t and the stage number as s then the size of the MinHash signature is defined using the equation [19]:

$$signature\ size = (\ln(1/s) / \ln(t) + 1) * s$$

It is noticeable from the equation that if we increase the similarity threshold, the signature size will increase. If we consider similarity threshold as 0.7 and if we

are in the second stage number $s = 2$ then the signature size will be:

$$\text{signature size} = (\ln(1/2)/\ln(0.7) + 1) * 2 \approx 6$$

The minimum MinHash signature size can be 1. In our experiment, we defined the similarity threshold as 0.7 and the number of stages as 3. After generating buckets, we assign a bucket for each user based on the similarity threshold. At the end of this phase, a bucket containing potential candidates is generated for each user. This bucket is denoted by the *search index* for that user. The approach LSH with MinHashing is more efficient whenever we have high-dimensional datasets. MinHashing saves lots of computation time by reducing data dimension whereas LSH reduces the number of candidates for similarity measurement. To make the search more efficient, the duplicate elements of each bucket are removed. As a result, same candidates are never checked twice while executing similarity measurement. The bucket size depends on the number of candidates having the same hash value at any of the hashing stages.

Encryption Phase

These buckets and user profiles are then secured via symmetric or asymmetric encryption and uploaded to the cloud. We have used both symmetric (AES-128 in CBC mode) and asymmetric encryption (Paillier). In AES-CBC mode, all the blocks are chained together and the encryption is randomised by using the initialization vector (IV) [49]. On the other hand, Paillier [50], the asymmetric encryption, is secure and probabilistic in nature. For example, if we encrypt the same message several times using Paillier, it will generate different ciphertexts every time.

The performance of asymmetric and symmetric cryptosystems with SGX re-

Algorithm 5: Encryption Phase

Input: Total number of users (n), secret key (sk), initialization vector (iv)**Output:** Encrypted user profiles (EU^*) and their corresponding encrypted buckets (EB)

- 1: **for** $i = 1$ to n **do**
 - 2: Build user profile (U_i^*)
 - 3: Compute bucket (B_i) consisting of candidate pairs for user profile (U_i^*) using LSH with MinHashing
 - 4: $EB_i = \text{BucketEnc}(B_i, sk, iv)$
 - 5: $EU_i^* = \text{UserEnc}(U_i^*, sk, iv)$
 - 6: Upload EU_i^* and EB_i to the cloud
 - 7: **end for**
 - 8: **return** EU^*, EB
-

garding secure NNS is studied in the experimental section. In the case of symmetric cryptosystem, the secret key is generated by the service provider and securely stored in the enclave of SGX. In the case of asymmetric cryptosystem, the public key and the private key are generated by the service provider. The public key is used for encrypting the user profiles and the corresponding buckets whereas the secret key is securely stored in the enclave of SGX. Algorithm 5 summarizes the main steps before outsourcing to the cloud considering using the symmetric encryption.

3.5.2 Cloud Server

Algorithm 6 presents the main steps of secure NNS at the cloud server end. All the operations regarding the computation of the nearest neighbour for a given user

profile are executed inside the secure enclave of the SGX.

Algorithm 6: Secure NNS

Input: Encrypted query (q'), Encrypted buckets (EB)

Output: Encrypted search result (s)

- 1: Decrypt query (q', sk)
 - 2: Decrypt query user profile (U_q^*, sk, iv)
 - 3: Decrypt the corresponding bucket (B_q, sk, iv)
 - 4: **for each** candidate i in B_q **do**
 - 5: $d = \text{ComputeDistance}(U_i^*, U_q^*)$
 - 6: $R = \text{RankResult}(U_i^*, d)$
 - 7: **end for**
 - 8: $S = \text{Encrypt}(R, sk)$
 - 9: **return** S
-

Decryption Phase

When a user initiates the social discovery service from the service provider, the query is encrypted and forwarded to the cloud server end. We listed an encrypted bucket for each user. Depending on the user who initiates the search, the corresponding bucket is decrypted inside the enclave of the SGX and the bucket elements are considered the qualified candidates for the similarity search.

Similarity Computation

The elements of the corresponding buckets are only considered as potential nearest neighbour candidates. After that, the candidate user profiles are decrypted and the similarity is measured using the defined metrics in Section 3.3.2. Then results

are ranked according to the distance among the candidates and the query user profile. The query retrieves the top similarity matches from the cloud and returns the encrypted result to the service provider. After receiving the encrypted NNS result, the service provider decrypts it and returns the result to the target user.

3.6 Analysis

In this section, we analyze the security and the complexity of the proposed framework. In the semi-honest trust model usually, there are two possible attacks; Cipher-text only attack (COA) and Known background attack [51; 52]. At first, we formally define our proposed SGX-based secure NNS scheme with notations and then prove the security of the scheme against COA and Known background attack using the definition and notations.

Definition 3.6.1 (SGX-based secure NNS). *A SGX-based secure NNS scheme is a tuple of seven polynomial-time algorithms ($keySetup$, $BucketEnc$, $UserEnc$, $QueryEnc$, $ResultDec$, $IntializeSearch$, $SecureNNS$)*

Algorithms executed at the service provider end:

$sk \leftarrow keySetup(1^\eta)$: Takes a security parameter η as input and outputs the secret key sk .

$\beta \leftarrow BucketEnc(b, sk, iv)$: Encrypts the buckets using the secret key and the corresponding initialization vector iv .

$\alpha \leftarrow UserEnc(u, sk, iv)$: Encrypts the user profiles using the secret key and the corresponding initialization vector iv .

$q' \leftarrow EncryptQuery(q, sk)$: Encrypts the query using the secret key sk .

$R \leftarrow \text{DecryptResult}(s, sk)$: Decrypts the search results using the secret key sk and returns the decrypted result R .

Algorithm executed at the cloud server:

$C \leftarrow \text{InitializeSearch}(q', \alpha, \beta)$: Calls the secure hardware function SecureNNS (Algorithm 6).

Algorithm executed at the enclave of the secure-hardware:

$S \leftarrow \text{SecureNNS}(q', \alpha, \beta)$: Performs NNS (using algorithm 6) inside the protected enclave of SGX decrypting the encrypted query, buckets and user profiles accordingly and output the encrypted search result S .

Proposition 3.6.1. *(Security against COA) The proposed scheme is secure against the COA.*

Definition 3.6.2 (COA). *The definition of COA assumes that an adversary A' has access to the only certain sequence of cipher-texts c_1, \dots, c_n . A scheme is COA secure if one can show that an adversary A' cannot deduce any useful information from the cipher-texts.*

Proof. In our proposed framework, all the user profiles and buckets are encrypted using AES-128 CBC or the Paillier cryptosystem before being outsourced to the cloud to preserve the privacy of user's sensitive data. The secret key sk is securely stored in the protected enclave os SGX. Due to the additional protection on the memory portion used by the enclave, the adversary A' cannot access the secret key sk . The well-established cryptographic primitives make it computationally difficult for the adversary A' to decrypt cipher-texts or deduce any useful information without knowing the secret key sk [53]. Therefore, we can say that the adversary A' cannot infer anything about the underlying data of the users and the proposed

model is secure against COA. ■

In any searchable encryption there are some leakages during the search process as the search index is deterministic. We denote the leakage function as L_{NNS} which contains the data size (i.e., the total number of encrypted data records stored in the cloud server) and the search pattern (i.e., whether same encrypted data is retrieved for two different queries). Patterns are the default information leakage in any efficient searchable encryption scheme [4; 54]. Next, we prove the security of our proposed framework against the leakage function L_{NNS} .

Proposition 3.6.2. *(Security against known background attack) The proposed scheme is secure against the known background attack given the leakage function L_{NNS} .*

Definition 3.6.3 (Known background attack). *Known background attack is a possible attack on the real world scenario for any cloud application. The cloud server (or an adversary) may have some statistical information revealed from the leakage function L_{NNS} . In this attack, the adversary A' becomes active during the computation regarding NNS inside the protected enclave of SGX and tries to gain information about the access patterns of the encrypted files α, β . Moreover, A' analyzes the encrypted query q' and the search result S . Finally, A' runs a polynomial number of adaptive experiments combining the leakage function L_{NNS}, q', s , and the access pattern information to deduce/identify certain keywords in the query q' . We say that a scheme is secure against known background attack is if A' cannot predict certain keywords in q' .*

Proof. In our model, the secret key generated by the service provider is securely stored in the enclave of SGX. The SGX hardware enforces additional protection on the memory portion used by the enclave. As a result, the cloud server can not

access the protected memory of the enclave. In particular, all other software in the system, including privileged software like OS, hypervisor, and firmware cannot access the enclave memory [41]. In addition to that, no cipher-texts are decrypted outside the enclave. The access patterns of encrypted files are not revealed as all the encrypted files are loaded in the enclave and corresponding files matching with query q' are decrypted. Moreover, the encrypted results and L_{NNS} do not reveal any information about the access patterns. In this scenario, it is computationally difficult for A' to identify certain keywords in q' without knowing the sk . Therefore, we can say that our proposed model is secure against the known background attack as an adversary A' cannot predict or deduce any keywords from the cipher-texts or infer anything except the leakage function L_{NNS} that we stated above. ■

Proposition 3.6.3. *(Complexity) The computation cost of Algorithm 4 and Algorithm 5 is linearly bounded by $O(sk + md)$ where m is the bucket size, k is the number of hash functions and s is the number of hashing stages.*

Proof. Given a query point q , the algorithm iterates over the s number of stages with a number of random hash functions k . At the end, for each query, a bucket is generated with m numbers of elements which are considered as potential candidates for being the nearest neighbours. Hence, if the dimension of the query point (i.e., data point) is d , the time complexity of LSH for finding the nearest neighbours is $O(sk + md)$. Usually, the number of elements, m in each bucket is much smaller than the number of users, n ; that is $n \gg m$. ■

Due to the static number of $O(n)$ searches in the dataset, the execution time will not differ and always be proportional to the number of users. Also, the access pattern into the dataset is sequential which does not reveal the similarity.

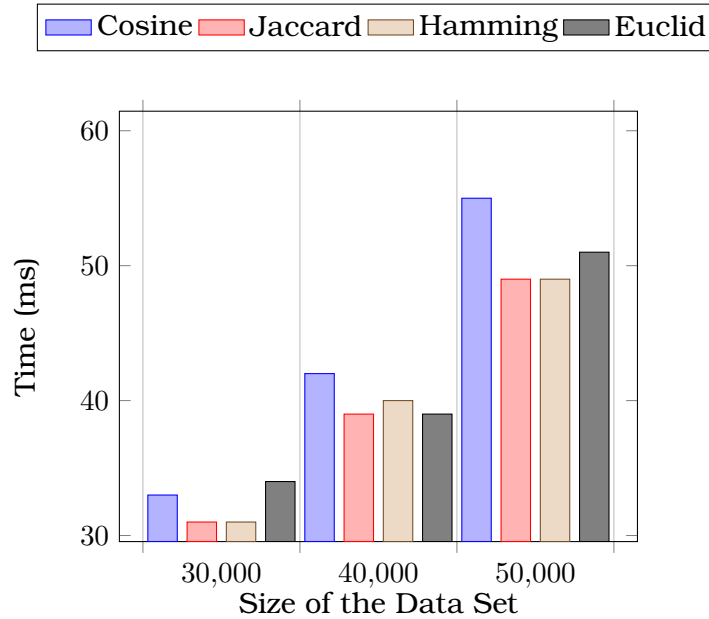


Figure 3.3: Computation time for variable dataset size in X-axis considering 17 features for 4 different similarity metrics in plain-text for a top-10 query (Naive Approach)

However, the indexing can be further improved. For example instead of loading all the encrypted files only the required files for performing NNS can be loaded in the enclave. But it requires hiding the access pattern information.

3.7 Experimental Evaluation

In this section, we analyze the performance of the proposed framework.

3.7.1 Implementation Details

The implementation of the prototype of the proposed framework (Figure 3.1) was done in two phases. In the first phase, we used the convolutional neural network for extracting features from images. For this purpose, we have utilized a

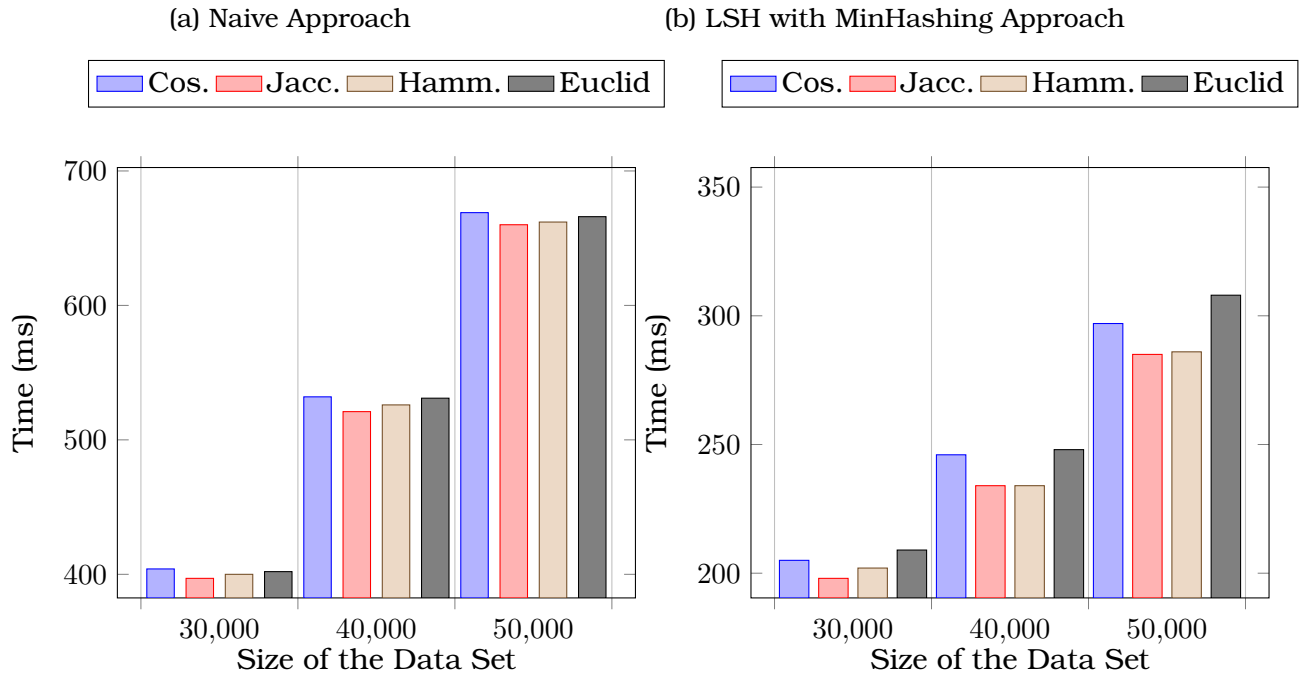


Figure 3.4: Computation time for variable dataset size (30k, 40k, 50k) in X-axis considering 17 features for 4 different similarity metrics for a top-10 query system with Intel core i5-4590 CPU and 8GB RAM. To implement CNN, we used a MATLAB toolbox called MatConvNet¹. We used a pre-trained model in the toolbox (imagenet [55]) and further tuned it according to MirFlickr-25k dataset [25]. After training and tuning the model, we extracted the features from these images. The extracted features were then randomly assigned to different users and used to generate binary valued user profile vectors. Then we constructed the search index for each user using LSH and MinHashing.

In the second phase, we evaluated the performance of the NNS using the SGX architecture. The experiments were run under Intel Core-i7-6700 processor with 8 GB of RAM with secure SGX features. We have measured the computation time of nearest neighbours inside the protected enclave of SGX applying naive approach

¹<http://www.vlfeat.org/matconvnet/>

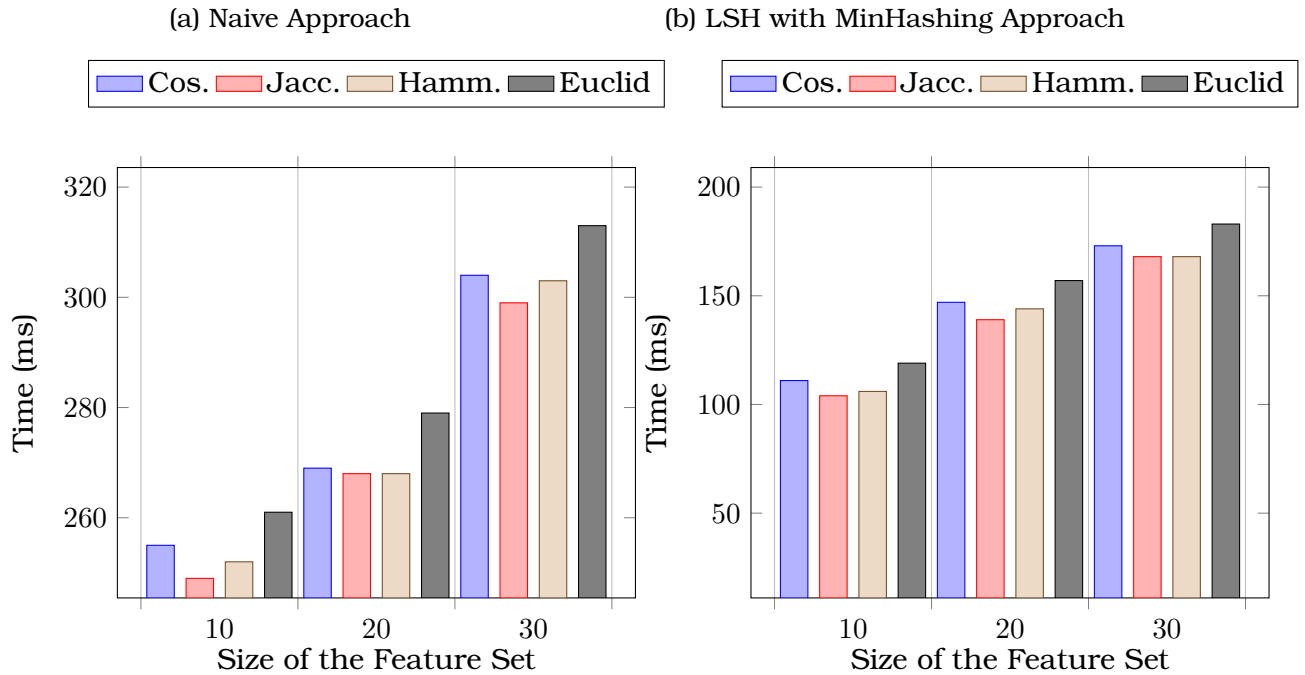


Figure 3.5: Computation time for variable feature set size (10, 20, 30) in X-axis considering fixed data set size 20k for 4 different similarity metrics for a top-10 query and LSH with MinHashing approach.

3.7.2 NNS Performance Evaluation

MirFlickr-25k consists of total 50k users where we vary the number of users while performing the NNS (30,000, 40,000, and 50,000). We tested the proposed model with different 50 queries where each of them was randomly assigned different images from the dataset.

Afterwards, features were extracted from those images using CNN and user profiles were generated in the same way described in Section 3.5. The performance of the NNS using the predefined 4 similarity metrics in plain-text is shown in Figure 3.3. The performance of the secure NNS is evaluated using the security

notion of symmetric and asymmetric encryption.

Symmetric Cryptosystem

In this experiment, all the encryption and decryption operations are done using AES-CBC mode. First, for the random target user profiles (from 50 queries), similarity computations and search queries were performed over encrypted data within the enclave with four predefined metrics discussed in Section 3.3.2. In this phase, we used the naive approach to find the *top 10* nearest neighbours for the target user profile. The results are shown in Figure 3.4a.

Second, we run the same experiment on the encrypted user profiles applying LSH with MinHashing technique. The results are shown in Figure 3.4b. The experimental results in Figure 3.4b demonstrates that the use of LSH reduces the computation time of the NNS operation more than twice (compared to Figure 3.4a). It is noteworthy that with LSH, the number of candidates during similarity search is reduced to the number of candidates in the bucket. Therefore, applying LSH improves the computation time of the nearest neighbours to a great extent.

Third, we run the experiment by varying the feature set size (different number of features) using a fixed number of users to find the top-10 nearest neighbours for the target user profile. The experimental results are shown in Figure 3.5a and Figure 3.5b which demonstrate that varying the feature set size varies the computation time. If we increase the feature set size, the dimension d is also increased. As a result, the computation time is also affected according to the time complexity analysis given in Proposition 3.6.3.

Asymmetric Cryptosystem

In this experiment, all the encryption and decryption operations are done using the Paillier cryptosystem [50]. Although asymmetric encryption provides better security guarantee by being probabilistic in nature, the required time for performing a secure NNS using asymmetric encryption is far from practical for any social discovery application. We have evaluated the performance of asymmetric key encryption using the naive approach on a small dataset and the corresponding results are presented in Figure 3.6.

The results depicted in Figure 3.6 illustrates that even for a small number of users, the performance of asymmetric cryptosystem in SGX is inefficient comparing to the symmetric cryptosystem. The primary reason behind that lies in the decryption (or exponentiation) which is much slower and computationally expensive. In comparison, the symmetric encryption merely does an XOR operation for decryption. As the performance of asymmetric cryptosystem is quite poor in comparison with the symmetric cryptosystem, we restricted our experiments with the asymmetric cryptosystem up to the performance evaluation of the naive approach demonstrated in Figure 3.6.

3.7.3 Accuracy Analysis

Accuracy is another important aspect of any social discovery platform. We measured the NNS accuracy of our model using the F_1 -score measure based on precision and recall values comparing with the baseline method outlined in Algorithm 3. Recall is calculated based on the number of relevant records retrieved by the top- k NNS using LSH to the total number of relevant records retrieved by the naive search. Precision is measured based on the ratio of the number of relevant records

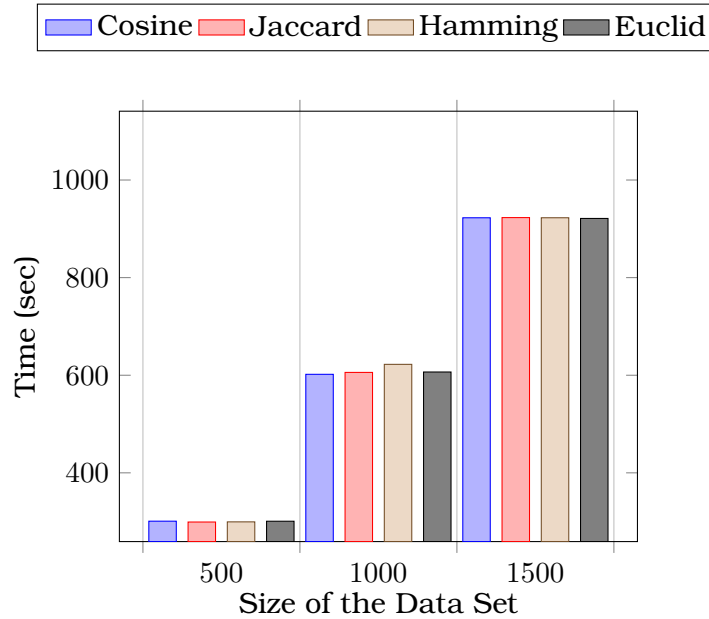


Figure 3.6: Computation time for variable records size (in X-axis) considering 17 features for 4 different similarity metrics using asymmetric cryptosystem for a top-10 query (Naive Approach)

retrieved to the total number of irrelevant and relevant records retrieved. After that F_1 -score is calculated based on precision and recall values [56].

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Algorithm 3 finds the $top-k$ nearest neighbours for a query based on measuring the closeness between the users using 4 different similarity metrics. This approach requires the distance comparison to be done over all the candidates and their features. As all the users are considered on a search query, this gives better accuracy but results in longer computation time which is not feasible for social discovery applications. On the contrary, LSH only measures the closeness amongst the candidates who falls in the same bucket after a number of hashing stages. The number of candidates in each bucket is usually much smaller than the size

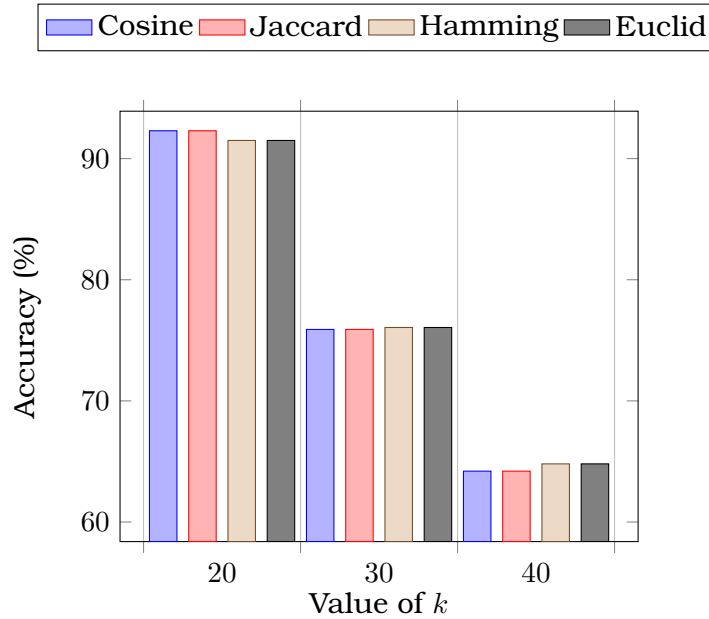


Figure 3.7: Accuracy for NNS on different top- k -values using LSH with 4 different Similarity metrics for data set size 50k

of the dataset or the number of users. The accuracy analysis of LSH based on the F_1 -score for different top- k values is shown in Figure 3.7.

Analysing the results presented in Figure 3.7, we remark that our approach achieves reasonable accuracy measurement, yet always consume smaller computation time. It is apparent from Figure 3.7 that the accuracy gradually decreases as the top- k value increases. It is because there is some information loss during MinHashing and LSH stages. When the top- k value is small the similarity difference between the top- k neighbours and other neighbours is bit higher, so LSH performs better in this case. On the other hand, when the top- k value is large the similarity difference is too low and due to information loss, other neighbors are considered as the top- k neighbours which affect the accuracy.

3.8 Related Work

The related work regarding the secure NNS can be broadly divided as cryptographic and secure hardware based solutions based on the available security primitives. We already discussed some related works in Table 3.1. Below, we further discuss different characteristics of the most relevant approaches.

3.8.1 Cryptographic Approach

There are some existing works performing the NNS over encrypted data using cryptographic protocols [36; 57]. However, most of them do not consider the high dimensionality of data. Gao *et al.* [53] proposed a scheme using locality sensitive hashing and order-preserving encryption. Recently, Wang *et al.* [9] proposed a scheme to perform NNS over high-dimensional encrypted data using R trees and order-preserving encryption.

Elmehdwi *et al.* [8] solves the NNS problem for encrypted data through Paillier encryption which is further modified by Samanthula *et al.* [37] who introduces secret sharing scheme with Paillier encryption providing more formal security proofs. To compute the nearest neighbours, they have executed the traditional k -nearest neighbour query based on euclidean distance in a secure fashion. In general, the secure NNS problem is similar to the existing work on search over encrypted data on the cloud [7; 30; 58] and secure ranked search over high dimensional encrypted data [59; 60; 31]. However, these approaches do not perform well in the high-dimensional space.

The computation regarding the secure NNS is also similar to secure social discovery applications. Yuan *et al.* [4] proposed a privacy-preserving image-centric

social discovery where the user image profiles were also encrypted and stored in the public cloud upon which the NNS operation was performed when social discovery request was generated. They used a cryptographic approach based on searchable symmetric encryption [7] combined with locality sensitive hashing and cuckoo hashing [61] for performing secure NNS.

3.8.2 Secure Hardware-based Solutions

The secure hardware architecture is the most recent technique introduced to run secure computations inside the protected enclave. Fuhry *et al.* [41] utilized the efficiency of secure-hardware with B^+ tree to perform range queries over single dimensional encrypted data. But their solution does not support queries over high-dimensional encrypted data. Scuster *et al.* [42] proposed a system that allows performing map-reduce computations in the cloud preserving the security of code and data. There are some recent works [43; 44; 62; 48] with impressive outcomes motivating the use of secure hardware architecture to run computation over encrypted data. Although related, none of these techniques use secure hardware-based solutions to perform a secure NNS over high-dimensional encrypted data. To the best of our knowledge, this work proposes the first secure hardware-based solution for the NNS problem.

Chapter 4

Obfuscated Image Classification and Nearest Neighbour Search for Friend Recommendation

4.1 Introduction

An online community is a group of people having common interests who use the online services to interact, work, and pursue together their interests over time. Understanding the link between people is a primary requirement to build online smart communities [63]. The social network is one of the tools people are adopting to create virtual neighbourhoods and bonding [64]. Recently, the recommendation has emerged as a vital service provided by almost all social networking sites (e.g., Facebook, Google+, and LinkedIn, etc.) in the road towards forming the online smart community among the citizens of the smart city.

The recommendation can come in various forms such as joining online blogs

or communities, friend recommendation and product recommendation, etc. Social network analysis might reveal some new insights regarding the preferences of an individual and it is essential for supporting the smart recommendation services. For example, today one of the most common behaviours of a smart citizen is to share pictures of their travels or a decent meal on the social networks even before enjoying it; this results in a massive amount of photos and videos to be hosted online. According to a report in 2014 [2], around 60 million images are shared every day on Instagram to that date. The availability of this large quantity of images enables the social networking service providers to provide smart recommendations of similar users using the visual features of the user's uploaded images. However, image-centric recommendation service requires building user profiles extracting visual features from the images. As a result, the quality of recommendation services depends on the accuracy of feature extraction.

The recent advancement in machine learning based on artificial neural networks has led to seminal improvements for extracting visual information from images. The deep convolutional neural (DCNN) network and autoencoder (AE) are two promising techniques in this field. Because of the resemblance with the human brain, these techniques can classify objects more accurately than the existing models (e.g., the bag of words and feature based methods) and helps service providers to build relevant user profiles. It is no surprise that many social networking applications are now considering the image features as one of the major factors to provide better recommendation service [4]. Unfortunately, there are some privacy concerns which may demotivate the users sharing image contents in order to get recommendation services. Overcoming these challenges and utilize efficient nearest neighbour search approach to provide image-centric friend

recommendation is the focus of this chapter.

4.1.1 Privacy Implications

The image-centric friend recommendation service has some privacy implications as these user's images contain many sensitive attributes which can be used by the malicious adversaries to re-identify a user. For example, one can upload images in a social network containing his preferred objects (e.g., dog, sky, flower, etc.) that can reveal his personal preferences to the adversaries. In addition to that, the metadata associated with images such as geolocation and time when the photo is taken is sensitive as it can reveal personal information and used for executing re-identification attack. Moreover, different techniques have been developed that can identify objects in images. For example, the face recognition technology developed by Facebook is one of its most appealing features which has become a matter of privacy concerns [65]. Due to this controversy, Facebook turned off the service of face recognition in 2012 but bought it back again due to the popularity of image search. Furthermore, Google has decided to disable face recognition in Google Glasses [11].

To some extent, the above privacy concerns come from the fear that images may be illegally accessed by some malicious adversaries as most of the service providers are using the cloud as a solution to storage constraint problem. Simply disabling the feature of automatic face recognition does not solve the problem as it also eliminates the data utility for image search functionality.

Apart from the concerns stated above the cloud services are also prone to privacy attacks. Some recently published news clearly demonstrates that privacy should not be expected to be preserved by cloud service providers [66]. One prime

example which underlines this scenario is the recent incident regarding the hacking of the iCloud image storage service and celebrity photo leakage [67]. Similarly, there exist some other glaring examples which demonstrate the risk of the privacy breach in cloud-based visual data storage. These are the primary reasons that made the users concerned about protecting the privacy of their images.

4.1.2 Privacy Protection

Various tools to ensure image privacy exist, including image blurring, mosaic, scrambling and encryption. In 2012, YouTube introduced a technique to automatically blur all faces in a video [68]. This automatic facial blurring is presented as an effective solution to improve video privacy of Youtube users. For example, this technology motivates someone to share sensitive protest footage without exposing the faces of the activists involved. Similar technologies can be adopted to protect the privacy of images.

There are some other privacy preserving techniques which perform face detection and divides the images into private and public parts. However, this approach does not address our privacy goals completely and makes the system more complex in architecture. For example, if an image is leaked from the cloud; the attackers can obtain significant information from the non-obfuscated parts (e.g., objects in the background) [65].

Some recent attack models in the literature concentrate on partially obfuscated parts of the images to re-identify obfuscated parts also demonstrate the risk [69]. Therefore, obfuscating the full image using the lightweight image obfuscation techniques such as symmetric encryption or photo blurring is a more practical solution for protecting the sensitive information of images while enabling

Table 4.1: Comparisons among previous schemes for obfuscated image classification problem. Our proposal is the only technique that provides friend recommendation service using obfuscated image classification

Existing Works	Year	Obfuscation Primitive	Extracted Features from Images	Feature Extraction Model
Ra et al. [65]	2013	Encryption	✗	✗
Yuan et al. [4]	2014	✗	✓	Bag of Words
Zhang et al. [11]	2015	Blur	✗	✗
Yuan et al. [70]	2015	Scrambling	✗	✗
Wang et al.[71]	2016	Encryption	✓	Autoencoder
Our proposal	2017	Encryption and Burring	✓	Deep AE and DCNN

classification task to provide different utility services.

4.1.3 Contributions

- The main contribution of this chapter is designing a practical framework for privacy-preserving image-centric friend recommendation utilizing efficient nearest neighbour search approach which protects the privacy of user’s shared images and at the same time allows user profile generation classifying the obfuscated images.
- We have used two popular image obfuscation techniques blurring and encryption in our proposed model. To the best-of our knowledge, our work is the first to deal with obfuscated image classification for providing image-centric friend recommendation service.

- Our work demonstrates that deep neural networks can be used as a tool for obfuscated image classification. The proposed model utilizes the efficiency of DCNN for classifying blurred images and deep autoencoder for classifying encrypted images.
- We compare the accuracy of our proposed image obfuscation techniques encryption and photo blurring. The experimental results signify the utility of the proposed model. Our proposed model achieves 83.94% accuracy while the existing model [71] achieves 79.83% for encrypted image classification over MNIST dataset.

The rest of the chapter is organized as follows: Section 4.2 gives an overview of our proposed model. Section 4.3 formally presents the methodology used in this chapter. Section 4.4 describes the steps of our proposed method. Our performance evaluation is done in Section 4.5. Section 4.6 discusses the related works.

4.2 System Overview

Generally, in recommendation systems, a user profile is built for every user capturing the user's preferences. When a user requests for recommendation to the service providers, a similarity comparison is performed with other users based on his/her user profile, and the computed results are returned back by the service providers. To get a recommendation or to be recommended, users need to reveal some information to the service providers which is utilized by the service providers to build user profiles. The matter of concern is how to hide information in such a way that it reduces the risk of privacy breach as well as preserves the utility of friend recommendation service.

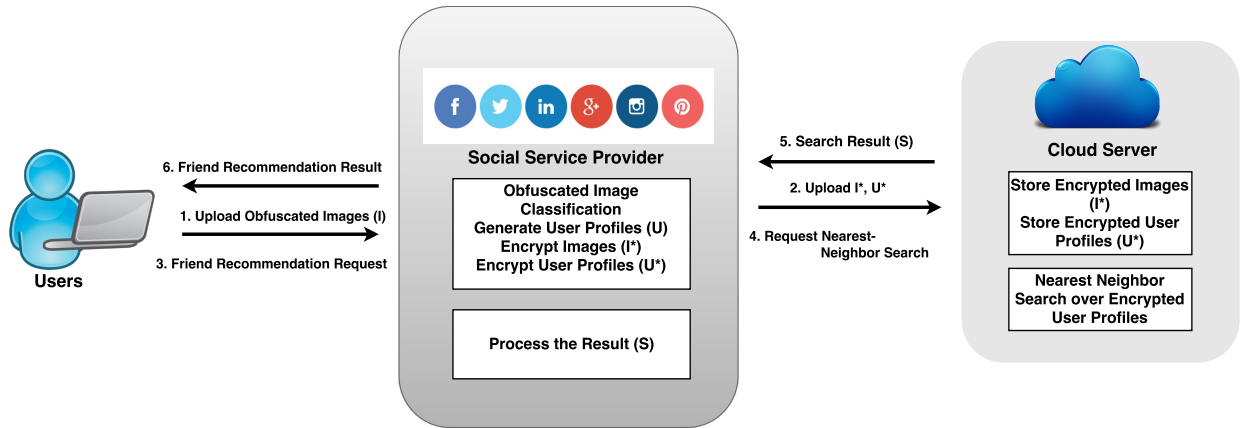


Figure 4.1: Proposed architecture of the system

4.2.1 Architecture and Entities

Fig. 4.1 presents an overview of our proposed system. Our proposed system has three main entities: *users*, *social service providers* and *third party cloud server*. The role of each entity is described below:

- *Users*: *Users* are the individuals who use the social networking applications. They frequently interact with their connected peers and as a part of their activity in the social networking sites, they upload images that they would like to share.
- *Social Service Providers*: This set consists of the entities who offers social networking services to the users. Facebook, Flickr, Google Plus, Instagram etc. fall into this category. In our model, they facilitate the communication between users and third-party cloud server.
- *Third Party Cloud Server*: Social media sites outsource their large volume of images to the third party cloud server (e.g., Amazon cloud). In our architecture, the third party cloud server is assumed to be semi-honest.

4.2.2 System Service Flow

The information flow in the proposed framework, as shown in Figure 4.1, begins when the users upload their images to the social service provider. Before uploading, the images are obfuscated by the users to hide sensitive information. After receiving the images, the social service provider executes the following steps:

- It builds a user profile (U) for each user classifying the obfuscated images using the convolutional neural network and encrypts the user profiles (U^*). To ensure a higher level of security, the obfuscated images are further encrypted (I^*) before outsourcing to the cloud which is not owned by them.
- The encrypted images and the user profiles are uploaded to the cloud.
- When a friend recommendation service is requested by a user, the service provider forwards the request to the cloud. The cloud performs a nearest neighbour search over the user profiles and recommends the top similar users as friend recommendation result.

In this chapter, we mainly focus on classifying obfuscated images to build user profiles which is one of the primary operations for any image-centric friend recommendation services. The remaining operations for the friend recommendation service such as secure nearest neighbour search over encrypted data can be performed adopting one of the efficient existing techniques in the literature [7; 4; 9; 8; 6].

4.2.3 Problem Definition

Suppose a social networking service provider (e.g Flickr) wants to provide friend recommendation service to the interested users based on their shared images.

User	Preferred Images				
Query					
Top1					
Top2					
Top3					
Top4					
Top5					

Figure 4.2: Finding top-5 similar users based on image-centric interest

We consider the fact that, to hide sensitive information, users prefer to obfuscate images before sharing to the service provider. The objective of this chapter is to build user profiles as shown in Table 4.2 for image-centric friend recommendation classifying the obfuscated images.

Example 2. Figure 4.2 shows a query consists of multiple images (row 1). The goal is to find those people who have uploaded similar images; hence showing similar interests. Figure 4.2 shows the top-5 results of such a query.

The images depicted in Figure 4.2 contains different targets, i.e., people, trees, dogs, etc. These targets can be represented as different attributes as one image can contain variant attributes. For example, in Figure 4.2, our query individual has an interest in dogs, nature and human portraits. We can generate a specific feature set from these images using DCNN (discussed in Section 4.3) and represent it as shown in Table 4.2.

Definition 4.2.1 (Obfuscated Image classification). Given a set of obfuscated im-

Table 4.2: Sample user profiles with binary feature attributes

User	Dog (f_1)	Cat (f_2)	Sky (f_3)	Portrait (f_4)
u_1	0	1	0	1
u_2	0	1	0	1
u_3	1	0	0	1
u_4	1	0	1	1

ages for each user, the problem of obfuscated image classification is to build a user profile (e.g., Table 2) without having access to the raw images.

In Table 4.2 each row represents a user, and each column except the first one represents a visual object class. If we call the table UP and $UP_{i,j} = 1$, then the visual object in the j^{th} column is associated with the preference of the user of the i^{th} row.

4.2.4 Threat Model

In our proposed framework, we consider primary attacks from the third party cloud server which is assumed to be semi-honest (also known as honest but curious) entity. In general, a semi-honest party will follow the protocol specification, but they may be interested in learning the sensitive contents of user’s shared images. The social service provider and all users in the system are assumed to be trustworthy. The assumption is justifiable because deviating from the protocol and performing illegal operations on user’s shared data will lead to negative user experience as well as potential revenue loss of the service provider [11]. However, the service providers may perform some operations for better user experience and to increase the utility of service. This may lead to unintentional privacy breach and

users may feel insecure to share the actual data rich in contents. Therefore, it is practical to apply light obfuscation over data before sharing to the service providers and perform a higher level of obfuscation before outsourcing to the cloud. We do not consider other possible attacks, such as malicious user attack at this time.

4.3 Methodology

Before introducing the proposed framework, we discuss the existing and used methodologies for image obfuscation and feature extraction.

4.3.1 Image Obfuscation Techniques

Image obfuscation is the process of hiding sensitive information from images through non-linear pixel transformation and making the image unrecognizable. There are a number of tools to obfuscate images in the existing literature such as blurring, scrambling and encryption. The effect of different obfuscation techniques is shown in Figure 4.3.

Blurring: A normalization box filter is applied to obfuscate the images. Blurring can be applied in images in several ways such as gaussian blurring, laplasian blurring and motion blurring. Blurring removes details from an image by applying one of these filter kernels. The normalized version of the image contains less information compared to the original image. Although blurring does not remove all information from an image, it aims to prevent human users from recognizing the sensitive text or face.

Scrambling: Scrambling is another widely used technique for partially obfuscating images. The most common approach of image scrambling is applying

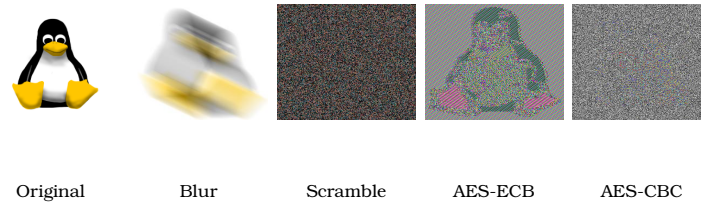


Figure 4.3: Different Image Obfuscation Techniques

discrete cosine transformation over image pixels. Other popular techniques include coefficient signs modification, cryptographic methods like *XOR* operation and coefficient permutation etc. However, scrambling performs quite heavy obfuscation of images and the information loss is high as shown in Figure 4.3. Applying scrambling to full images may affect the classification accuracy to large extent. So, we have excluded scrambling from the proposed image obfuscation techniques.

Encryption. Images can be encrypted by using both probabilistic and deterministic encryption schemes. Since image data has low commercial value compared to other types of data like banking data and confidential documents, conventional probabilistic key encryption techniques are not optimal solution [70]. Moreover, the probabilistic key encryption increases the complexity to operate over data and will result in huge utility loss in classification accuracy. Therefore, symmetric encryption techniques such as The Advanced Encryption Standard (AES) is more suitable for encrypting images. In this chapter, we have explored two modes of AES for encrypting images: Electronic Codebook (ECB) and Cipher Block Chaining (CBC). The ECB mode is deterministic in nature and reveals some information about the objects present in the images as shown in Figure 4.3 (some edges of the objects are visible). However, it successfully hides the background information (encrypted images shown in Figure 4.5c). On the other hand, AES-CBC is a probabilistic encryption technique due to the Initialization Vector (IV). Although

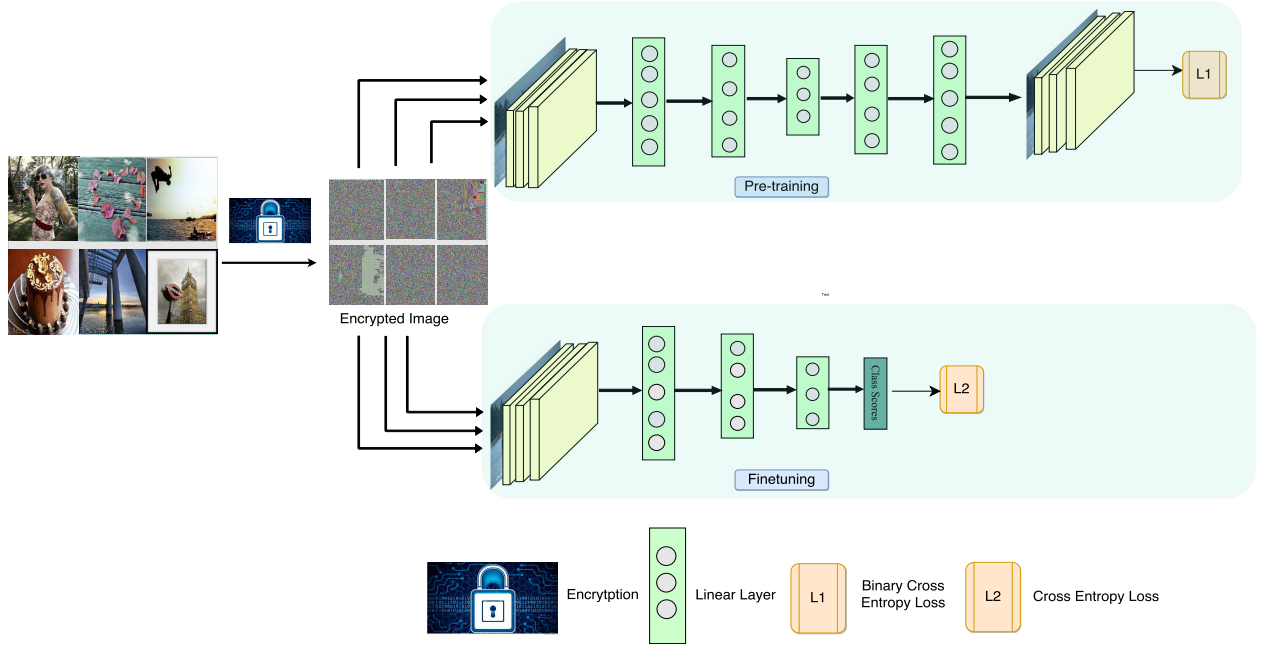


Figure 4.4: Proposed Architecture of Obfuscated Image Classification

it provides stronger security guarantee, it causes huge information loss as shown in Figure 4.3. Hence, we reported experiments only with AES encryption in ECB mode.

4.3.2 Feature Extraction

Feature extraction plays an important role in different computer vision related problems such as object detection [72], image classification [38], semantic segmentation [73] etc. where the input is typically image and applications try to retrieve output extracting different features. There are many feature extraction methods in the current literature such as the bag of words model, SIFT, etc. However, these techniques are not efficient on obfuscated images, as the features of obfuscated images are difficult to extract. Recently, neural networks with many layers, also known as the deep neural network is considered as one of the most

effective machine learning techniques for classifying high-dimensional data such as images or speech signals [74].

Deep Convolutional Neural Network. Convolutional neural network (CNN) is a relatively new and popular multi-layer neural network architecture have become very effective in classifying high dimensional data such as images and speech signals [75; 76; 77; 78]. The architecture of CNN is comprised of one or more convolutional layers and then followed by one or more fully connected layers. The network layers act as a detection filter and try to predict specific features or patterns present in the original data. CNN was first introduced to overcome the limitations of using a large number of parameters with the concept of three-dimensional neurons where the input data is considered as an image with a specific height, width and depth [38; 79].

The designed architecture of the network decides how different layers interact with one another. The performance of the model is greatly dependent on the architecture. CNN can be larger and deeper because of the presence of multiple convolutional layers and some other layers for extracting low-level complex features from input data [38]. A deep convolutional neural network (DCNN) is produced by increasing the depth of the network where all layers filtered by very small size kernel. The networks are comprised of a number of convolutional and sub-sampling layers with several fully connected layers. Each neuron in a convolutional layer works on a small connected region of the previous layer (e.g., neighbouring pixels in the first layer) over their full depth. In general, a number of convolutional neurons operate in the same region in the previous layer. As a result, multiple low-level features for the same region is extracted.

Limitations. DCNN is very popular for its high effectiveness on feature

extraction based on the nonlinear transformation on the input images. However, DCNN is specifically designed to handle large images where a huge amount of parameters needed to extract high-level features from the input image. In our model, image data is forwarded into the network, but those images are obfuscated before passing into the network. Due to obfuscation, input images lose a large amount of information and typical DCNN does not perform well on obfuscated (specifically on encrypted) inputs.

We experimented with some pre-trained DCNN models (i.e. VGGNet [80], AlexNet [38]) which are already trained on a huge amount data, on account of this, those models are biased on some particular inputs and try to extract features which are more related to those images. In our experiments although VGGNet can produce a significant accuracy result for original and blurred input images, produces random features on AES encrypted images without maintaining any particular pattern, therefore the accuracy is poor. Since VGGNet works well for original or blurred input, we keep it in our framework for those specific inputs however we need to modify our classification framework for AES encrypted input.

Unsupervised Pre-Training and Supervised Finetuning. To overcome the limitations of pre-trained DCNN regarding encrypted image classification, we shifted towards Deep Auto-encoder which is another efficient deep neural network architecture in an unsupervised setting. This neural network strategy not biased to any particular input (i.e., images) and it actually takes input as a representation.

In our work, we forward AES encrypted input as a representation which is an obfuscated version of an image. Deep Auto-encoder tries to produce a similar representation in each of its epochs and by this way it actually extracts features from the input pattern. Deep Auto-encoder does not provide a classifier directly

which only tries to produce features from the input. But our ultimate target is to get a classifier, therefore after getting trained network based on auto-encoder, we convert it into a typical Deep Neural Network supervised setting by removing its decoder part and attach another classifier layer into it with some particular neurons which can produce classification scores. We term this architecture as *Unsupervised Pre-Training and Supervised Finetuning*.

The architecture of deep autoencoder has a number of encoding and decoding layers. It receives an input representation x and forwards it into some specific operations of deterministic mapping $y = f_{\vartheta}(x) = Wx + b$ to map input into another hidden representation y which is parametrized by $\vartheta = \{W, b\}$. In each epoch of training, a particular representation $x^{(i)}$ is mapped into its corresponding representation $y^{(i)}$ and finally produces a new representation $z^{(i)}$ [81]. After each particular epoch, the model tries to update its parameters and the parameters are required to be optimized for the measurement of reconstruction error in each step of training.

4.4 Proposed Framework

The approach begins with user uploading obfuscated images to social networking sites to find a match with other users who have common visual interests.

4.4.1 Image Obfuscation

In our proposed framework images are obfuscated by Gaussian blurring and AES encryption in ECB mode. There are some privacy and utility trade-offs between choosing the modes for image obfuscation. From our experiments, we recommend

that if images are highly confidential then encryption should be used as a model of obfuscation; otherwise, we can use blurring to obfuscate the images. In our model, the users decide which method of the image obfuscation should be applied to the uploaded images. The equation for the gaussian mask is given below:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2 + y^2)}{2\sigma^2}}$$

The mask creates a kernel of gaussian low pass filter with size [x,y] and standard deviation σ . For example, if we set the parameter [x,y] as [5 5], it defines the extent to which the Gaussian filter is applied as a specification of the number of rows and columns and σ basically controls how heavy the kernel function is going to be; higher sigma values blur over a wider radius. To encrypt the images, we used AES-128 in ECB mode with key expansion based on the size of the images. The encryption methodology of AES involves multiple rounds of operation such as Byte substitution, shift rows, mix columns and add round keys. We do not discuss the AES architecture in detail in this chapter. The obfuscated images encrypted in AES-ECB mode is shown in Figure 4.5c.

4.4.2 Obfuscated Image Classification

In this step, we concentrate on deep neural networks for feature extraction purpose and finally, these extracted features pushed into the system. The training process of Deep Auto-encoder performed in two steps such as unsupervised pre-training and supervised fine-tuning. The proposed architecture for obfuscated image classification is shown in Figure 4.4. Before forwarding images into the network, we obfuscate the entire dataset and then split it into a training set and a test set. For the CIFAR-10 and MNIST datasets, images are designated as training

or test images by default but the attribute is missing in the MirFlicker-25k dataset. Therefore, we split the images based on random selection into training and test set.

Training. Designing a network architecture involves training process which learns weights for each layer of the network. These weights are termed as neural network parameters. The goal is to find the values of the defined parameters that minimize the classification error. Training is an iterative process. In each iteration of the training, network weights are updated on the model to reduce the error in the training set. This is done by stochastic gradient descent algorithm [80] which computes the gradient of the model’s error over each parameter and updates the parameter towards lower errors. The magnitude of the update is controlled by the learning rate hyper-parameter. For training our model, we set the learning rate of 10^{-3} with the learning rate decay of 10^{-7} , a momentum of 0.9, and weight decay of 5×10^{-4} . We fix the mini-batch size in our experiments as 16. In each mini-batch we forward 16 images into the network.

Pre-Training. In this stage, the network is trained in an unsupervised setting where network always tries to produce a replica of the input representation. The encoder part of the network compresses the original input and decoder part tries to decompress while the learning process is going. In the output layer of the unsupervised network, for N number of training input representing the data d_i produce output representation y_i ; binary cross entropy loss can be introduced for measuring the error of the reconstruction in the deep auto-encoder of input data d_i as,

$$\mathcal{L}_{BCE} = \frac{1}{N} \sum_{i=1}^N (d_i \log y_i + (1 - d_i) \log(1 - y_i)) \quad (4.1)$$

This loss function is differentiable with respect to its parameters and can be

optimized using the stochastic gradient descent. In this way forward and backward propagation performed into the network for certain epochs to update network parameters.

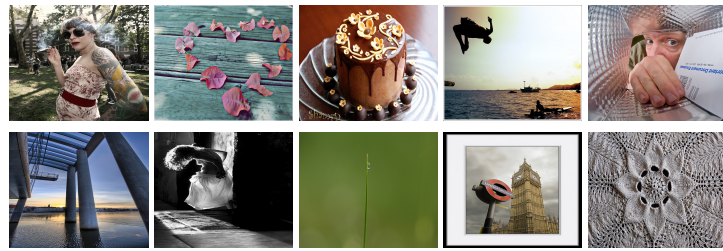
Fine-tuning. At this stage, we fine-tuned our network for classification in a fully supervised manner as we have class labels of input images. Here, we added an output layer removing the decoder part of the auto encoder and the output layer produces the confidence scores for each class present in the image. In this stage, cross entropy loss function introduced in the learning stage for measuring the classification loss of the network. For example, if there are N training images forwarded with $x^{(n)}$ representation and $y^{(n)}$ image label, the logistic loss function with weight parameter w can be defined as,

$$\mathcal{L}_{cls} = \frac{1}{N} \sum_{n=1}^N \log \left(1 + \exp \left(-y^{(n)} f \left(x^{(n)}; w \right) \right) \right) \quad (4.2)$$

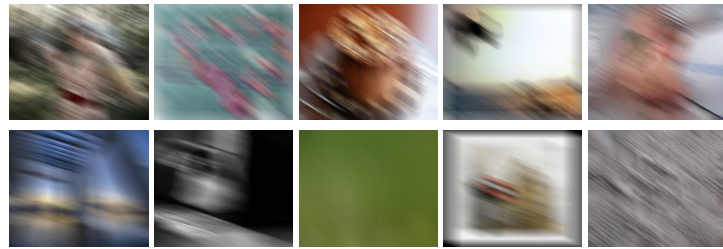
This loss function tries to optimize the learning efficiency of the network. Repeating the process, forward and backward propagation is performed in the network for a certain epochs to train the parameters of the final classifier.

4.4.3 User Profile Generation and Friend Recommendation

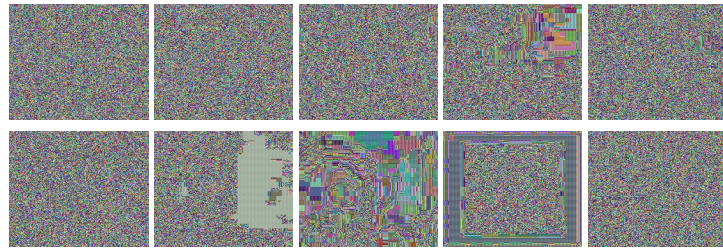
After feature extraction from obfuscated images, binary valued user image profiles are generated as shown in Table 4.2 normalizing the predicted scores for objects. This normalization helps to build efficient user profiles by focusing on the preferences of users and eliminating noise that has a negative impact on the friend recommendation result. Then all the user profiles are encrypted and uploaded to the cloud. The selection of encryption method to encrypt user profiles depends on the secure nearest neighbour search technique adopted by the service



(a) Original Images



(b) Images Obfuscated by Gaussian Blurring



(c) Images Obfuscated by AES-ECB Encryption

Figure 4.5: Image Obfuscation (MirFlickr-25k Dataset)

provider. Moreover, the obfuscated images are also encrypted with a higher level of encryption (e.g., probabilistic encryption) before uploading to the cloud. Due to encryption in service provider end, the cloud server cannot infer anything from the stored images and user profiles. When a target user, request for friend recommendation service, a secure nearest neighbour search operation is performed over encrypted user profiles using one of the existing approaches [7; 4; 9; 8; 6; 30] and the processed results are recommended to the user.

4.4.4 Security Analysis

The proposed image obfuscation techniques using gaussian blurring and encryption in AES-ECB mode is secure under the threat model we have considered. The obfuscated images successfully hide the background information. To justify the statement, we refer to Figure 4.5c, where some encrypted images are shown from the MirFlicker-25k Dataset. Here, the original images shown in Figure 4.5a contain background information which is completely hidden after the encryption in ECB mode (shown in Figure 4.5c). In particular, obfuscation of images causes lots of information loss to images. Therefore, it is not possible to get back to the original images without prior information of the model used during the obfuscation process [82].

If the images are encrypted by AES, it is not possible to decrypt the image without the key. Blurring is another widely use obfuscation technique and considered as secure [11]. Here, we assume that the service provider is not active adversary in the proposed model and the cloud does not have any information about the obfuscation model used. Notably, the obfuscated images are further encrypted with a higher level of encryption (probabilistic encryption) before storing to the cloud. This makes it difficult for the cloud server to infer anything from the stored images.

4.5 Experiment and Result

4.5.1 Implementation Details

The implementation of the prototype of the framework as described in Fig. 4.1 was done in two phases. In the first phase, we obfuscated the images using the

proposed blurring and AES encryption. For the blurring, we have used the Matlab image filtering toolbox [83]. For encryption, we have used a library which performs AES encryption of images in ECB mode [84]. In the second phase, we have used the autoencoder for extracting features from the images. For all the implementations, we have used a PC with Intel core i5-4590 CPU, 16GB RAM, 2TB HDD + 32GB SSD, Nvidia GTX 750Ti 2GB GPU.

4.5.2 Datasets

We simulated our experiment on the three popular datasets: MNIST, CIFAR-10 and Mirflickr-25k for friend recommendation.

MNIST. The MNIST dataset [85] contains 28×28 grayscale images of handwritten digits which were collected from US Census Bureau employees and high-school students. Each image contains one handwritten digit (i.e., an Arabic numeral between 0 to 9). The dataset contains two sets of images as training and test.

CIFAR-10. The CIFAR-10 dataset [38] consists of 32×32 color images where each image contains an object belonging to one of the pre-defined 10 classes (e.g., plane, car, etc.) or an animal (e.g., dog, cat, etc.). For unsupervised pre-training, we use all 50K training images without the label but when we fine tune our network, after getting a trained version on CIFAR-10, we use class labels to get our final classifier. The classes designed into the dataset structured as mutually exclusive where different classes are very identical to be distinguished.

MirFlickr-25k. The MirFlickr-25K [25] dataset contains 25,000 color images. The image sets were collected from the Flickr website and have been manually annotated for 20 class labels. The collection contains images under the Creative Common license that scored highest Flickr's *interestingness* score. These images

contain tags including objects and scene categories such as dog, portrait, sky, water or indoor etc [86]. In our experiment, we have selected 10 popular classes among 20 for measuring classification accuracy. However, MirFlicker-25k images are not split into training and test set in default, on account of this we randomly select 20K images from the dataset and forward into the network as training set and other 5K images as a test set.

4.5.3 Accuracy Analysis

The accuracy analysis of our proposed model is done in two phases. At first, we classify the images using our proposed architecture without obfuscation. Then in the second phase, we classify the obfuscated images. The accuracy comparison for MNIST, CIFAR-10 and Mirflickr datasets are shown respectively in Table 4.3, Table 4.4 and Table 4.5. The results demonstrate that the accuracy of blurring approach of obfuscation is higher than the encryption approach. As for encryption, the information loss is higher due to more privacy guarantee which reduces the accuracy.

The only existing work [71] which classify the encrypted images use the MNIST dataset. However, this dataset only consists of hand written digits and do not go with the perception of friend recommendation. We have run our model of image obfuscation in the MNIST dataset to justify the efficiency of the proposed model. The accuracy comparison is shown in Table 4.3. The reported accuracy for the existing approach is taken from the original article [71]. Our model has achieved an accuracy of 95.93% for blurring and 83.93% for encryption compared to the existing model [71] which has accuracy up to 79.83% for encryption.

From the experimental results, it can be noted that the classification accuracy

Table 4.3: Classification Accuracy on MNIST Dataset.

Method	0	1	2	3	4	5	6	7	8	9	Average
Non-Secure Version	99.697	99.585	99.451	99.491	99.517	99.645	99.465	99.467	99.185	99.527	99.503
Encryption (existing approach [71])	79.30	95.01	81.40	78.63	78.81	78.92	79.91	77.86	71.84	76.61	79.83
Blurring	96.54	98.62	97.12	94.44	95.09	95.08	98.25	94.92	95.43	93.81	95.93
Encryption	88.76	91.47	88.89	74.57	88.38	67.04	95.19	87.94	72.34	84.72	83.93

Table 4.4: Classification Accuracy on CIFAR-10 Dataset

Method	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck	Average
Non-Secure Version	96.6	94.317	92.7	94.683	93.217	96.017	96.167	96.650	96.033	95.12	95.15
Blurring	79.5	69.1	60.24	69.44	62.60	80.62	75.22	90.38	74.66	73.46	73.7
Encryption	68.347	69.654	58.998	62.689	64.069	70.483	72.982	67.040	81.428	69.600	68.529

Table 4.5: Classification Accuracy on Mirflickr-25k Dataset

Method	baby	bird	car	clouds	dog	female	flower	male	night	river	Average
Non-Secure Version	79.651	64.428	83.663	85.281	69.067	85.207	72.086	66.970	49.172	82.835	73.836
Blurring	65.3	52.3	47.31	56.64	49.1	66.23	61.18	76.5	60.3	59.45	59.43
Encryption	52.797	87.194	64.483	64.614	85.871	75.575	84.379	66.314	66.657	66.826	71.471

for CIFAR-10 and the MirFlickr-25K dataset is not as high as the MNIST dataset. The reason behind this is the images of CIFAR-10 and MirFlickr-25K datasets are more complex and cluttered than the simple digits of the MNIST dataset. However, increasing the dimension and complexity of images make the classification task more difficult. The experimental results clearly reflect the impact of image dimension and complexity on classification accuracy along with privacy and utility trade-off among the obfuscation methods.

4.6 Related Work

The problem addressed in the chapter is related to privacy preserving photo sharing, friend recommendation and secure nearest neighbour search. We already discussed some related works in Table 4.1. Below, we further discuss different characteristics of the most relevant approaches.

Privacy preserving photo sharing is introduced in some of the recent works. Zhan et. al [11] proposed a photo blurring technique to obscure the private parts of the images dividing the images into private and public parts. An efficient scrambling technique for private photo sharing is introduced by Yuan et. al [70]. Ryong et. al designed a secure photo sharing technique based on selective encryption where the private parts of images are encrypted and public parts are released.

There are some other related works regarding privacy preserving photo sharing. But they only focus on preserving the privacy of images rather than classification. To the best of our knowledge only existing work which addresses the privacy preserving image classification is done by Wang et. al [71]. They use the symmetric key encryption AES and DES to encrypt the images after that classify the encrypted images using multilayer neural network. There are also some related works regarding privacy preserving image classification [87; 88; 89]. However, these works focus on the secure model of classification rather than encrypted image classification.

Yuan et al. [4] proposed a privacy preserving image-centric friend recommendation system where they classify the plain images rather than encrypted ones. Their approach focused on discovering visual similarity in the images available in social media to facilitate friend discovery. For measuring the visual similarity

between the target user and the other users in the network, they used the bag-of-words model which is less accurate than the more recent deep neural network proposed in this work. After the generation of user profiles, the profiles were encrypted and stored in the public cloud upon which the similarity search operation was performed when friend recommendation request was generated.

Chapter 5

Conclusion

In this thesis, we have proposed different privacy-preserving techniques to perform nearest neighbour search over secured data in the outsourced environment. Our algorithms and methods are tested for real-world scenario and they provide provable security guarantee for data protection.

5.1 Summary

In chapter 2, we presented a framework for privacy preserving image-centric social discovery utilizing nearest neighbour search over the anonymized data. The proposed technique provides a solution to two challenges – storing images in the semi-honest cloud infrastructure and providing social discovery by the nearest neighbour search on anonymized data at the cloud server end. The designed framework provides security of shared images and enables efficient search over the anonymized data with utility preservation.

In chapter 3, we presented a framework for performing nearest neighbour search over encrypted data for providing image-centric social discovery service.

The proposed technique is practical and enables efficient search over the encrypted data in outsourced environment. The experimental results show that a competitive accuracy performance is achievable for a small run-time overhead in order to ensure the security of the sensitive data.

In chapter 4, we studied the problem of obfuscated image classification for privacy preserving image-centric friend recommendation utilizing secure nearest neighbour search. The proposed design obfuscate the content rich sensitive images using an appropriate technique and classify the obfuscated images to build user profiles. Then, nearest neighbour search is performed in an efficient way to generate friend recommendation.

To summarize, this thesis addresses the research problem of secure nearest neighbour search over securely stored data combining the field of security, image processing and recommendation system.

5.2 Future Work

In this thesis, we have utilized an efficient indexing technique LSH with minhashing to search over secured data. Some other tree based indexing techniques can be utilized such as B^+ trees, R-trees, etc to improve the performance [90; 91; 92]. The improvement in indexing techniques will improve the computation time to search over data.

Additionally, nearest neighbour search can be utilized for genomic research and health related applications. The recent advancement of genomic technologies is encouraging the research communities to perform analysis on genomic data efficiently. Genomic research has entered into the mainstream of medicine leading

to the share of genomic data among institutions for collaborations [93]. Nevertheless, genetic sequencing for personalized medicine has been broadly applied to the discovery and study of disease [94]. In this model, genetic testings are employed for selecting effective and optimal treatment based on the context of a patient's genetic information, such as DNA sequence, molecular, and cellular analysis [95].

Similar Patient Query (SPQ) based on similarity matching on the sequenced genomes of patients is an interesting research problem. For instance, given the DNA sequence of a patient, a doctor might want to know the top-k most similar other patients in the hospital database or even in the databases of other hospitals to know their treatments and outcomes, in order to decide what can be the best drug regimen for his own patient [96]. To perform SPQ on genomic sequence NNS with different similarity metrics can be applied. Hamming distance determines the number of mismatched nucleotides between two sequences and it is very useful for SPQ [97]. Executing efficient SPQs will lead to significant improvement in disease diagnosis and early detection of diseases. These are some of the future directions of this thesis.

Bibliography

- [1] Charith Perera, Chi Harold Liu, Srimal Jayawardena, and Min Chen. A survey on internet of things from industrial market perspective. *IEEE Access*, 2:1660–1679, 2014.
- [2] Camila Souza Araujo, Damilton Correa, Luiz Paulo, Ana Paula Couto da Silva, Raquel Oliveira Prates, and Wagner Meira. It is not just a picture: revealing some user practices in instagram. In *9th IEEE Latin American Web Congress (LA-WEB)*, pages 19–23, 2014.
- [3] <http://www.wired.com/2014/06/facebook-instagram/>. Online; accessed 31 July 2017.
- [4] Xingliang Yuan, Xinyu Wang, Cong Wang, Anna Squicciarini, and Kui Ren. Enabling privacy-preserving image-centric social discovery. In *34th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 198–207, 2014.
- [5] Megha Pandey and Alex Chia Yong Sang. Capturing the visual language of social media. In *IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2015.
- [6] Kazi Wasif Ahmed, Mohammad Zahidul Hasan, and Noman Mohammed.

- Image-centric social discovery using neural network under anonymity constraint. In *IEEE International Conference on Cloud Engineering (IC2E)*, pages 238–244, 2017.
- [7] Mehmet Kuzu, Mohammad Saiful Islam, and Murat Kantarcioglu. Efficient similarity search over encrypted data. In *28th IEEE International Conference on Data Engineering (ICDE)*, pages 1156–1167, 2012.
- [8] Yousef Elmehdwi, Bharath K Samanthula, and Wei Jiang. Secure k-nearest neighbor query over encrypted data in outsourced environments. In *30th IEEE International Conference on Data Engineering*, pages 664–675, 2014.
- [9] Boyang Wang, Yantian Hou, and Ming Li. Practical and secure nearest neighbor search on encrypted large-scale data. In *INFOCOM- The 35th IEEE Annual International Conference on Computer Communications*, pages 1–9, 2016.
- [10] Kazi Wasif Ahmed, Momin Al Aziz, Dima Alhadidi, and Noman Mohammed. Secure nearest neighbour search over encrypted data using intel sgx. *Future Generation Computer Systems*, 2017 (Submitted).
- [11] Lan Zhang, Taeho Jung, Cihang Liu, Xuan Ding, Xiang-Yang Li, and Yunhao Liu. Pop: Privacy-preserving outsourced photo sharing and searching for mobile devices. In *35th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 308–317, 2015.
- [12] Kazi Wasif Ahmed, Omit Chanda, Noman Mohammed, and Yang Wang. Obfuscated image classification for secure image-centric friend recommendation. *Sustainable Cities and Society*, 2017.

-
- [13] Bernardo Ferreira, João Rodrigues, João Leitão, and Henrique Domingos. Privacy-preserving content-based image retrieval in the cloud. In *34th IEEE Symposium on Reliable Distributed Systems (SRDS)*, pages 11–20, 2015.
- [14] Pierangela Samarati. Protecting respondents identities in microdata release. *IEEE transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.
- [15] Charu C Aggarwal. On k-anonymity and the curse of dimensionality. In *Proceedings of the 31st international conference on Very large data bases*, pages 901–909. VLDB Endowment, 2005.
- [16] Baichuan Zhang, Noman Mohammed, Vachik S Dave, and Mohammad Al Hasan. Feature selection for classification under anonymity constraint. *Transactions on Data Privacy*, 2017.
- [17] D Li, Q Lv, H Xia, L Shang, T Lu, and N Gu Pistis. A privacy-preserving content recommender system for online social communities. *WI-IAT. IEEE*, 2011.
- [18] Takeaki Uno, Masashi Kiyomi, and Hiroki Arimura. Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In *FIMI*, volume 126, 2004.
- [19] A Rajaraman and JD Ullman. Finding similar items. *The book Mining of Massive Datasets.*, pages 71–127, 2010.
- [20] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *47th Annual IEEE Sym-*

- posium on Foundations of Computer Science, 2006, FOCS'06, pages 459–468, 2006.
- [21] Cheng-Hao Chu, Wan-Chuen Wu, Cheng-Chi Wang, Tzung-Shi Chen, and Jen-Jee Chen. Friend recommendation for location-based mobile social networks. In *Seventh IEEE International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, pages 365–370, 2013.
- [22] Andrea Vedaldi and Karel Lenc. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 689–692, 2015.
- [23] <http://research.nii.ac.jp/~uno/code/lcm.html>. Online; accessed 21 June 2017.
- [24] <https://github.com/tdebatty/java-LSH/>. Online; accessed 21 June 2017.
- [25] Mark J Huiskes and Michael S Lew. The mir flickr retrieval evaluation. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, pages 39–43, 2008.
- [26] Hongjuan Li, Xiuzhen Cheng, Keqiu Li, and Zhi Tian. Efficient customized privacy preserving friend discovery in mobile social networks. In *35th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 225–234, 2015.
- [27] Yong Wang, Ting-Ting Zhang, Hong-Zong Li, Long-Ping He, and Jing Peng. Efficient privacy preserving matchmaking for mobile social networking against malicious users. In *11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 609–615, 2012.

-
- [28] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear complexity. In *Financial Cryptography and Data Security*, pages 143–159. Springer, 2010.
- [29] Wei Dong, Vacha Dave, Lili Qiu, and Yin Zhang. Secure friend discovery in mobile social networks. In *IEEE Proceedings in INFOCOM*, pages 1647–1655, 2011.
- [30] Mikhail Strizhov. Towards a practical and efficient search over encrypted data in the cloud. In *IEEE International Conference on Cloud Engineering (IC2E)*, pages 496–498, 2015.
- [31] Jiadi Yu, Peng Lu, Yanmin Zhu, Guangtao Xue, and Minglu Li. Toward secure multikeyword top-k retrieval over encrypted cloud data. *IEEE transactions on dependable and secure computing*, 10(4):239–250, 2013.
- [32] Richard Chow, Manas A Pathak, and Cong Wang. A practical system for privacy-preserving collaborative filtering. In *12th IEEE International Conference on Data Mining Workshops*, pages 547–554, 2012.
- [33] Fida Kamal Dankar and Khaled El Emam. Practicing differential privacy in health care: A review. *Transactions on Data Privacy*, 6(1):35–67, 2013.
- [34] Yasser Jafer, Stan Matwin, and Marina Sokolova. Using feature selection to improve the utility of differentially private data publishing. *Procedia Computer Science*, 37:511–516, 2014.
- [35] Baichuan Zhang, Vachik Dave, and Mohammad Al Hasan. Feature selection for classification under anonymity constraint. *arXiv preprint arXiv:1512.07158*, 2015.

- [36] Bin Yao, Feifei Li, and Xiaokui Xiao. Secure nearest neighbor revisited. In *29th IEEE International Conference on Data Engineering (ICDE)*, pages 733–744, 2013.
- [37] Bharath K Samanthula, Yousef Elmehdwi, and Wei Jiang. K-nearest neighbor classification over semantically secure encrypted relational data. *IEEE transactions on Knowledge and data engineering*, 27(5):1261–1273, 2015.
- [38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [39] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R Savagaonkar. Innovative instructions and software model for isolated execution. In *HASP@ ISCA*, page 10, 2013.
- [40] Bernardo Ferreira, Joao Rodrigues, Joao Leita0, and Henrique Domingos. Privacy-preserving content-based image retrieval in the cloud. In *34th IEEE Symposium on Reliable Distributed Systems (SRDS)*, pages 11–20, 2015.
- [41] Florian Kerschbaum and Ahmad-Reza Sadeghi. Hardidx: Practical and secure index with sgx. In *Data and Applications Security and Privacy XXXI: 31st Annual IFIP WG 11.3 Conference, DBSec 2017, Philadelphia, PA, USA, July 19-21, 2017, Proceedings*, volume 10359, page 386. Springer, 2017.
- [42] Felix Schuster, Manuel Costa, Cédric Fournet, Christos Gkantsidis, Marcus Peinado, Gloria Mainar-Ruiz, and Mark Russinovich. Vc3: trustworthy data

- analytics in the cloud using sgx. In *2015 IEEE Symposium on Security and Privacy*, pages 38–54, 2015.
- [43] Ben A Fisch, Dhinakaran Vinayagamurthy, Dan Boneh, and Sergey Gorbunov. Iron: Functional encryption using intel sgx.
- [44] Cristian Borcea, Xiaoning Ding, Narain Gehani, Reza Curtmola, Mohammad A Khan, and Hillol Debnath. Avatar: Mobile distributed computing in the cloud. In *3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pages 151–156, 2015.
- [45] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *IEEE Proceedings in Infocom*, pages 1–9, 2010.
- [46] Yuanzhong Xu, Weidong Cui, and Marcus Peinado. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *IEEE Symposium on Security and Privacy (SP)*, pages 640–656, 2015.
- [47] Victor Costan and Srinivas Devadas. Intel sgx explained. Technical report, Cryptology ePrint Archive, Report 2016/086, 2016. <https://eprint.iacr.org/2016/086>.
- [48] Andrew Baumann, Marcus Peinado, and Galen Hunt. Shielding applications from an untrusted cloud with haven. *ACM Transactions on Computer Systems (TOCS)*, 33(3):8, 2015.
- [49] Ing Christof Paar and Ing Jan Pelzl. More about block ciphers. In *Understanding Cryptography*, pages 123–148. Springer, 2010.

- [50] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptology-EUROCRYPT*, pages 223–238. Springer, 1999.
- [51] Ke Li, Weiming Zhang, Ce Yang, and Nenghai Yu. Security analysis on one-to-many order preserving encryption-based cloud data search. *IEEE Transactions on Information Forensics and Security*, 10(9):1918–1926, 2015.
- [52] Ning Cao, Cong Wang, Ming Li, Kui Ren, and Wenjing Lou. Privacy-preserving multi-keyword ranked search over encrypted cloud data. *IEEE Transactions on parallel and distributed systems*, 25(1):222–233, 2014.
- [53] Yaqian Gao, Meixia Miao, Jianfeng Wang, and Xiaofeng Chen. Secure approximate nearest neighbor search over encrypted data. In *Ninth IEEE International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA)*, pages 578–583, 2014.
- [54] Boyang Wang, Yantian Hou, Ming Li, Haitao Wang, and Hui Li. Maple: scalable multi-dimensional range search over encrypted cloud data with tree-based index. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pages 111–122, 2014.
- [55] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 248–255, 2009.
- [56] Richard Jizba. Measuring search effectiveness. *Creighton University Health Sciences Library and Learning Resources Center*, 2000.

- [57] Mark Shaneck, Yongdae Kim, and Vipin Kumar. Privacy preserving nearest neighbor search. In *Machine Learning in Cyber Trust*, pages 247–276. Springer, 2009.
- [58] Ming Li, Shucheng Yu, Ning Cao, and Wenjing Lou. Authorized private keyword search over encrypted data in cloud computing. In *31st IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 383–392, 2011.
- [59] Zhangjie Fu, Xingming Sun, Nigel Linge, and Lu Zhou. Achieving effective cloud search services: multi-keyword ranked search over encrypted cloud data supporting synonym query. *IEEE Transactions on Consumer Electronics*, 60(1):164–172, 2014.
- [60] Zhihua Xia, Xinhui Wang, Xingming Sun, and Qian Wang. A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. *IEEE Transactions on Parallel and Distributed Systems*, pages 340–352, 2016.
- [61] Yu Hua, Bin Xiao, and Xue Liu. Nest: Locality-aware approximate query service for cloud computing. In *IEEE Proceedings in INFOCOM*, pages 1303–1311, 2013.
- [62] Sumeet Bajaj and Radu Sion. Trusteddb: A trusted hardware-based database with privacy and data confidentiality. *IEEE Transactions on Knowledge and Data Engineering*, 26(3):752–765, 2014.
- [63] Adeyinka Tella. *Social Media Strategies for Dynamic Library Service Development*. IGI Global, 2014.

- [64] Laura Alvarez, Katharina Borsi, and Lucelia Rodrigues. The role of social network analysis on participation and placemaking. *Sustainable Cities and Society*, 28:118–126, 2017.
- [65] Moo-Ryong Ra, Ramesh Govindan, and Antonio Ortega. P3: Toward privacy-preserving photo sharing. In *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 515–528, 2013.
- [66] Jilin Chen, Werner Geyer, Casey Dugan, Michael Muller, and Ido Guy. Make new friends, but keep the old: recommending people on social networking sites. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 201–210, 2009.
- [67] Dave Lewis. icloud data breach: Hacking and celebrity photos. *Forbes*, 2014. Online; accessed 4 April 2017.
- [68] Youtube official blog. face blurring: when footage requires anonymity. <https://youtube.googleblog.com/2012/07/face-blurring-when-footage-requires.html>, 2012. Online; accessed 13 April 2017.
- [69] Richard McPherson, Reza Shokri, and Vitaly Shmatikov. Defeating image obfuscation with deep learning. *arXiv preprint arXiv:1609.00408*, 2016.
- [70] Lin Yuan, Pavel Korshunov, and Touradj Ebrahimi. Privacy-preserving photo sharing based on a secure jpeg. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPs)*, pages 185–190, 2015.
- [71] Weiru Wang, Chi-Man Vong, Yilong Yang, and Pak-Kin Wong. Encrypted im-

- age classification based on multilayer extreme learning machine. *Multidimensional Systems and Signal Processing*, pages 1–15, 2016.
- [72] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.
- [73] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.
- [74] Samer Hijazi, Rishi Kumar, and Chris Rowen. Using convolutional neural networks for image recognition, 2015.
- [75] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.
- [76] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.
- [77] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

-
- [78] Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*, volume 3, pages 958–962, 2003.
- [79] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.
- [80] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [81] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.
- [82] José Ramón Padilla-López, Alexandros Andre Chaaraoui, and Francisco Flórez-Revuelta. Visual privacy protection methods: A survey. *Expert Systems with Applications*, 42(9):4177–4195, 2015.
- [83] <https://www.mathworks.com/help/images/ref/fspecial.html>. Online; accessed 20 April 2017.
- [84] <https://github.com/ralphleon/JImageEncryptor>. Online; accessed 31 July 2017.
- [85] Yann LeCun. The mnist database of handwritten digits. <http://yann.lecun.com/exdb/mnist/>, 1998.
- [86] Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Multimodal

- semi-supervised learning for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 902–909, 2010.
- [87] Zhan Qin, Jingbo Yan, Kui Ren, Chang Wen Chen, and Cong Wang. Secsift: Secure image sift feature extraction in cloud computing. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 12(4s):65, 2016.
- [88] Zhan Qin, Jingbo Yan, Kui Ren, Chang Wen Chen, and Cong Wang. Towards efficient privacy-preserving image feature extraction in cloud computing. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 497–506, 2014.
- [89] Zhan Qin, Jingbo Yan, and Kui Ren. Private image computation: the case of cloud based privacy-preserving sift. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 179–180, 2014.
- [90] Hosagrahar V Jagadish, Beng Chin Ooi, Kian-Lee Tan, Cui Yu, and Rui Zhang. idistance: An adaptive b+-tree based indexing method for nearest neighbor search. *ACM Transactions on Database Systems (TODS)*, 30(2):364–397, 2005.
- [91] King Lum Cheung and Ada Wai-Chee Fu. Enhanced nearest neighbour search on the r-tree. *ACM SIGMOD Record*, 27(3):16–21, 1998.
- [92] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.
- [93] Seungmin Kang, Khin Mi Mi Aung, and Bharadwaj Veeravalli. Towards secure and fast mapping of genomic sequences on public clouds. In *Proceedings of*

- the 4th ACM International Workshop on Security in Cloud Computing*, pages 59–66, 2016.
- [94] Edward D Esplin, Ling Oei, and Michael P Snyder. Personalized sequencing and the future of medicine: discovery, diagnosis and defeat of disease. *Pharmacogenomics*, 15(14):1771–1790, 2014.
- [95] Bing Wang, Wei Song, Wenjing Lou, and Y Thomas Hou. Privacy-preserving pattern matching over encrypted genetic data in cloud computing. In *INFOCOM-IEEE Conference on Computer Communications*, pages 1–9, 2017.
- [96] Xiao Shaun Wang, Yan Huang, Yongan Zhao, Haixu Tang, XiaoFeng Wang, and Diyue Bu. Efficient genome-wide, privacy-preserving similar patient query based on private edit distance. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 492–503, 2015.
- [97] Astrid Rheinländer, Martin Knobloch, Nicky Hochmuth, and Ulf Leser. Prefix tree indexing for similarity search and similarity joins on genomic data. In *International Conference on Scientific and Statistical Database Management*, pages 519–536. Springer, 2010.