Metadata filtering for secure and high performance user identification

Marcus Rodan
dat13mro@student.lu.se
Niklas Jönsson
dat13njo@student.lu.se

Department of Electrical and Information Technology
Lund University

Supervisor: Christian Gehrmann

Examiner: Thomas Johansson

May 31, 2018

© 2018
Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

The aim of this thesis is to investigate the possibility of enabling biometric identification given a large number of enrolled users. This area is relatively unexplored compared to biometric authentication where a fingerprint template is compared against just a few number of templates. The selected approach is to use auxiliary information available during an identification. This information will be used in a filtering process, only retaining probable candidates for further consideration. The auxiliary information as used in this context is known as metadata and some examples includes location, name and age. The lack of existing suitable metadata data sets led to the development of a simulator able to simulate different metadata types. The goal of the simulator is to mimic realistic user behaviour as well as a growing database of enrolled users containing simulated metadata records. The simulator enabled rigorous testing of different filtering techniques in order to evaluate the feasibility of the different techniques. The results show that it is possible to perform a secure, reliable and high performance user identification using a combination of two or more metadata types. The security evaluation also indicates that it is possible to enhance the security of the system by using different strategies protecting against possible attacks. The conducted work resulted in a proof-of-concept implementation supporting further work within the newly explored area.

Keywords: biometric identification, metadata filtering, metadata properties, metadata simulation, privacy protection

Table of Contents

List of Figures

List of Tables

Glossary

BTS	Biometric Trusted Service.
EER	Equal Error Rate.
FAR	False Acceptance Rate.
FRR	False Rejection Rate.
JOSE	Javascript Object Signing and Encryption.
JSON	JavaScript Object Notation.
KD-trees	K-dimensional trees.
LSH	Locality Sensitive Hashing.
PIN	Personal Identification Number.
SCB	Statistiska centralbyrån.
SVM	Support Vector Machine.
TLS	Transport Layer Security.

Chapter 1

Introduction

It is common in today's technological society to store sensitive personal information in the cloud. This information should only be available to authorized users and it is therefore crucial to perform a reliable identification and authentication.

The authentication has traditionally been carried out using a password, PIN and physical cards since these methods are easily understood. However these methods have some drawbacks such as low password complexity as people tend to select easily remembered passwords [1]. Another drawback is that it is common to only have a few selected passwords used for all services and this poses a security risk [2]. A third drawback with the traditional methods are that they rely on the user to remember a secret or carry a physical card that will be used during authentication. This can be both tedious and aggravating as passwords and PINs can be forgotten and physical cards can be lost.

In order to address these drawbacks, alternative authentication methods has emerged and as biometric features are highly individual and always accessible, these features could be used for safe and effortless authentication and identification [3].

1.1 Background

There exists many different biometric features that are good candidates for reliable authentication and identification and it is also possible to use a combination of several features in order to further improve the accuracy [4, 5, 6, 7]. This thesis will be focused towards using fingerprints as the biometric source but it should be noted that it is believed that the techniques developed are not dependant on the biometric source and will therefor work equally well for any biometric source.

Since the terms authentication and identification can be mistakenly interchanged, it is important to understand that authentication is the process of proving the identity of a subject, while identification is the process of binding an identity to a subject [8]. This means that authentication can be divided into two major steps. The first step requires the subject to state an identity, such as a username, and in the second step provide a proof of the claimed identity. When performing identification the user should only be required to present some information binding an identity to the user and it is the system's task to find the corresponding identity. This thesis will be devoted to identification using biometric features and available metadata.

In the last few years fingerprint scanners has been integrated into mobile phones, introducing fingerprint scanners to a broader user base. The primary reason for integrating these scanners has been to perform an effortless identification of users trying to access the device by comparing a template of a scanned fingerprint against some locally stored templates known to belong to potentially different authorized users [8].

One drawback with this approach is that the user has to go through a tedious registration procedure for each device that the user would like to access. This approach also places a burden on the service provider since each device will need to implement matching algorithms that are able to compare a template of a scanned fingerprint against locally stored templates. Taking these limitations into consideration it would be desirable if the burden could be placed on a trusted central matching service instead. Henceforward the trusted central matching service will be known as the biometric trusted service (BTS). This approach would remove the tedious process of having to enroll on each device. It would also remove the burden of having to implement an advanced algorithm from the different service providers since the matching will be done by the central matching service. Figure 1.1 illustrates the architecture of the suggested approach. The user communicates with the application and the application communicates with the matching back end service during an identification.

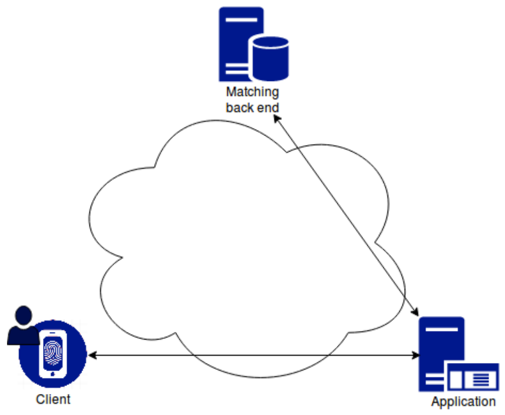


Figure 1.1: Central matching service approach

1.2 Problem description

Looking at figure 1.1, one question that arises is: how should a potential template match be found by the BTS? One naive approach would be to compare the template associated with the query against each stored template in the BTS database, however this puts harsh constraints on exactness for the algorithm used by the BTS to compare two templates [8].

There exists several different limitations when comparing two scanned fingerprints against each other. One limitation is that typically the whole fingerprint is not scanned, instead only a segment of the fingerprint is scanned and then transformed into a template that is then used during comparison [9]. During comparison a slight variation is allowed since different readings may result in slightly different templates. Because of these limitations there exists a small probability that two non-matching templates will be deemed to match by the matching algorithm. From probability theory it follows that the probability of a false match to occur grows as the total number of comparisons increases. This implies that in order to maintain a reliable identification the number of comparisons that are carried out must be limited [10]. Since one of the implications of a BTS is a large number of enrolled users, it implies that it will not be feasible to compare a query template against all stored templates in the BTS in order to provide a reliable identification. This means that the number of comparisons needs to be reduced.

1.3 Project aims

The main goal of this thesis is to investigate different methods and techniques for reducing the number of comparisons needed when searching for a matching template. Since the system will have large number of enrolled users, some kind of selective preprocessing needs to be carried out before the existing matching algorithm is executed [10], as seen in figure 1.2. This thesis will explore the possibility

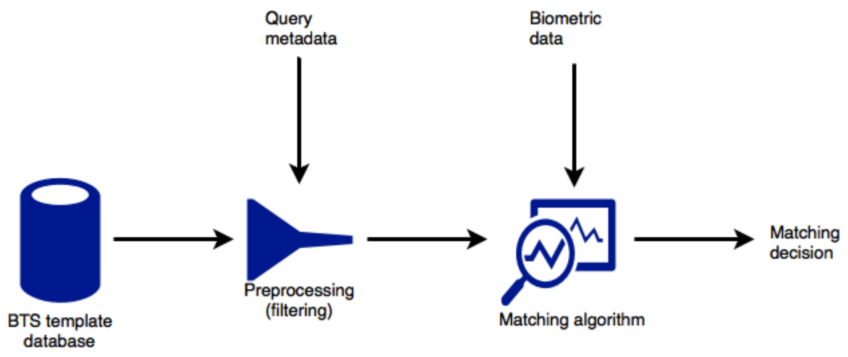


Figure 1.2: Overview of filtering approach

of using metadata associated with the biometric data in order to carry out filtering in the preprocessing stage. If this can be done efficiently then it will be feasible to perform a reliable identification even with a large number of enrolled users.

The aim is to answer the following questions since they are relevant for the given problem:

- Which algorithms are suitable to use for metadata filtering?
- How does combinations of different metadata effect the efficiency of the filtering algorithms?
- Is it possible to collect metadata after successful identification in order to further improve the accuracy of the filtering algorithm?
- Which metadata types are suitable to incorporate in a filter process?
- How can the efficiency be measured in order to evaluate the performance of the filtering?
- Can algorithms requiring recorded experience be used even though no such experience will be given explicitly?
- Is there any possibility that the application can assist the BTS with the filtering process?

At the time of writing this thesis, this area of computer science is rather unexplored compared to other branches. This thesis work will contribute by showing another real world application of how metadata filtering can be used in new ways in order to improve an existing algorithm by preprocessing the input data.

1.4 Thesis outline

Chapter 1. Introduction This chapter will introduce the reader to the the problem of performing biometric identification with a large number of enrolled users.

Chapter 2. Terminology In this chapter all needed area related terms are explained.

Chapter 3. Generic Filtering A detailed description will be provided of how filtering can be applied as a preprocessing stage in order to improve the accuracy and performance of existing identification algorithms.

Chapter 4. Theory A summary of state of the art algorithms and an evaluation of their applicability for filtering with metadata.

Chapter 5. Evaluation Framework Contains a detailed description of the selected evaluation framework. This chapter describes the simulator together with simulated metadata types.

Chapter 6. Filter design & implementations Discusses different filtering strategies together with their performance. Both unimodal as well as multimodal filtering techniques are considered.

Chapter 7. Security evaluation A security evaluation of the developed algorithms and of the system architecture. Possible attacks and mitigation strategies are presented.

Chapter 8. Location privacy An elaborate discussion about location privacy is given together with potential mitigation strategies in order to protect against privacy breaches originating from different adversaries.

Chapter 9. Discussion Discusses the performance of the filtering techniques in order to judge their applicability. This chapter also discusses the developed simulator as well as the security of the implementation.

Chapter 10. Future Work A list of possible additions and some advised work that can be done to prepare for future deployment of the discussed system.

Chapter 11. Conclusions Contains final thoughts and conclusion derived from the entire thesis work.

Chapter 2

Terminology

This chapter will introduce area related terminology in order to provide a strong foundation for further discussions about different filtering algorithms. The introduced terminology will be used throughout the thesis.

2.1 Fingerprint template

A fingerprint template is a data representation of a physical fingerprint [9]. The fingerprint is scanned and fingerprint features are extracted from the scanned image. These features are then transformed into a data representation of the image. This data representation is called a fingerprint template and it is these templates that are stored in the BTS template database, not the actual images of the fingerprints.

2.2 Enrolled users

An enrolled user is a user who is registered in the BTS database containing fingerprint templates. All enrolled users have gone through an enrollment phase where fingerprint scans and metadata collection have been carried out [8].

2.3 Biometric Trusted Service (BTS)

The BTS is a service developed by Fingerprint Cards AB. It is the trusted central matching service that performs the matching of fingerprint templates [11]. The BTS has a database of all enrolled fingerprint templates together with collected metadata. The database is used to find probable candidates during an identification.

The BTS communicates with different applications and provide a trusted environment where reliable identification will be carried out.

2.4 Query template

A query template is the fingerprint template associated with an identification request. This template is used by the matching algorithm of the BTS [11]. Looking back at figure 1.1, the query template, together with associated metadata, will be sent by the client to the application which then relays this data to the BTS.

2.5 False Acceptance Rate, False Rejection Rate & Equal Error Rate

The two terms False Acceptance Rate (FAR) and False Rejection Rate (FRR) are terms widely accepted in academia and industry when evaluating biometric identification solutions and implementations. The first term is a measurement of how often an access is allowed when it should have been denied [12]. This means that if a system should provide high security it should have a low FAR. The second term is a measurement of how often a valid access is denied [12]. This means that a system with a high FRR will more often deny genuine users access. The ideal case would be a FAR of 0 and a FRR of 0. This however, is not possible to achieve for many systems and typically the case is that a low FAR will cause a high FRR. In the same way a low FRR will typically cause a high FAR. This behaviour is illustrated in figure 2.1. It is also common to come across the term Equal Error Rate (EER) which refers to the rate when the FRR and FAR are equal.

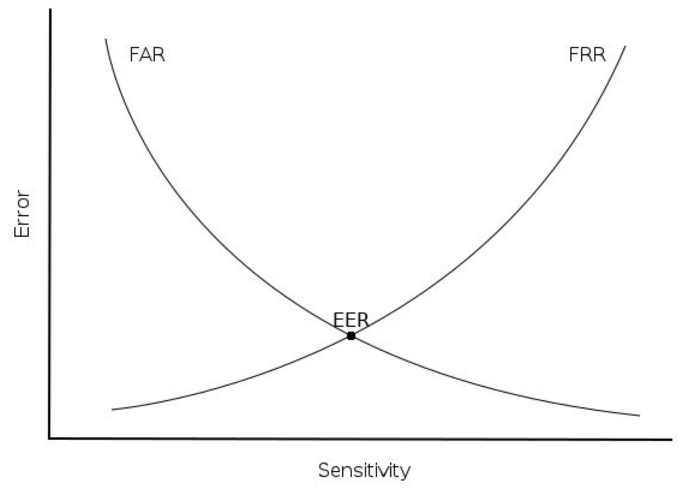


Figure 2.1: Typical characteristics of FAR and FRR

2.6 Data set

Generally a data set is defined to be a collection of data. The term data itself can be of any kind, such as an image or a string. This means that one example of a data set can be a collection of strings. This means that the general definition of a data set is unstructured.

However for this thesis, the term data set will be redefined to be a collection of entries. The entries will have a fixed number of possible attributes and each entry will have the same set of possible attributes. Since it is common in the real world that some data is missing, entries will be allowed to have empty attributes. One example of a data set following the given definition is shown in table **??**. This particular example has one missing attribute for one entry.

Table 2.1: An example of a data set

Name	**Age**	**Address**
Marcus	23	Kings' Landing
Niklas	25	Winterfell
Sofia	22	

2.7 Test set and training set

When dealing with data collections and algorithms it is common to encounter the two terms test set and training set. Using the definition of a data set, a training set can be defined as a data set containing examples of data that will help during the learning phase of an algorithm [13]. A test set can be defined as a data set that will be used in order to evaluate the performance of an algorithm. Table 2.2 show an example of a training set. This set is given to an algorithm with the hope that the algorithm will find the inferred pattern. A test set for this case would look similar to the showed training set with just different examples.

2.8 Metadata

Metadata as used in the context of this thesis is defined as any data associated with some biometric data, excluding the biometric data itself [14]. E.g. age, device

Table 2.2: An example of a training set

x	**y**	**output**
1.0	2.0	3.0
5.0	3.0	8.0
2.0	8.0	10.0

ID or geographical position. Metadata can be linked to different entities in the system. For example, the different enrolled users will have metadata stored about them. There will also be metadata linked to each identification request [11].

2.9 Trustworthiness of metadata

Since metadata will be collected during different circumstances it is important to consider how trustworthy the collected data is. It can be imagined that different circumstances will provide different levels of trustworthiness and therefore there exists a need to model the trustworthiness of collected data. In order to model the trustworthiness, a metadata record a of some metadata type can be given a score between zero and one where zero corresponds to non existent trustworthiness and one corresponds to maximal level of trustworthiness [11, 14].

Mathematically speaking one score T_a is assigned for each metadata record a such that $T_a \in [0,1]$. It will be assumed that this score is always available for any metadata record a.

2.10 Reliability of metadata

The reliability of metadata is a measurement of the validity of the data. Even though the data may be completely trustworthy, there might be uncertainty in the metadata, caused by faulty sensors or error prone measurements etc. Therefore any metadata can be assigned a reliability score between zero and one, where zero is not reliable at all and one is completely reliable metadata [11, 14].

Mathematically speaking one score R_a is assigned for each metadata record a such that $R_a \in [0,1]$. It will be assumed that this score is always available for any metadata record a.

2.11 Identification request

An identification request is the request that a user sends to an application in order to identify him or herself. This request contains the query template together with collected metadata.

2.12 Candidate set & filtered candidate set

The candidate set is the set of all enrolled user's fingerprint templates and their recorded metadata. This set is used as input into the filtering algorithm in the preprocessing stage and the output is the filtered candidate set. The goal is to reduce the size of the filtered candidate set as much as possible in order to achieve efficient and reliable identification [10]. As it is the filtered candidate set that is used as input to the matching algorithm, it is very important that the filtering algorithm does not filter out the template that matches the query template, as this would increase the FRR.

2.13 Application

Looking at figure 1.1, an application is a service that the user can register to. The application will communicate with the BTS in order to achieve a final identification decision. An example of an application can be a website or a physical store.

The user must be enrolled in the BTS database before a registration to an application can be done. However, it may be possible for the application to perform the BTS enrollment on behalf of the user.

Chapter 3
Generic Filtering

As concluded earlier, it is difficult to perform a reliable identification with a large number of enrolled users. The proposed solution is to reduce the number of candidates before a more detailed matching algorithm is executed [10, 11]. This reduction will be done by utilizing metadata stored about different users as well as metadata collected during an identification.

This chapter will be based upon an internal Fingerprints report proposing a central matching service [11]. It will present how filtering can be integrated into existing software, making it possible to perform reliable identification even with a large number of enrolled users. All of the different stages needed to actualize the solution will be described and an actual usage scenario will be presented.

The previously described approach is not limited to the metadata and biometric source given in the example. It can be applied to any identification using metadata and some biometric source.

3.1 Enrollment to the BTS

Before an identification or a registration to a third party service can be performed, a user needs to enroll in the system by taking part in an enrollment procedure.

The first step of the enrollment procedure is that the user must authenticate towards the enrollment system in order to assert the users identity. This authentication can be performed in different ways but the exact details of this authentication is omitted as it is not part of this thesis.

The next step of the enrollment procedure is to collect the biometric features which in this context only includes fingerprints. The principles for this data collection is not considered in this thesis and it will be assumed that the collection will return some fingerprint template that will be stored together with all collected metadata in the BTS database.

During the whole procedure, relevant metadata will be collected and stored and additional metadata, such as sensor information, could also be automatically extracted during the collection of biometric features. It is also possible to extract further metadata by querying the authenticated user with simple questions, e.g. age and gender. Since the collected metadata will come from different sources it is important to determine the trustworthiness of the metadata. It is also important to measure the reliability of the provided metadata since even a trustworthy source will be affected by various errors.

After the procedure has been completed, the user is considered enrolled and an entry corresponding to the user will be added to the BTS database. The entry will combine the metadata and the biometric template in order to provide a binding between the two, which will be used in the identification procedure.

3.2 BTS identification

The first step to be carried out when a user wants to identify thyself to an application is to generate a fingerprint template. The template is generated by scanning their fingerprint using a mobile sensor or a sensor provided by the application. After the fingerprint template has been collected, automatic metadata collection will be carried out, collecting metadata such as device ID and position etc. It is also possible to collect additional metadata such as user account information and application information. The collected metadata will be sent to the application together with the fingerprint template. The application will possibly add further metadata before relaying this information to the BTS. The BTS is then responsible for providing a final identification decision. In case of a no match response from the BTS, it is possible to query the user for additional metadata and retry the identification request.

In order to start out the identification process, the BTS extracts all enrolled users metadata into the candidate set since this is required input to the filtering algorithm, as seen in figure 3.1. The algorithm also requires the metadata of the identification request since it will be used in the filtering process. Since the FAR of the template matching algorithm grows with the number of candidates, the process will cancel if this number exceeds a threshold k. The threshold k needs to be selected by carefully analyzing the biometrics performance of the matching algorithm. A response stating that no match could be found will be sent back to the application if the size of Γ is larger than k. If this is not the case, the identification procedure will continue by ranking the filtered candidate set, $R(\Gamma)$, such that candidates with higher match probability are ranked higher. The ranking will be a part of the filtering algorithm and the details are discussed in a later chapter. The ranked candidate set $R(\Gamma)$ is then sent to the template matching algorithm. Besides taking a candidate set as input the algorithm will also use the query template in the matching process. The matching algorithm will then return a decision stating either no match or the index of the matching template.

If the identification procedure is successful it can be a good idea to store the metadata associated with the identification request together with the metadata of

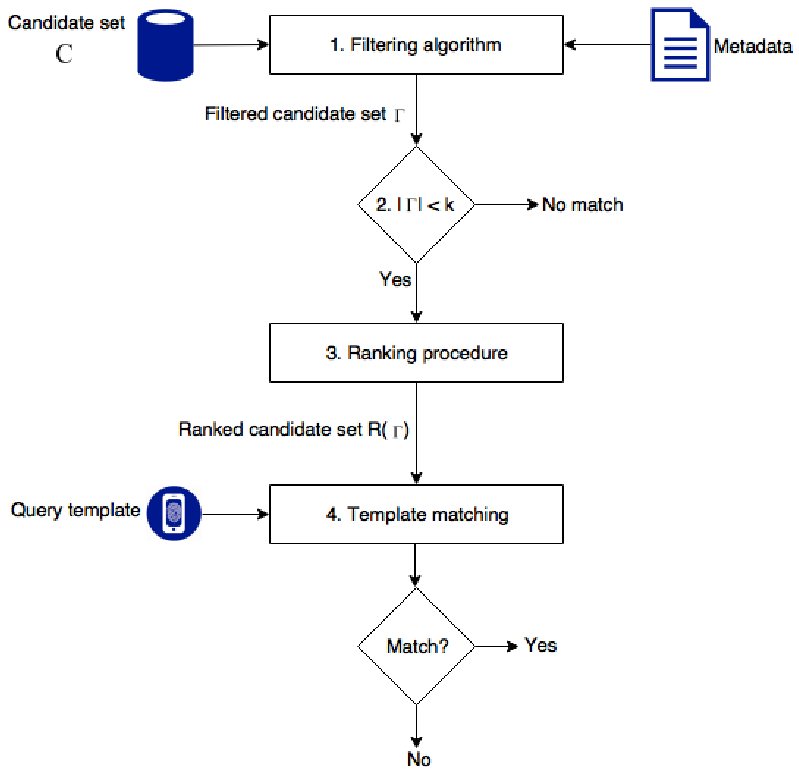


Figure 3.1: Flow chart of BTS identification

the matching user. This is because it might be possible to further improve the filtering algorithm by using this data as training data. The exact details of how this could be done will be discussed in chapter 6.

Since some metadata is prone to change it is important to dynamically update metadata records in order to maintain reliable identification. The metadata can be updated once a successful identification has been carried out.

3.3 User scenario

This section will introduce a user scenario in order to show how the proposed system can be used by different applications in order to achieve a reliable and effortless identification. It will be assumed in this section that the users have gone through the enrollment procedure described in section 3.1.

It shall be noted that even though this example is specifically focused on a web login scenario, it is applicable in many other different scenarios, e.g. mobile applications. It should also be noted that this problem has been solved before by other means such as username and password combinations [3]. This section will introduce another approach, enabling users to authenticate using only their fingerprint.

Depending on the status of enrollment of the user to the application, two different scenarios can occur. If the user has not yet enrolled to the application it will likely be required to fill out additional information. If the user has already enrolled, the user can log in and user related data will be fetched by the application. A detailed description of these scenarios will be described below.

3.3.1 Registration on web page

Before a user can login to a web page it must typically create an account. This will be done by first letting the user provide some additional information such as name and age. The next step in the registration procedure is to scan the users fingerprint in order to create a template representation. This information will then be sent to the application server together with automatically collected metadata. This corresponds to step 1 in figure 3.2. This registration request will then be relayed to the BTS which will check if the user is enrolled or not. If the user is not enrolled it will send a response and terminate the process. In the other case the record corresponding to the user will be updated. The application will also store some information about the user in its database, which corresponds to step 3 in figure 3.2.

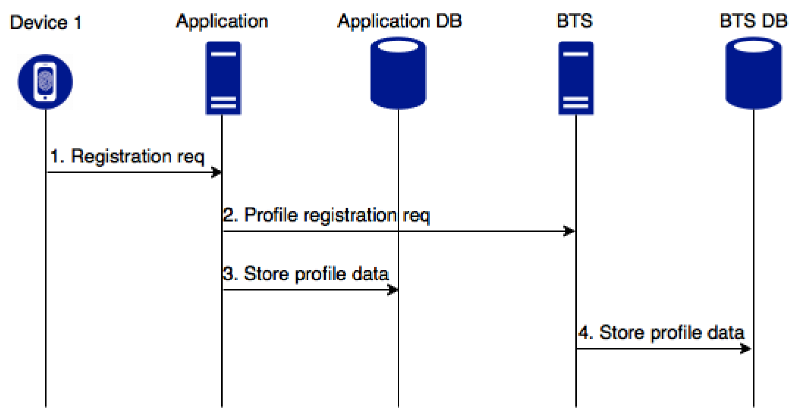


Figure 3.2: Illustrating a registration scenario on a web page

3.3.2 Login to web page

The first step in the login procedure is that the user request access to the web page. The web page then responds with an login request, requesting a fingerprint template and associated metadata. This corresponds to step 1 in figure 3.3. The next step is that the user responds with the template, automatically and possibly manually collected metadata. The web page forwards this data to the BTS which then fetches the candidate set from the BTS database. After the candidate set has been fetched it is used in the filtering algorithm in order to filter out improbable candidates. The filtered candidate set is then used as input together with the fingerprint template, to the template matching algorithm. The algorithm will return either no match or the index of the matching template in the filtered candidate set. If the algorithm found a match, the BTS then proceeds with updating its database. Lastly the identification response is sent to the web page.

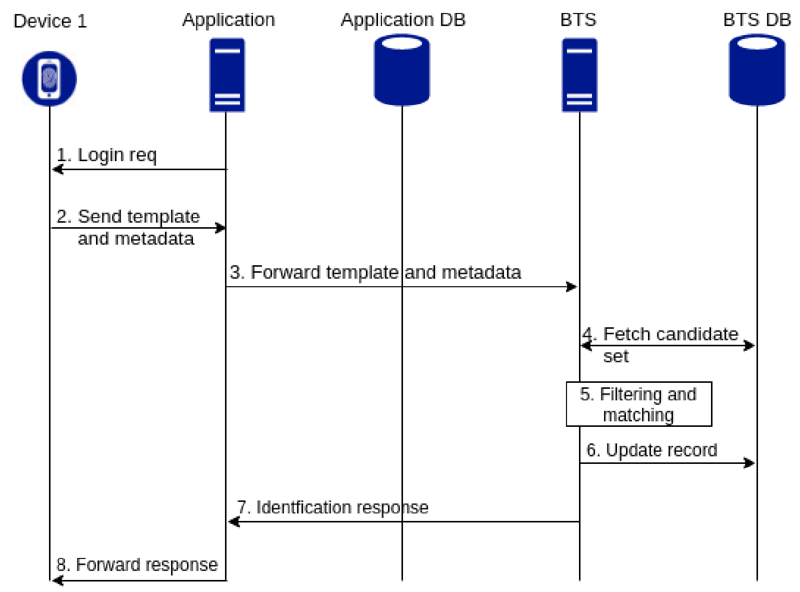

Device 1
Application
Application DB
BTS
BTS DB
1. Login req
2. Send template and metadata
3. Forward template and metadata
4. Fetch candidate set
5. Filtering and matching
6. Update record
7. Identfication response
8. Forward response

Figure 3.3: Illustrating a login scenario on a web page

Chapter 4

Theory

This chapter deals with the problem of implementing the filtering algorithm used in the preprocessing step, as seen in figure 4.1. As discussed previously, this filtering is needed to keep the FAR low, even with a very large candidate set C. All algorithms presented in this chapter will use the candidate set C and the metadata associated with the identification request as inputs and produce a ranked filtered candidate set $R(\Gamma)$.

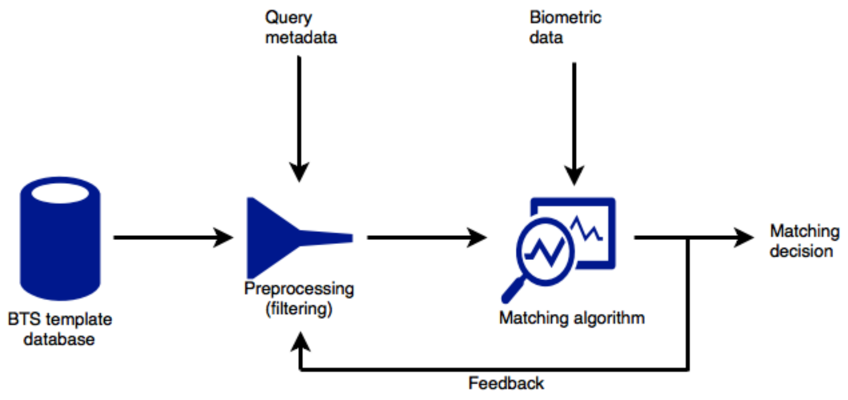


Figure 4.1: Illustrating the proposed solution to lacking training data

An ideal algorithm would take the candidate set C and output a ranked filtered candidate set $R(\Gamma)$ with a fraction of the cardinality of C while the true match still resides in $R(\Gamma)$, given that a true match exists. Furthermore an optimal algorithm should also rank the true match higher than any other element in $R(\Gamma)$. Such an algorithm would be able to maintain a low FAR for a large candidate set C without increasing the FRR.

This thesis will divide approaches into two different categories, namely machine learning and traditional approaches. The machine learning approach assumes that there exists a data set that can be used for learning a mapping between inputs and outputs. Machine learning techniques can furthermore be divided into supervised and unsupervised machine learning [15]. The differences between these two will be discussed later in this chapter. Traditional algorithms does not assume that there exists a data set in the beginning and instead relies on prior knowledge or heuristics.

Since this thesis is based on a pilot project, no real training data will be available. This fact makes it harder to utilize machine learning since machine learning requires a training set. Two possible solutions to the lacking data problem could be to either simulate training data mimicking a real training set or to make use of a feedback system. The idea behind the feedback approach is for the filtering algorithm to utilize valuable information produced by the BTS during the identification procedure. If the identification procedure is successful, it will return the index of the matching user. This index could then be fed back to the filtering algorithm in order to help the algorithm learn patterns between query metadata and metadata of match/non-match users. The feedback flow can be seen in figure 4.1. The exact details of how this information could be used will be discussed later in this chapter.

The rest of this chapter will be devoted to exploring different algorithms that could be used in order to perform filtering. The algorithms will be evaluated according to their applicability for the problem at hand. The studied algorithms will also be compared against each other in order to decide which algorithms that are suitable for further investigations. The selected algorithms has been found by doing an extensive state of the art study on related filtering techniques. During the study one of the aims has been to explore vastly different approaches in order to analyze their different strengths and weaknesses. The algorithms presented in this chapter is a representative collection of algorithms investigated during this study.

4.1 General problems

The nature of the metadata used in this thesis entails some problems when used for generic filtering. This section will describe the problems of missing data and data comparison.

4.1.1 Dealing with missing data

In an ideal case, the metadata in an identification request would be identical to some candidates metadata in the BTS database. However, this is an improbable scenario as metadata in this thesis context is collected continuously and the metadata record in the BTS database tends to grow as more and more information is collected. It is therefor the case that a candidate record in the BTS database tends to contain more information compared to the metadata available in an identification request.

All filtering algorithms must do some sort of comparison between two records in order to find a similarity measurement. Because of the probable scenario of missing metadata, some strategy to counter this must be thought out. There exists many different solutions, where some suggests that the missing data can be counted as zero value, some suggests that the missing data can be calculated as the mean of all available data for that attribute and some strategies suggest use of probabilistic algorithms to calculate the value [16]. In recent years, techniques using machine learning have emerged and can also be seen as a viable option to deal with missing data [17].

The most appropriate approach depends on the algorithm in use, and it is at this time to early to decide which path to choose. Different approaches will be implemented and evaluated during the implementation phase.

4.1.2 Similarity measurements

This subsection will introduce similarity measurements for different metadata types such as strings and numbers. The goal of each similarity measurement is to take two entities of the same type and output a numerical value measuring how similar the entities are.

Comparing strings

The Damerau-Levenshtein distance counts how many operations that are needed in order to transform one string to another string [18]. The operations that are allowed are insertions, substitutions, deletions and transpositions. For an example lets consider transforming the string "aMrcuk" into "Marcus". This transformation can be done by using one transposition and one substitution, therefore resulting in a distance of two.

Another alternative is the Jaro-Winkler distance [19] which is a metric of the number of single character transpositions changes between two strings. The Jaro-Winkler distance uses a scale factor p which gives better scores to strings that match in the beginning.

Comparing numbers

The distance between numbers are easily computed by computing the absolute difference between the two.

Comparing positions

The position metadata collected is in latitude/longitude format and therefore there exists a need to compute the distance between two such coordinate points. The Haversine formula will calculate the great-circle distance between two such coordinate points [20]. This represents the flight distance between two coordinate points on the earths surface. The earth is however not an exact sphere so the Haversine distance is an approximation of the distance between two points.

4.2 Traditional algorithms

The algorithms introduced in this subsection is classified as traditional algorithms since they do not contain a learning phase in the same way as machine learning methods. However, it is important to notice that the classification of an algorithm is not a clear cut and some "traditional" algorithms might actually be classified as machine learning as well.

4.2.1 Locality Sensitive Hashing (LSH)

Locality Sensitive Hashing is an approach that can reduce the dimensionality of data by using a chain of hash functions to cleverly get similar multi-dimensional data objects, with a high probability, to produce hash codes that ends up in the same hash bucket [21]. This property can be seen in figure 4.2, where similar records from the candidate set ends up in same hash bucket. By hashing the query metadata with the same hash functions as used for the candidate set C, records from the resulting hash buckets can be collected for a more detailed comparison. This comparison will return the set of metadata records that are the most similar to the query metadata together with a ranking of the returned records, $R(\Gamma)$. The comparison can be done with an arbitrary chosen similarity measurement where one example may be the Euclidean distance. It will then be checked that the size of $R(\Gamma)$ is lower than the threshold k. This approach suffers from one big limitation

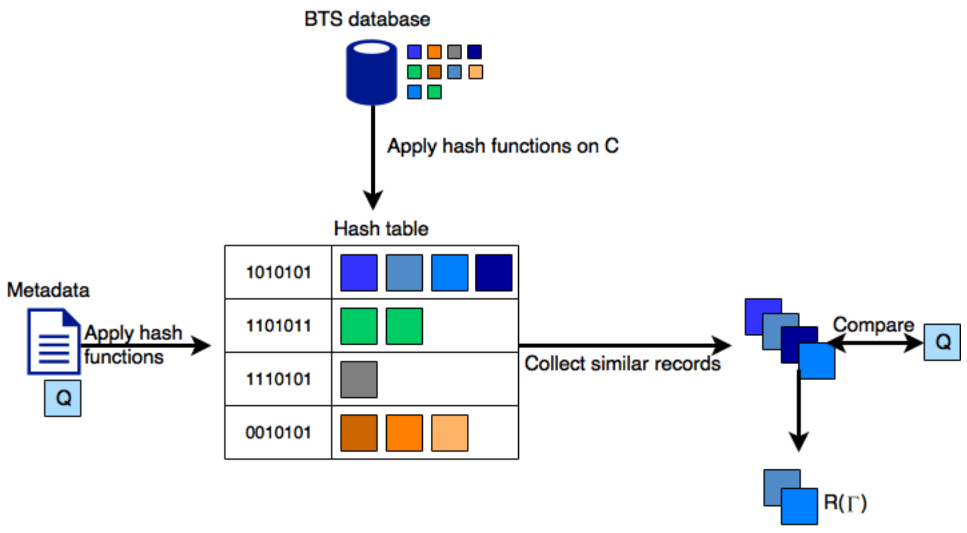


Figure 4.2: Illustration of the LSH approach

as it is not possible to emphasize the discriminating power of different metadata types. To put this into context one scenario may be when the metadata available at identification consists of device ID and position coordinates. The device id may be equal to some user in the database, but the position differs as the user may have traveled somewhere. This means that discriminating power of the device

ID attribute is higher than the discriminating power of the location attribute. It is also unclear to which degree missing metadata affects the performance of this approach, as it may place non similar records into the same buckets, producing poor filtering performance.

The main advantage of this approach is that it is able to prune the candidate set of improbable candidates, and therefore reduce the number of comparisons needed. This would result in a reduced time needed to find a potential match.

Since this approach has a problem with emphasizing the discriminating power of different attributes, we will not consider it further.

4.2.2 Attribute based cascade filtering

A more straightforward method is to perform the filtering using one attribute at a time. The algorithm starts by assigning Γ to contain the records of the candidate set fetched from the BTS database. The filtering process will then be repeated until the size of the filtered candidate set Γ is lower than some given threshold k, as seen in figure 4.3. This strategy works well if it can be assumed that the attribute values form a uniform distribution with many possible values. The efficient reduction comes from the fact that in each step, the algorithm removes all items from Γ not agreeing with the query for the given attribute A. Since different attributes have different discriminating power it is important to select the most discriminating attributes in an early stage to achieve good performance. One way to do this would be to manually determine the ordering but it is also possible to let the algorithm dynamically select the best ordering using a measurement of their discriminating power [22, 23]. One drawback with the approach from the referenced

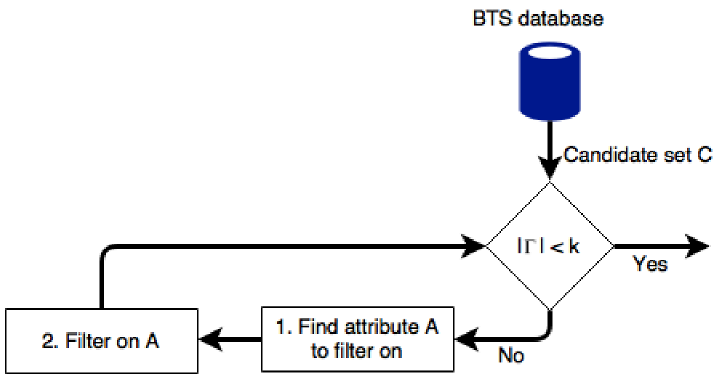


Figure 4.3: Illustration of a cascade filter approach

papers is that the basic implementation lacks support for numeric attributes. This approach is also sensitive to changes in metadata, as the query may have a field with an updated value for some attribute compared to the matching record in the BTS database. This would result in a higher FRR as the filtering disregards the matching record. It may however be possible to implement some kind of support for similarity measurement between two attributes, instead of comparing them for equal values. Additionally support for numeric values could be integrated with the help of the similarity measurements.

The main advantage of this method is that it does not require any prior knowledge, such as a training set. Another advantage is that the method is quite simple to understand and implement compared to other methods.

This method will not be considered further due to its inability to incorporate decisions on several attributes at the same time. This inability will most likely result in poor performance. It may however be possible to perform cascade based filtering as a preprocessing stage in order to remove candidates that are clearly not a match.

4.2.3 Score fusion

The word fusion means combining two or more entries into a new entity. This is the basic idea behind the score fusion methods where the aim is to combine a number of distinct scores into a new score [7]. The individual scores are computed by some abstract modules and it is assumed that these modules will output a numerical similarity value. For an example, consider measuring the similarity between records containing the two attributes: name and age. Furthermore, let us assume that we have a module able to compute a numerical similarity measurement of names and ages. These modules should take two inputs of the corresponding attribute type and return a numerical measurement of how similar they are. The next step is to ensure that the numerical output from the different modules are in the same domain by applying a normalization procedure to the outputs [5]. There exists different methods for performing normalization. The next step is to combine the normalized scores into a single scalar score by applying some score fusion method f. This method could be seen as a function $f : \mathbb{R}^n \to \mathbb{R}$, where n is the number of individual scores. One example of such a function f is a simple summation.

The score fusion method could be used for implementing a metadata filtering algorithm by first retrieving the candidate set C from the BTS database, as seen in figure 4.4. The next step is to compare the query metadata against each candidate in order to compute a similarity measurement. The similarity measurement will be computed by first computing the individual similarity measurements of the different metadata types. The final similarity measurement is obtained by applying the function f as described above. The ranking $R(\Gamma)$ can then be done using the fused similarity scores.

The main drawback of this approach is the need of a comparison with every candidate in the candidate set. Another drawback with this method is that it is somewhat unclear which function f that is the most suitable.

One advantage with this method is that it is quite modular and more metadata

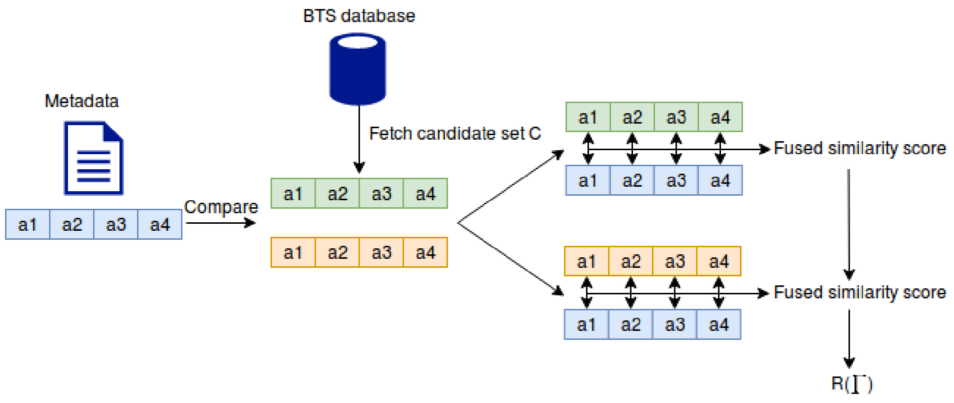


Figure 4.4: Illustration of score fusion

types can be added easily by just defining a similarity measurement. Another advantage is that the method produces one score for each metadata type which is good since it allows easy integration of trustworthiness and reliability levels.

The conclusion is that this method seems quite suitable for further considerations. The drawback of having to implement similarity measurements is not that important since other methods will also need some kind of data transformations similar to the transformations needed for this method. Some advanced functions f using collected data will be discussed later in this chapter.

4.3 Machine learning

Machine learning algorithms is commonly divided into two categories, namely supervised and unsupervised algorithms.

The name supervised comes from the fact that supervised algorithms requires help from a teacher in order to learn to perform the task on its own. This teacher-algorithm relationship resembles an ordinary teacher-student relationship where the teacher shows examples together with their solution to the student with hope that the student will then be able to solve similar problems on its own [13]. The problems can be of different nature such as learning to recognize objects in images or predicting the next word in a sentence given the previous words. It is common that the supervision comes in the form of recorded experience from the past where expected outputs are given for different combinations of inputs. An example of recorded experience can be seen in table 4.1 where the next word of a sentence is given together with a constant number of previous words. In this example the expected outputs are given in the third column meanwhile the corresponding inputs are given by the previous word together with the current word. Experience given in such a way is often called a training set and usually the training set consists of many more examples.

Table 4.1: Table showing an example of a training set

Previous word	**Current word**	**Next word**
My	name	is
Coffee	is	good
Walk	with	me

All supervised algorithms utilizes a training set in the learning phase where the algorithm will try to learn to perform the given task on potentially unseen inputs. After the learning phase is completed the algorithms will usually be evaluated using an unseen test set in order to measure the accuracy of the model [13]. It is crucial that the test set has not been used during the learning phase of the algorithm, as this will cause the performance measurement to become biased. One simple approach is to divide the given training set into two parts where the first part is used during training and the second part is used as a test set. Figure 4.5 illustrates the approach where the available data is divided into a training and a test set.

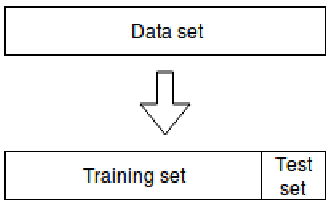


Figure 4.5: Illustrating testing using partitioning of data

As the available data set is divided into two parts, not all data is used during the learning phase. Since it is desired to use as much data as possible during the learning phase, it is common to use a relatively small test set. This causes a problem since a small test set may cause relatively noise performance estimations. One solution is to use a more advanced estimation technique called cross-validation. This procedure is illustrated in figure 4.6. The procedure uses several combinations of test and training sets in order to compute a more accurate estimation of the true performance.

In contrast to supervised algorithms, unsupervised algorithms does not make use of a training set with given input/output value pairs. Instead, from a given input data set, the unsupervised algorithms are able to learn inferred patterns [15]. Using these patterns, the algorithms are able to create a model and use this in order to classify or associate future data input.

It is relatively difficult to estimate the performance of unsupervised algorithms compared to supervised algorithms. This is because a test set may be lacking

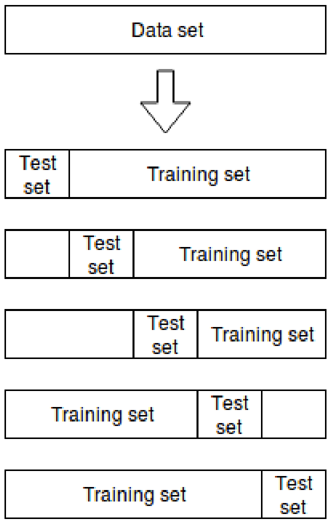


Figure 4.6: Illustrating testing using cross-validation

when applying unsupervised algorithms. However, there exists methods that are problem and algorithm dependant for computing an estimation of the performance.

Whenever thinking about using machine learning one should always consider using an existing implementation since there exists many libraries for both supervised and unsupervised algorithms. Using an existing library saves a lot of time as well as providing a higher level of correctness assurance given that the library is extensively tested and maintained. One example of a simple popular library implementing both supervised and unsupervised algorithms is Scikit-learn [24]. This library is a good starting point and could be used in order to solve many classical problems such as classification and regression. Another increasingly popular library is Keras which supports more complex supervised algorithms such as artificial neural networks able to approximate any continuous function [25, 26]. This library is more powerful than Scikit-learn and could be used if more complexity is needed.

The rest of this chapter will be devoted to introducing algorithms of both categories that could potentially be useful for the problem at hand.

4.3.1 K-means clustering

K-means clustering is an unsupervised algorithm that builds on the idea that similar items will be close to be each other in a multi-dimensional space [27]. The variable k is an input parameter to the algorithm, which decides the number of clusters to divide the given data into. Each cluster will be represented by a centroid point, which is placed in the center of the cluster. The parameter k can be determined by looking at the data and making an educated guess.

The algorithm starts with first placing k centroid points arbitrarily in the multi-dimensional space. The algorithm then alternates between two phases, until the clustering is stable or a fixed number of iterations have been executed.

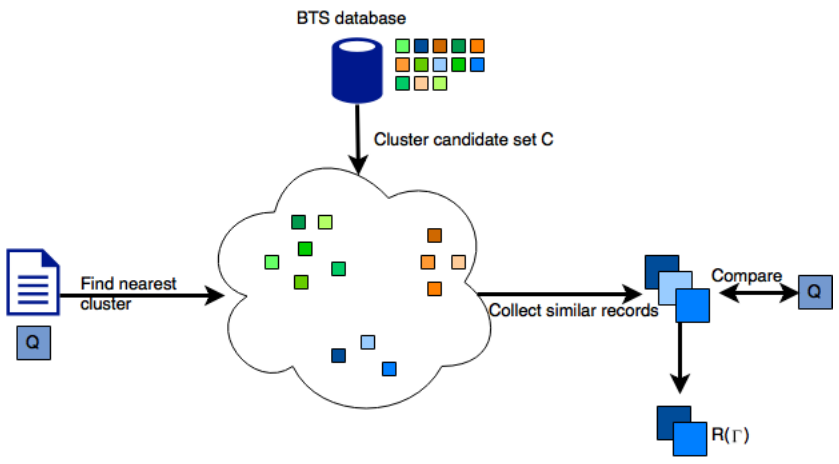


Figure 4.7: Illustration of cluster approach

The first phase is to assign all data points to a cluster. This is done by calculating the distance between the data point and all centroid points, then assigning each point to its nearest cluster. The distance formula used to compute the distance can be any distance measurement, e.g. Euclidean or Hamming distance, whichever fits the given data the best. The second phase is to update the centroid points of all clusters by calculating the mean of all data points in a cluster.

Figure 4.7 illustrates how K-means clustering can be applied to perform meta-data filtering. The first step is that all candidates in candidate set C are clustered according to the algorithm described above. When the filtering algorithm runs, it calculates the distance from the query point to all centroid points and selects the cluster with the smallest distance. A more detailed comparison between the candidates in the selected cluster and the query is then carried out, in order to perform a ranking. The ranking can be performed by introducing different weights for the different metadata types in the Euclidean distance computation. A short distance will be ranked higher than a large distance. Before passing along the ranked candidate set $R(\Gamma)$ along to the matching algorithm, lowest ranked candidates may potentially be removed.

There are some problems with this approach as the clustering assumes that all variables have the same variance as well as all clusters have the same amount of observations. If these conditions are not true, the clustering may result in poorly distributed candidates between the clusters. The weights to use for the distance calculations must be tuned in some way such that the discriminating power of certain metadata types are taken into consideration. Finally the algorithm requires that the data is numerical to enable calculation of the mean values. To bypass this, non-numerical data could be converted into numerical data in some way. Reddy and Kavitha [28] have come up with another approach to effectively cluster a mix of categorical and numerical data by modifying how the centroid points of a cluster is determined.

K-means clustering has the power to reduce the search space due to the splitting of data into clusters. The efficient reduction builds on the clusters being of roughly the same size.

This algorithm will not be considered further as the data will have to be clustered into very small clusters in order to effectively reduce the search space. With missing metadata and different metadata available this clustering would likely not work well.

4.3.2 Probabilistic record linkage

Record linkage is a supervised approach that aims to decide if two different records are sufficiently similar. There are different versions of record linkage and probabilistic record linkage is one method that aims to assign a probability of two records representing the same entity [29]. This method works by computing a similarity measurement of two records. This measurement can then be used for deciding if two records point to the same entity. The similarity measurement of a record pair is computed as the sum of the individual similarity measurement for each attribute. The individual similarity measurement will be a positive number for matches and negative for non matches. The magnitude of the number will be influenced by the importance of the attribute. The total similarity measurement is then computed as a function of the individual similarity measurements [30]. If the total similarity score is above a certain threshold that could either be computed automatically or manually determined it is considered to link to the same entity. The individual similarity measurement is a function of the two probabilities m and u. The probability u is defined as the probability of an attribute to agree for a non matching pair and m is defined as the probability of an attribute to agree for a matching pair. There exists different algorithms for estimating m and u, e.g. the EM algorithm and a frequency based estimation [30, 31].

This method could be used for candidate filtering by first fetching the candidate set C from the BTS database. A received query will then be compared against each candidate in the candidate set C, producing a similarity score for each candidate. The candidate set will then be ranked on the similarity score and candidates with low similarity score will be discarded.

One disadvantage with this method is that it requires a linear search through the whole candidate set C. Another disadvantage is that it is complex to obtain good estimates for m and u without any prior training data.

The most important advantage with this approach is that the discriminating power

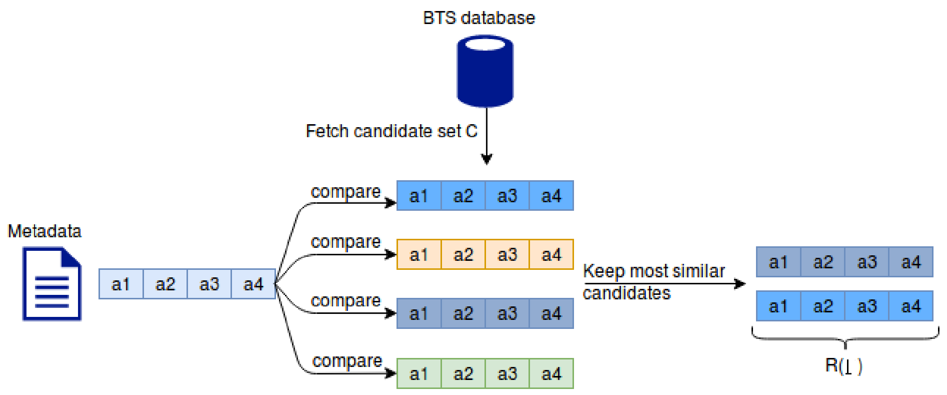


Figure 4.8: Illustration of a probabilistic approach

of the different metadata types are taken into consideration. Another advantage is that is it possible to include the trustworthiness and reliability scores into the algorithm. A third advantage is that the method produces a ranking of the candidates.

This method seems appealing given the discussed advantages and the not so critical disadvantages. The biggest problem that could hinder an efficient implementation is the estimation of m and u and the linear computation time of comparing the identification requests with all records in the database.

4.3.3 KD-trees

KD-trees is a tree structure used to store multi-dimensional data [32]. KD-trees works similar to binary trees and repeatedly splits the data along one single dimension at each level in the tree and the dimension used at the split varies between the levels of the tree in a round robin type of fashion. The split is done by calculating the median of the attribute from all of the data points, then placing the node on the left if the attribute value is less than or equal to the median, and on the right if it is greater.

To apply this to our candidate set C, the whole set could be stored as an KD-tree and the splits could be done by the different metadata attributes. When comparing the query metadata one would just traverse down the tree, comparing one attribute at each level, until a close enough match is found, or stop at some level in the tree to get a specified number of candidates extracted to the matching algorithm.

This approach has some major limitations as it requires all metadata to have the same dimension, in order to properly build the tree. KD-trees filtering approach is very harsh as it disregards a lot of candidates because of differences in a single attribute. This is a naive approach as the distance in one dimension does not properly represent the total distance if all dimensions were to be taken into consideration.

KD-trees have been proven effective when used as a underlying data structure in an nearest neighbour search. However the efficiency greatly decreases as the dimension increases [33], which is a problem with the possible metadata available.

Because of the limitations of this approach, it will not be considered further.

4.3.4 Self learning score fusion

Score fusion as introduced earlier in section 4.2.3, was a method for fusing together a number of individual scores to a new single scalar score. The score fusion was performed by using a fusion function $f : \mathbb{R}^n \to \mathbb{R}$, where n was the number of individual scores. The function f was determined by selecting a suitable function for the particular problem.

The goal of self learning score fusion is to remove the requirement of having to manually specify a fusion function f. The function will instead be learned by applying machine learning techniques using available training data [6]. There exists different suggestions for how to apply machine learning in order to learn the fusion function [34]. This method needs training data and it can be imagined that such data can be generated by extracting this data from successful identification request as described earlier. Figure 4.1 illustrates this approach where information from the identification decision is used in order to improve the accuracy of the algorithm. The positive training examples can be generated by storing the query metadata together with the matching data record in the BTS database and the negative examples can be generated by storing the query metadata together with a random non-matching record in the BTS database. This procedure would also make sure that the training data is well balanced with positive and negative examples.

The filtering process would be the same as the process described earlier for score fusion filtering. The only difference is the training part as described above.

One disadvantage with this approach is that it requires training data and such data will not be available in the beginning. It can therefore be a good idea to use simple score fusion in the beginning meanwhile collecting training data. The training data can be used to train a more advanced score fusion function using a SVM or a neural network [6, 34]. The simple fusion function can then be replaced with the more advanced function when it has been trained enough.

One advantage with this method is that it will be able to learn a more advanced fusion function compared to the simple score fusion. This method will also make it easy to integrate the trustworthiness and reliability scores. One way to incorporate them would be to multiply the similarity scores with them. Another way to incorporate them would be to add them as input letting the machine learning method learn how to incorporate them into the fusion process.

Chapter 5

Evaluation Framework

The previous chapter introduced several generic filtering techniques as a part of a background study, aimed at finding suitable techniques for further consideration. The next step in the journey towards a concrete filtering implementation is to develop an evaluation framework, to enable comparisons between the different filtering techniques. It is important that the framework is able to provide unbiased performance measurements in order to be a sufficiently satisfying framework. The obtained measurement would otherwise not accurately represent the performance of the filtering algorithm seen when deployed in real life. In order to obtain unbiased performance measurements, rigorous testing using input data representing the operational environment needs to be conducted. There exists different approaches to ensure a representative performance measurement which will be discussed below.

One very common testing procedure is to use a data set recorded in a real life scenario. This approach builds on probability theory claiming that the measured accuracy on a sufficiently large data set will approximate the accuracy on a much larger data set [35]. This approach is suitable to use if there exists prerecorded data or if there exists a publicly available data set, representing the environment, gathered by an unbiased third party.

Another similar approach would be to let people use the system in order to get a measurement of the systems performance. This approach requires careful selection of testing individuals in order to get an unbiased measurement. It would also be very costly.

A third approach is to simulate testing individuals in order to obtain performance measurements more effectively. This approach only works if the simulation is able to capture real world dynamics. Because of this fact it is crucial to build the simulation on empirically collected data since otherwise it is very hard to associate trustworthiness to the obtained measurements.

Since we were unable to find a data set, sufficiently representing our scenario, it was decided that some kind of simulation would be the most appropriate in this case. Since manual simulation is both costly and time consuming, it was further decided that computer driven simulation was the best alternative.

The primary purpose of the developed simulator is to provide necessary performance measurements as discussed previously. These performance measurements will be used in a later chapter to compare the performance of different filtering implementations in order to determine their efficiency from several perspectives. The simulation engine should be tailored to represent the usage scenario of users identifying themselves via a biometric template and some collected metadata, in a real life representative manor. To satisfy this constraint, almost all data has been sampled from real distributions, gathered from SCB[1]. The simulation engine should also allow simulation of a growing database of enrolled users, in order to capture the performance changes over time. Furthermore the engine should be highly modular in order to allow easy integration of new metadata types.

There are two events that are simulated, namely registration and login events. The registration event corresponds to a user enrolling to the BTS, simulating a growing user base. The login event simulates a user trying to identify themselves using a biometric template and collected metadata. The performance measurement will be carried out by measuring the amount of successful logins divided by the amount of genuine login attempts. In order to capture a reliable measurement, the login procedure must be repeated for different users, many times.

5.1 Metadata types

As discussed previously in section 3.1 and 3.2, there exists many different metadata types that could potentially be used in the filtering process. This means that there exists a need to simulate different metadata types in order to investigate their different properties:

- **Uniqueness:** The entropy of the metadata values
- **Privacy concerns:** How the inclusion of the metadata type affect the users privacy
- **Availability:** Deals with answering how user friendly it is to collect the metadata types
- **Complexity:** How computationally hard it is to use the metadata type in a similarity measurement

The rest of this section will describe the registration procedure, login procedure as well the defined properties for a chosen set of metadata types individually. These types have been selected because their applicability to filtering has been deemed efficient. It should be noted that filtering with metadata types is not limited to these types only and that all metadata will be transferred to the BTS in an encrypted form, meaning that no third party will be able to read the the data in clear text.

[1]www.scb.se

5.1.1 Device ID

It is standard procedure that most hardware modules are assigned a unique identifier to assure traceability and enable quality control. The following discussion uses assumptions that the biometric sensors used in the identification procedure are manufactured by Fingerprints and that each biometric sensor has a uniquely designated device ID. This ID is statically determined and it should not be changed during its lifetime. Furthermore it is also assumed that each biometric sensor will contain a digital signature associated with its device ID. Because of the nature of digital signatures it is infeasible to generate valid signatures without knowing the private key [36]. In the case of user identification using device ID from a biometric sensor it is assumed that the device ID has been used when generating the digital signature, therefore protecting against attacks using spoofed device IDs. This means that the device ID can be used as a secure identifier in an identification process. It is assumed that the supplied device ID is authenticated before the identification request reaches our filtering algorithm and it is therefore safe to assume that the device ID is valid.

Uniqueness

The uniqueness of the device ID metadata type is very high since a device ID is uniquely assigned to each sensor by the manufacturer. This means that it is easy to find likely candidates by performing a device ID lookup given that the querying user has not changed device. It is also a likely scenario that some users may share device, e.g. in a family. As long as the set of users that share the same device is not large, the device ID will still likely be enough to uniquely identify a querying user with the help of the biometric matcher.

Privacy concerns

Since the device ID is digitally signed and unique, it is possible to track an individual to some extent raising privacy concerns. This however requires the attacker to either crack the encrypted communications or to get their hands on the BTS database. Having several users share the same device would reduce this concern as it would then be unfeasible to track the individuals based on only the device ID. There is also a serious concern for the individual user that someone can impersonate them if the device ID protection is compromised [37].

Availability

It is assumed that the device ID is always available since each sensor has a associated device ID that could be collected automatically without any user involvement. This also means that collection of device ID is very user friendly.

Complexity

Since the device ID can be regarded as a unique string, it is easy to perform an efficient hash lookup on the device ID, allowing a very efficient implementation.

Collection procedure

The procedure would be that the application automatically includes the device ID in the identification request without any user involvement. In the same manor, the device ID would be included during enrollment. The requests would then be transmitted to the BTS in an encrypted form.

Registration simulation

A device ID for a newly registered user is determined by the following pseudo code algorithm.

Algorithm 1: Generate ID at enrollment

```
Data: current assignments(A) and distribution of nbr users per id(B)
Result: enrollment device ID
X ← ids of devices with less then their designated number of users in A;
if X is empty then
    generate new id y;
    generate number of users M for y from B;
    add y to A;
    return y;
else
    x ← random id from X;
    add x to A;
    return x;
end
```

The algorithm samples a device ID from a given set of available IDs, where the number of user slots for each device ID is sampled according to the distribution seen in figure 5.1. If there exists no available identifiers, a new one will be generated and the number of user slots on that ID will also be sampled from the distribution seen in figure 5.1.

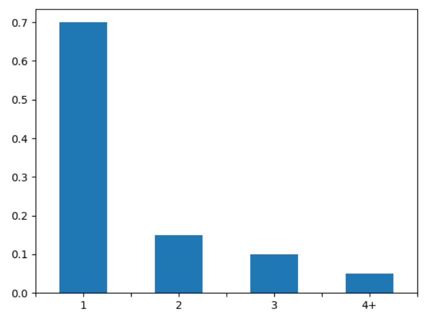


Figure 5.1: Distribution of number users per device ID

Simulated login

The device ID associated with a login request is generated by the following pseudo code.

Algorithm 2: Sample login ID

```
Result: query device ID
x ← user device ID;
roll ← dice roll 1-100;
if roll ≤ 10 then
    x ← sample a new ID;
    return x;
else if roll ≤ 20 then
    x ← sample another existing ID;
    return x;
else
    return x;
end
```

The algorithm performs a dice roll in order to determine how the query device ID should be generated. The algorithm then either keeps the previous ID, samples an existing device ID possibly used by other users or generates a completely new ID according to a given distribution. The distribution can be seen in the pseudo code of algorithm 2. The initial goal was to build this distribution on available statistics but as no data describing this scenario was found, it was decided to select a distribution that was believed to accurately represented the worst case scenario. It should however be noted that this distribution could be changed easily in light of new information.

5.1.2 Age

It is a well known fact that all persons have an age that can be represented as an integer, stating how many years have passed since the person was born. This fact together with the ease of entering a two digit number motivates investigations into the possibility of using age in an algorithm. The foundation of the age simulator is statistics gathered from SCB [38].

Uniqueness

Since the age of a person may vary between around 0 to 100 years, the discriminating power of the age type is high. As seen in figure 5.2, the highest probability lies at the age of 27. This implies that if the user's ages is represented by the distribution in 5.2, using the exact age in years when filtering would reduce the number of candidates with an average of 98.4% or more. The exact distribution that would represent the users in an deployed system is unknown, but a likely scenario would be that most users would be around 15 to 60 years old.

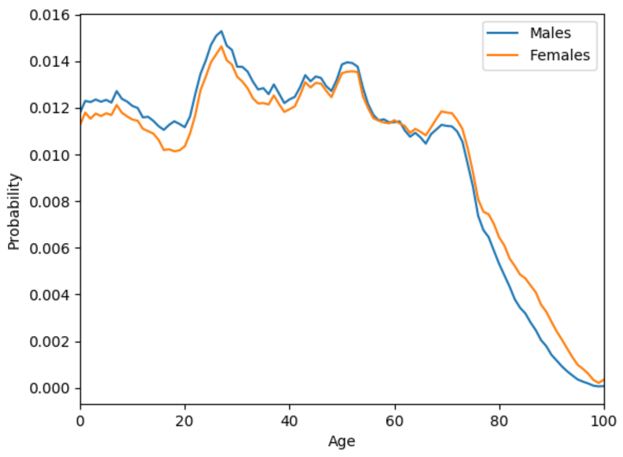


Figure 5.2: Age distributions of the two genders

Privacy concerns

The age metadata type is not sensitive on its own but can be sensitive when combined with other information.

Availability

As stated earlier, all persons have an age that can be used given that the users are able to provide their ages. This might be a problem for very small children and people with disabilities, raising availability problems that must be considered when deploying the system into the real world.

Complexity

Filtering on age is efficient from a computational perspective since the filtering can be done by using an efficient indexing data structure, such as a hash table. This means that age filtering can be efficiently deployed for a large database.

Collection procedure

The collection procedure could be done by manually means, e.g. prompting the user to enter their age.

Registration simulation

The age associated with a registration is simulated by using the distributions seen in figure 5.2. The first step of the simulation procedure is to decide which distribution to use by inspecting the gender of the enrolling subject, as can be seen in algorithm 3. The age is then computed by drawing a sample from the selected distribution.

Algorithm 3: Generate enrollment age

```
Data: Male age distribution (M), Female age distribution (F), gender of
      user (G)
Result: Enrollment age
distribution ← get M if user is a male else F;
age ← get one sample from distribution;
return age;
```

Simulated login

The login age associated with an identification request for a particular user is simulated by first obtaining the recorded age of the user. The next step is then to perform two dice rolls in order to simulate the possibility of users entering a slightly wrong age, as can be seen in algorithm 4. The probability of a user entering a wrong age was selected by considering the worst case scenario. This probability can be changed easily in the program in order to study the performance under different circumstances.

Algorithm 4: Sample login age

```
Data: Login user (user)
Result: Login age
age ← obtain saved age of user;
roll ← dice roll 1-100;
if $roll < 20$ then
    roll ← dice roll 1-100;
    sign ← 1 if roll < 50 else -1;
    age ← age + sign;
end
return age;
```

5.1.3 Name

Since each person using the identification service normally has a first and surname it is a good idea to include name as a potential candidate for metadata filtering. The name simulation is built on statistics from SCB in order to accurately simulate the name distribution seen in Sweden [39, 40]. Besides simulating the actual name distribution, the simulator also simulates common typing errors according to distributions found in an empirical study by Baba and Suzuki [41].

Uniqueness

Figure 5.3 shows the name distribution for the 100 most common first and surnames for both male and females. This data was extracted from the statistics gathered from SCB. As can be seen in the figure, names differ in popularity, which leads to different uniqueness for different names. More common names will have less uniqueness, rendering less efficient filtering opportunities. On the other hand, less common names will result in more efficient filtering.

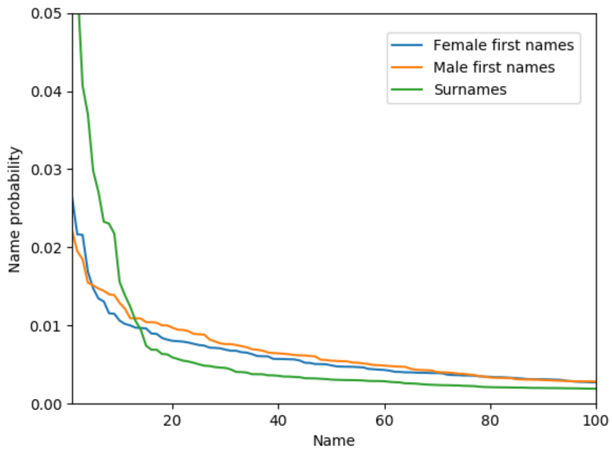


Figure 5.3: Probability distribution of the 100 most common names in Sweden

Privacy concerns

Since some names are more unique than others it is reasonable that some persons with highly unique names will be less inclined towards sharing their full name than persons with more common names. This concern could be mitigated by using a similarity conserving hash function, storing the hashed names in the database instead of the real names [42]. These existing hash functions does however not work very well with names as the similarity after the hashing is quite distorted, resulting in a big accuracy decrease.

It has also been found in an empirical study that the brand of the company asking for personal information has an impact on the users willingness to provide personal information [43]. This fact can be used in order to increase the users willingness to provide sensitive information by clearly displaying the Fingerprint logo. This action will make it clear to the user that the information is only shared with Fingerprints and not with the application itself.

Availability

During normal usage scenario, it will not be feasible to automatically collect the name of the querying user. Instead a manual name collection has to be carried out by explicitly asking the user to enter their name. From a usability perspective, this is not ideal and should therefore be avoided if alternative metadata can be used instead. The name collection may be used as a last resort, only performed if the user is willing to share this information with the identification service.

Complexity

Name based filtering builds on a similarity measurement and the complexity of this approach is therefore highly influenced by the complexity of the similarity measurement. Some examples of similarity measurements includes the Damerau-Levenshtein and Jaro-Winkler string distances, discussed in section 4.1.2.

Collection procedure

The procedure during enrollment would be that the enrolling users are prompted to fill in their first and surname that will be stored together with the enrollment template in the BTS database. This information could then potentially be used during an identification by comparing the stored name against the entered name during identification. The name collection during an identification would be done by prompting the querying user.

Registration simulation

The procedure for generating a full name and typing error probability can be seen in the following pseudo code algorithm.

Algorithm 5: Generate first- and surname at enrollment

```
Data: distributions of surnames(A), male first names(B), female first
      names(C) and typo probability distribution(T)
Result: First- and surname
surname ← random surname from A according to distribution;
typo_prob ← random number from distribution T;
store typo_prob as a simulation parameter;
gender ← random gender;
if gender = male then
    first_name ← random first name from B according to distribution;
    return first_name, surname;
else
    first_name ← random last name from C according to distribution;
    return first_name, surname;
end
```

During enrollment, the gender of the user is first sampled from a uniform distribution between the two genders. Then, depending on the gender, the first name is sampled from either the male or female first name distribution [39] and the surname is then sampled from the surname distribution [40]. Taking spelling errors into account, each user is assigned a probability of making typing errors according to a truncated Gaussian distribution. The parameters of this distribution can be tuned as a simulation parameters.

Simulated login

The procedure for generating manually entered first- and surnames during login can be found below. The first step of the simulation is to obtain the querying users first- and surname. The next step is to obtain the simulation parameter typing errors_prob associated with each user in order to determine the number of typing errors to simulate. The simulated typing errors that corresponds to realistic typing errors done by real users are generated by algorithm 7, taking a string and the number of typing errors to simulate. This typing error algorithm is applied on both the first and surname in algorithm 6 in order to generate names with realistic typing errors.

Algorithm 6: Sample login name

```
Data: current first name(first_name), surname(last_name) and typo
      probability(typo_prob) associated with the user
Result: login name
first_name_typos ← random number according to typo_prob;
last_name_typos ← random number according to typo_prob;
first_name ← simulate_typos(first_name, first_name_typos);
last_name ← simulate_typos(last_name, last_name_typos);
return first_name, last_name;
```

Algorithm 7: Generate typing errors

```
Data: string(s) to inject $x$ typos into and probability distribution of typos
      (C)
Result: string with typing errors
changed_s ← s;
for $i$ in $[1, x]$ do
    typo ← random typo from typo distribution (C);
    changed_s ← inject_typo(typo);
end
return changed_s;
```

Algorithm 7 uses the distribution seen in table 5.1, extracted from Baba and Suzuki [41]. Transpositions are typing errors where two characters have switched places, substitution are where one character has been replaced by another, deletions are missing characters and insertions are extra characters. The insertion and substitution errors uses the QWERTY keyboard layout[2] in order to simulate more realistic typing errors. The effect of this algorithm can be seen in table 5.2.

[2]https://en.wikipedia.org/wiki/QWERTY

Table 5.1: Probability distribution of typing errors

Typo	Probability
Transposition	0.03
Substitution	0.55
Deletion	0.22
Insertion	0.20

Table 5.2: Table showing some examples simulated login names

Stored enrollment name	Simulated login name
arne karlsson	aren karlsson
anna larsson	anna larsso
caroline gustafsson	caroline gustafsson
helena lundqvist	elena lundqvist
sofia söderström	sofia söderström
jakob petersson	jakob peterssob
michael nilsson	michael n8ilsso
li håkansson	li håkansson
krister eriksson	kkrister eriksson

5.1.4 Location

Since the majority of sold smartphones today comes with an embedded GPS module it enables effortless collection and is therefore an interesting candidate to consider for the filtering algorithm. It has been assumed in the simulator that the position is represented as a latitude longitude pair. If some device would use another representation, a conversion to latitude and longitude is required. The location simulation has been built upon several different statistical distributions, making sure that the simulation generates realistic user movements.

Uniqueness

According to a study done by Bhattacharya and Das, a user tends to have a number of significant places that he/she visits frequently [44]. This implies that it should be feasible to use location history in order to estimate a probability that a given candidate is at some certain location at some certain time. This probability can be used in a location based filtering process by removing candidates with a very low probability. However, according to Shaw et al. [45], location should not be used by itself since it is not very effective in densely populated areas. These findings implies that the recall of location is not high enough, and that some additional metadata types are needed when using location in a filtering process.

Privacy concerns

It has been shown that location history can be used to identify persons to some extent, raising potential privacy concerns [46, 47]. A study done by Rossi et al. [48] shows that with just a few high resolution spatio-temporal points, the number of candidates in a dataset can be greatly reduced. It is therefore important that the location history is kept secret by the BTS service in order to maintain the users privacy. One potential migration strategy used to preserve the users privacy is to divide the world into hash buckets. Where one hash bucket, represented by a single coordinate, represents a region and all coordinated within that region would be mapped to the same bucket. The size of the region that the hash bucket represents is however not straightforward to select as there is a trade-off between privacy and accuracy. As the region grows the privacy increases but at the same time the accuracy of the filtering decreases as more users would be mapped to the same bucket. One solution that builds on this idea is GeoHash[3], invented by Gustavo Niemeyer in 2008.

Because of the potential privacy implications of using location history in a filtering process, it should be up to each user to decide if the service should be allowed to perform automatic collection. The user should however be warned that omitting location could potentially have a negative impact on the user experience.

Availability

As mentioned before, the availability of a location measurement is assumed to be high. The collection would be straightforward and no user involvement will be needed.

Complexity

The complexity of location points is very low as the location is represented by longitude and latitude, two float point numbers. A distance between two location points can be calculated using the Haversine formula presented in section 4.1.2.

Collection procedure

During enrollment a user would be prompted whether he/she wants to allow location to be part of the enrolled data. This applies to identification requests as well, and the choice whether to use location or not would only have to be done once. If the location is used in an identification request, the location would be collected automatically.

Registration simulation

The goal is to simulate the spatio-temporal dynamics of a real world user base chosen to be the Swedish population. This means that the simulator must mimic the demographic aspects of the Swedish population in order to be able to correctly

[3]http://geohash.org/

simulate the behaviour of the chosen user base.

The population distribution among different cities are one important aspect and should therefore be used by the simulator. The probability distribution for the 10 most populated cities can be seen in figure 5.4. The distribution used by the simulator contains around 300 of the most populated cities in Sweden and uses statistics gathered from SCB [49].

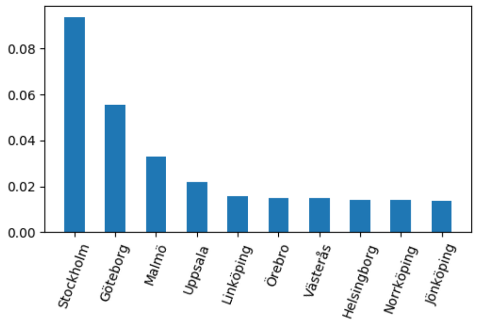


Figure 5.4: Top 10 most probable Swedish cities

The temporal aspect is also very important to integrate into the simulator. Results from studies have been used to model the activity time of a user as well as where the user is likely to be active during specific times during a day [50, 51]. The user activity probability can be seen in figure 5.5 and the user activity location probability can be seen in figure 5.6. The effect of using a spatial city distribution in the simulator can be seen in figure 5.8, where the simulator is able to capture the fact that e.g. Stockholm is more densely populated than northern Sweden.

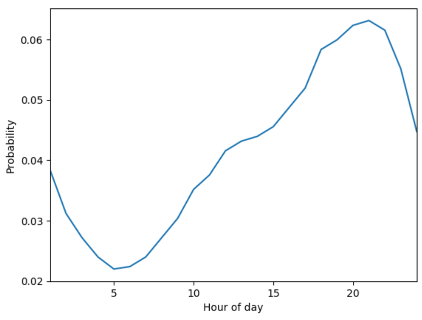


Figure 5.5: Probabilities of user activity during day [50]

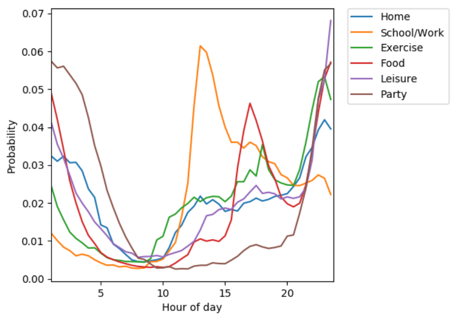


Figure 5.6: Probabilities of user location during day [51]

Algorithm 8 describes the generation of a users location and movement pattern.

The algorithm first samples a city, its location and the city radius from the distributions M and R. The simulator then determines the number of significant places a user has by sampling from the type distribution N, which can be seen in figure 5.7. This data is extracted from a study done by Sibren et al. [52]. The last

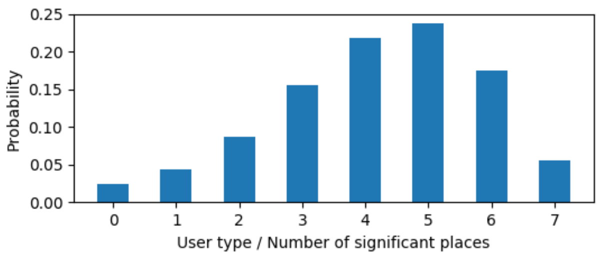


Figure 5.7: User type / number of significant places probabilities

step done by the algorithm is to simulate the location of the significant places by generating different amount of significant places within the city, depending on the user type. One of the significant places locations is then returned as the enrollment location.

Algorithm 8: Generate enrollment location

```
Data: city distribution(M), city radius(R), type distribution(N) and venue
      distribution(V)
Result: Enrollment location
city ← sample a random city from M;
radius ← get the radius of city from R;
user_type ← draw random user type from N;
save user_type as a simulation parameter for the user;
nbr_spots ← get number of significant places for user type user_type;
city_lat, city_lng ← get position of city;
user_lat, user_lng ← move a random uniformly distributed distance
  $[0, radius]$ from position (city_lat, city_lng);
for $i$ in $[1, nbr\_spots]$ do
    spot_lat, spot_lng ← move a random uniformly distributed distance
      $[2, 7]$ from position (user_lat, user_lng);
    save spot_lat, spot_lng as a simulation position for the enrolling user;
end
user_lat, user_lng ← select random from saved simulation positions
  according to the venue distribution V;
return user_lat, user_lng;
```

Simulated login

The first step of the login simulation is to retrieve the stored simulation spots for the simulated user. The venue distribution is then used to decide which of the

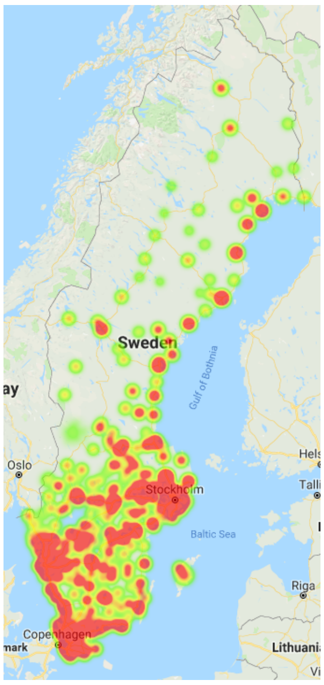


Figure 5.8: Heat map showing locations of 200 000 enrolled users

simulated users spots that should be simulated. The next step of the simulation is to retrieve the stored position and radius of the selected simulation spot. The simulated location is then selected by moving a uniform distance from the center of the simulated spot. The effect of the simulation algorithm 9 can be seen in figures 5.9 and 5.10, where the first figure shows the login locations of an active user and the second figure shows the login locations of a stationary user.

Algorithm 9: Generate login location

```
Data: current user type (user_type), saved simulation spots (user_spots),
      venue distribution (V) and radius of different venues (R)
Result: Login location
cluster_id ← select cluster id from distribution V;
cluster_lat, cluster_lng ← extract saved simulation spot with id
  cluster_id from the current user;
radius ← get the radius of cluster with id cluster_id from R;
lat, lng ← move a random uniformly distributed distance $[0, radius]$ from
  position (cluster_lat, cluster_lng);
return lat, lng;
```

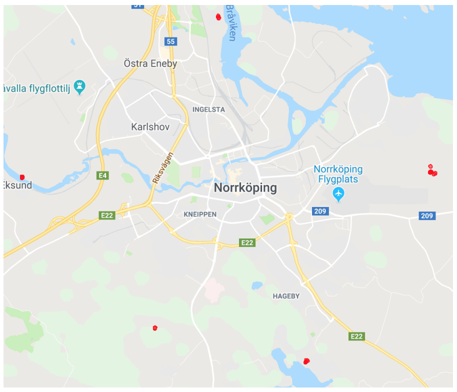


Figure 5.9: Login locations for an active user

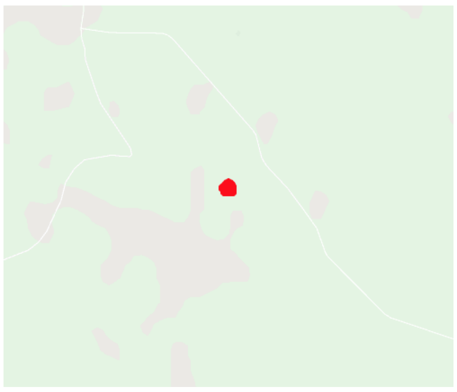


Figure 5.10: Login locations for a stationary user

5.2 Implementation

This section will give an architectural overview of the simulator, showing how the different metadata types are integrated into the simulator. A brief high level description of the package structure will be followed by a more in depth description of the individual packages in order to further explain the inner workings of the simulator.

5.2.1 Package overview

An overview of the outer package structure can be seen in figure 5.11. There exists 5 different important packages that together simulates the entirety of the service, as can be seen in figure. Each package has a well defined purpose, containing either simulation or filtering code.

The attribute package contains wrapper classes for the different metadata types. There exists both query related and database related classes for the different metadata types. This package is dependent on the database package.

The database package contains an implementation of a database containing enrolled users. The database package is dependant on the attribute package. The database implementation could be swapped out for any viable database implementation, as long as the required abstract methods are implemented.

The BTS package contains the filtering implementations and the template matching algorithms. This package uses the database and attribute packages in order to perform the filtering.

The simulation package is the core of the simulation, containing all simulation code for both enrollment and login requests. This package uses all other packages in order to simulate a growing number of enrolled users during the operation of the system.

The log package is a small package designated to perform collection of performance statistics of the filtering algorithms. This package is independent and is only used by the simulation package.

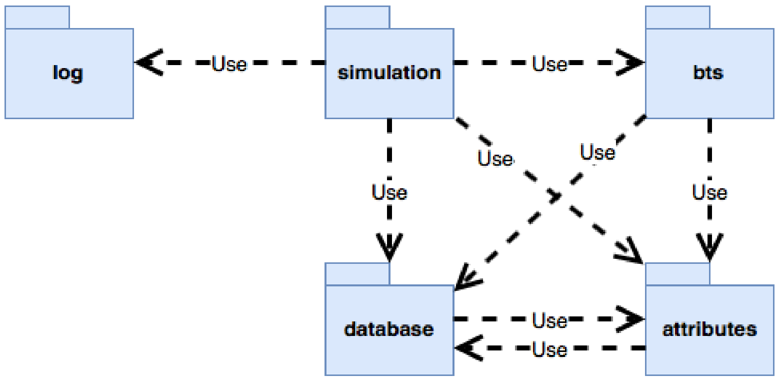


Figure 5.11: Package overview

5.2.2 Attributes package

As mentioned earlier, the attribute package contains classes related to the different metadata types, such as location and device ID. The structure of the package can be seen in figure 5.12. Each metadata type has one class representing a query value and a class representing the stored metadata value of the user. The `UserAttribute` classes can also contain history for some attributes, if it is needed for a filtering algorithm. The `QueryAttribute` classes contains the query values associated with an identification request as well as update methods that are called when a successful identification has been carried out. It is the update methods responsibility to correctly update the stored metadata value of the user by informing the database.

5.2.3 Database package

As previously mentioned, the database package contains a concrete implementation of a file database using the `Database` interface as well as a class representing a user, as can be seen in figure 5.13. The database is responsible for maintaining a list of the enrolled users. The metadata values are represented by `UserAttribute` classes from the attribute package. The `Database` interface contains methods for querying the database for certain users, enrolling users as well as updating existing users and their metadata values. The `FileDatabase` contains some indexes, enabling efficient database lookup.

The `User` class is a simple representation of an enrolled user. It contains the stored biometric template as well as a list of all the metadata attributes associated with a certain user.

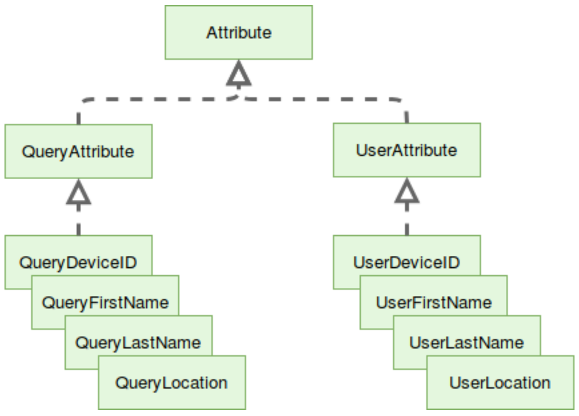


Figure 5.12: Overview of the attribute package

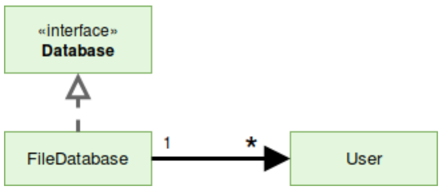


Figure 5.13: Overview of the database package

5.2.4 BTS package

The BTS package contains all filtering and matching related code. The `IDService` class is the centerpiece of the package, with a responsibility of supporting communication with other packages, as can be seen in figure 5.14. This class contains a list of different `Filter` classes that implements the filtering in their own way. When additional metadata is needed in order to filter, the class will query the `QueryService`. The implementation of `QueryService` will differ between simulation and when deployed, as when deployed the user has to be prompted to enter additional information and when in simulation, the query value is generated.

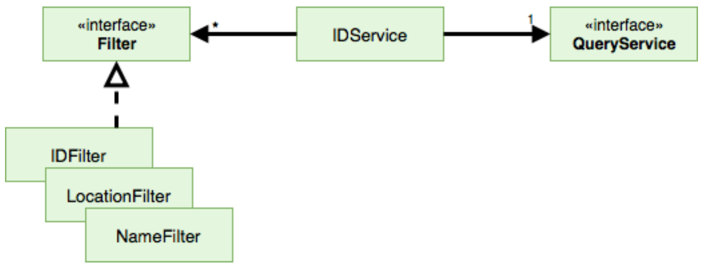


Figure 5.14: Overview of the BTS package

5.2.5 Simulation package

The simulation package is the core of the simulator, driving the entire simulation. The package contains an engine responsible for controlling the simulation by creating different tasks, as can be seen in figure 5.15. The two possible tasks are registration and login where registration corresponds to a user enrolling and login corresponds to an identification request. The login event uses the `QuerySimulation` class in order to simulate the user entering metadata and the registration event uses the `UserFactory` to generate realistic user metadata values. Each metadata factory contains the relevant enrollment and login simulation algorithms described earlier in section 5.1. The `QuerySimulator` is a concrete implementation of the interface `QueryService` from the BTS package.

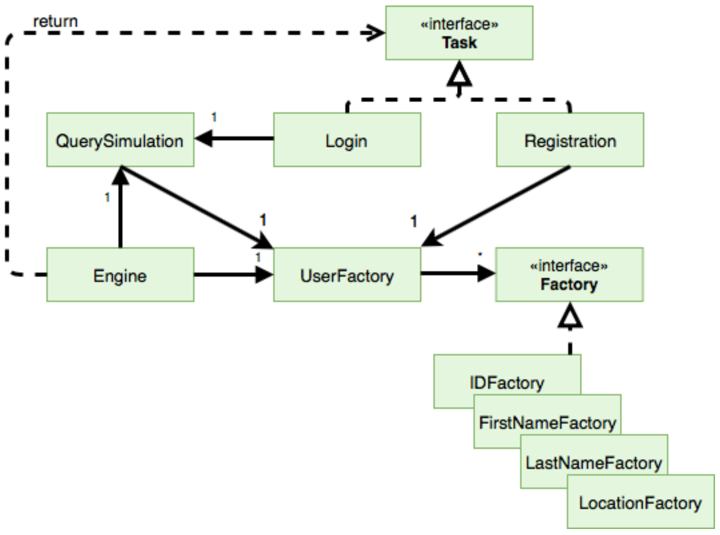


Figure 5.15: Overview of the simulation package

5.2.6 Log package

The log package is responsible for monitoring the performance of the filtering algorithms. As seen in figure 5.16, the `StatMonitor` class contains a list of different statistics to measure. It is possible to add extra statistical measurements by adding them to the list of statistics that the `StatMonitor` monitors. The `StatMonitor` communicates with the other packages in order to get measurements of different kinds and these measurements can be plotted for visual feedback of the performance.

5.2.7 Implementation discussion

The structure of the packages has been well thought out in order to enable complete separation between simulation and filtering code. This implies that the filtering code can be used without simulation code in a real deployment, as long as the

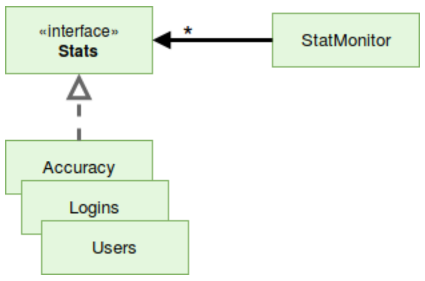


Figure 5.16: Overview of the log package

internal data flow remains. A real deployment would also need to consider implementing a new database class as a file database is not feasible for distributed systems.

It would be straightforward to implement new metadata types since the system is designed in a modular fashion. This would be done by implementing a filter as well as query and user attribute wrappers for the new metadata type.

If one would like to simulate the performance of the new data, a factory generating data from some distribution would need to be implemented.

Besides being easy to integrate new metadata types it is also very easy to adjust the current simulation parameters by changing them in the code in order to simulate different dynamics.

Chapter 6

Filter design & implementations

This chapter will use previous ideas discussed in chapter 4 in order to conceptualize filtering algorithms that can be tested using the simulator from chapter 5. The implemented algorithms will be a part of the BTS package as seen in figure 5.14, where each algorithm will be implementing the `Filter` interface. This modularity allows different filtering techniques to be evaluated individually and combined under the same circumstances. It has been decided that all filtering algorithms should be tested with both 50.000 and 200.000 enrolled users as well as with a continuously growing number of enrolled users in order to capture the dependency between performance and the number of enrolled users. Furthermore it has been decided that the accuracy should be measured as the probability that the correct candidate is placed within the set of the 50 most likely candidates returned by the filtering algorithm. This number has been decided by Fingerprints by taking the performance of the biometric matcher into consideration. The probability of a successful identification will be estimated by using the biometric matcher in order to be able to determine if the correct candidate is within the set of the most likely candidates.

Before considering how combinations of filters may be used for filtering, an analysis of unimodal filtering will be conducted in order to obtain a baseline measurement of what is possible to achieve. For every unimodal filtering algorithm, the implementation will be described before an evaluation is presented according to the previously described circumstances. This procedure will then be repeated for more advanced multimodal alternatives in order to investigate potential performance gains when combining filters.

6.1 Unimodal filtering

6.1.1 Device ID based filtering

As discussed in section 5.1.1, the discriminating power of a unique verified device ID is very high and very effective to use in a filter. To filter on device ID, the database must be queried for users that use a certain device ID. This can be done in multiple ways where the naive solution is to do a linear search of the database, comparing each candidates device ID. This search has a time complexity of $\mathcal{O}(n)$

and the performance decreases as n increases. To mitigate this, a simple index using a hash table have been implemented, where the keys are the device IDs and the values are the users that use that device ID. When users enroll or successfully authenticate with a new device ID, this index is updated in order to stay correct. On identification requests, the index can be used for efficient lookup with a time complexity of $\mathcal{O}(1)$ because of the underlying hash structure. As the amount of enrolled users grows, the index would still have the same time complexity but the memory needed to store the index would increase.

It should however be noted that using only device ID will not always work, as it would be impossible to filter out the right candidate if a user uses a new device with a new device ID in an identification. This insight leads to the conclusion that there must exist a fallback strategy that could be used if the device ID based filtering fails.

Filter implementation

Algorithm 10 contains the pseudocode of the proposed idea, implementing an efficient hash based lookup on the device ID associated with the identification request. The algorithm works by simply returning the candidates known to have used the device associated with the identification request.

Algorithm 10: Device ID based filtering

```
Data: candidate set ($C$) and device ID (query_id) associated with the
      identification request
Result: Filtered candidate set $R(\Gamma)$
candidates ← extract candidates with a previous device ID query_id from
  $C$;
return candidates;
```

Result

Figure 6.1 shows the accuracy of the filtering algorithm on the two test cases. As can be seen in the figure both cases converges towards an accuracy of approximately 80%. Figure 6.2 shows the accuracy of filtering on device ID as the number of enrolled users grows.

As these results are heavily dependant on the simulation parameters shown in algorithm 2, it can be inferred that the accuracy will be approximately the probability that a user uses the same device ID between logins. Changing the simulation parameter will cause the results shown in figures 6.1 and 6.2 to change accordingly.

The oscillations seen in figure 6.2 are caused by a relatively small number of samples for each accuracy measurement. It is therefore more enlightening to pay attention to the trend displayed as a orange line in order to see the general accuracy development. This is also the case for all other filter implementations.

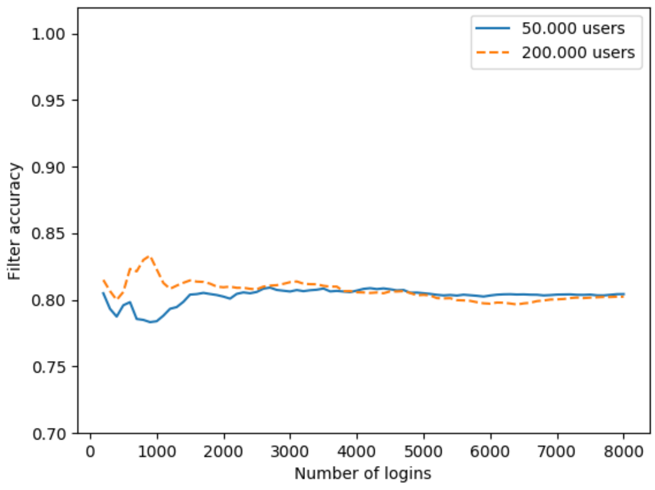


Figure 6.1: Top-50 accuracy with 50.000 and 200.000 enrolled users, using device ID

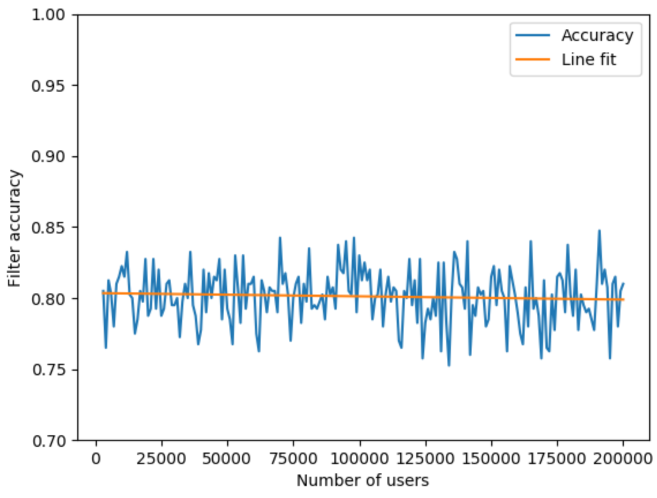


Figure 6.2: Top-50 accuracy with a growing database, using device ID

6.1.2 Name based filtering

As has been discussed in section 5.1.3, names comes with a high entropy allowing for an efficient filtering algorithm to be implemented by comparing the similarity

Name	2-grams
Marcus	ma, ar, rc, cu, us
Niklas	ni, ik, kl, la, as
Christian	ch, hr, ri, is, st, ti, ia, an

Table 6.1: Example of n-grams from names with $n = 2$

	N=2	N=3	N=4	N=5
Recall	0.9997	0.9865	0.9073	0.8483
Relative set size	51.1%	23.5%	15.5%	3.8%

Table 6.2: Mean recall and relative set size for different N values

between a pair of names. This comparison can be done by comparing the name associated with the identification request to each candidates name in the database. The similarity measurement used for comparing a name pair can be of different kinds, as discussed in section 4.1.2. This approach is however very naive, resulting in a high execution time since it will need to consider every candidate in the database. One solution proposed by Yang et al. [53] is to precompute a n-gram dictionary mapping each possible n-gram to the set of candidates containing the n-gram. This means that the BTS database could precompute one n-gram dictionary for first names and one for surnames. Table 6.1 shows example of names and the extracted n-grams. These dictionaries could then be used in a preprossesing stage, only selecting candidates sharing at least one n-gram with the name associated with the query.

Filter implementation

The filter implementation consists of two filtering phases, which can be seen in algorithm 12. In the first phase, the candidate set is prefiltered using n-grams to find candidates that are roughly similar. This phase could be seen as a retrieval stage and should therefore be more conservative in the filtering in order to maximize the chances of the genuine user remaining in the candidate set. The algorithm uses a parameter n in order to create n-grams from the first and surname of the identification request. Using different values for n may affect the performance of the algorithm and an investigation of the implications of different n-values was carried out. Table 6.2 shows the recall rate and relative set size of the candidate set after the prefiltering phase using different n-values and it can be seen that there exists a trade-off between recall rate and set reduction. The n-value was chosen to be 2 in order to maximize the recall rate as it is more important that the genuine user remains in the prefiltered set, rather than reducing the cardinality of said set.

In the second phase, a more advanced filtering takes place where the query name is compared against all the candidates that passed through the first filtering phase. The comparison is done using a Jaro-Winkler similarity measure (algorithm 11), as discussed in section 4.1.2. This measurement gives a score between 0 and 1, where 1 represents equality and 0 represents completely different strings. Users are added to the filtered candidate set if the comparison result is above the specific users typing threshold. This threshold is updated upon successful identification in order to represent how exact a user is when he/she types in his/her name. This threshold is used to allow users that perform many typing errors to still be considered in the filtering. Lastly the candidates are ranked on the similarity measurement of the query name and the users stored name in order to output a ranked set of candidates.

Algorithm 11: name_sim

```
Data: name1, name2
Result: Similarity measurement $\in [0, 1]$
return jaro_winkler(name1, name2);
```

Algorithm 12: Name based filtering

```
Data: candidate set ($C$) and name (first_name, last_name) associated
      with the identification request
Result: Filtered candidate set $R(\Gamma)$
fn_grams ← extract n-grams of first_name;
ln_grams ← extract n-grams of last_name;
cands ← all candidates in $C$ sharing at least one n-gram with either
  first_name or last_name;
score_vector ← (candidate, 0) for every candidate in cands;
for cand in cands do
    cand_score = name_sim(cand.first_name, first_name) +
      name_sim(cand.last_name, last_name);
    if cand_score > cand.spelling_threshold then
        update score_vector for cand with score cand_score;
end
return sorted score_vector;
```

Results

The accuracy difference between 50.000 and 200.000 enrolled users can be seen in figure 6.3. The accuracy drops as the number of enrolled users increases, as visualized by the line fit of the accuracy in figure 6.4. This implies that using solely name for filtering purposes will not be sufficient as the number of enrolled users grows large.

6.1.3 Location based filtering

From previously presented studies in section 5.1.4, a users' movement pattern has been shown to be deterministic to some extent. This finding together with the availability aspect of location collection is the primary motivation for implementing a location based filter.

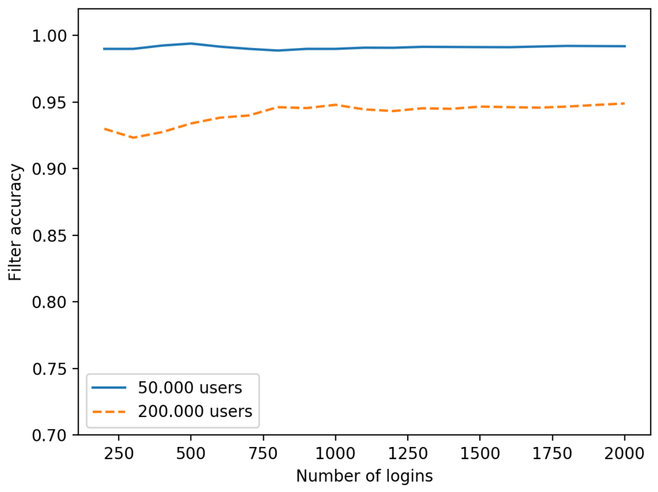


Figure 6.3: Top-50 accuracy with 50.000 and 200.000 enrolled users, using name

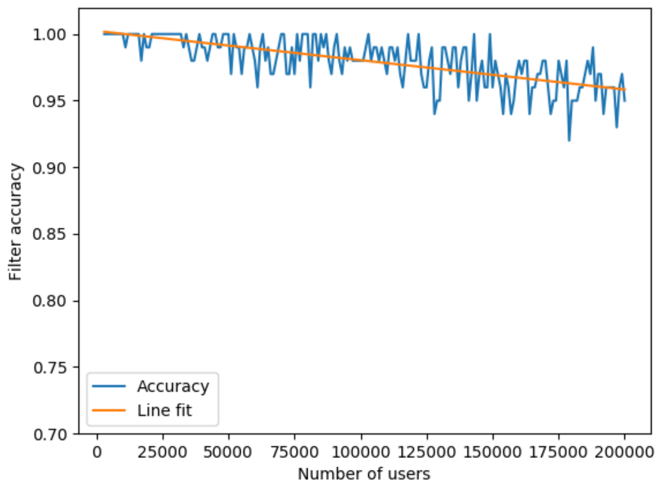


Figure 6.4: Top-50 accuracy with a growing database, using name

When an identification request with location metadata is sent to the BTS, both spatial and temporal is available in the request. The spatial data is represented by the location from the request and the temporal data is represented by the time of the request. The combination of these features are known as spatio-temporal data and has been shown by Rossi et al. [48] to be important features to include in an identification procedure. This means that an optimal identification model should include the spatio-temporal features in some way in order to achieve peak performance. There has been a lot of research in this area and many different approaches have been shown to be successful. Some examples include a machine learning approach that learns to combine the spatio-temporal features from existing training data and a more traditional approach using transition probability matrices from location history to predict position [45, 54]. Another approach by Umair et al. [55] uses location traces to determine significant places of users and to predict future user location. Location traces can be very efficient for filtering applications, however they rely on stable and continuous collection of location points and this collection would have to be enforced by the application that wishes to use the identification service.

One way to include the spatial information into the identification procedure is to utilize the fact that users tend to have a limited number of significant locations. This means that it should be possible to perform a location based filtering by ranking the candidates by their probability to appear at the query location. One concrete approach suggested by Zhao et al. [56] is to use a kernel in order to approximate the location density distribution. One huge advantage with this approach is that it works for arbitrary underlying distributions according to the authors.

In order to filter out candidates that are not close to some location, a search within the database must be carried out. An exhaustive search for candidates that are within a certain distance of a query location is an operation with time complexity $\mathcal{O}(n)$ and this method grows more and more inefficient as the number of enrolled users increases. To counter this a spatial index called R-tree has been used [57]. R-trees (rectangle trees) are able to perform searches with a time complexity of $\mathcal{O}(\log n)$ by using a underlying data structure that groups the data by bounding rectangles. R-trees are balanced tree structures where each node in the tree has a bounding rectangle that contains all child nodes bounding rectangles, as illustrated by figure 6.5. All enrolled users will have a bounding rectangle in the R-tree, representing their last known location. When a search is performed, a query rectangle is created from the query location and all candidates which bounding rectangle intersects the query rectangle is returned from the search, ensuring that only candidates that are sufficiently close to the query location are considered for further filtering.

Filter implementation

The location based filtering algorithm is divided into the two consecutive phases, pre-filtering and density filtering which will be described separately below.

The prefiltering uses the previously discussed R-tree index to efficiently prefilter the candidate set in order to enable more complex and expensive search to be carried out on the remaining candidates. Algorithm 13 describes the prefiltering procedure where candidates are only preserved for the second phase if they are

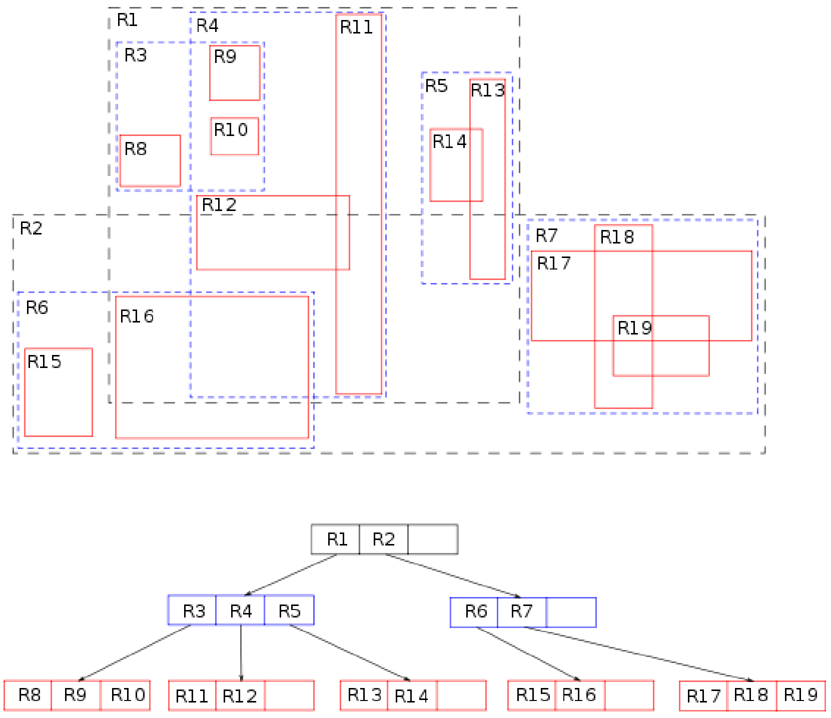


Figure 6.5: Illustrating the data structure behind R-trees [58]

within a square with sides of 30km of the query location. By using this bounding rectangle, the search space can be reduced. The efficiency of the reduction depends on where the query location is located. If it is located in a densely populated area it will be less efficient compared to a sparsely populated area. Figure 6.6 shows the efficiency of the the prefiltering on location bounds using R-trees. As can be seen in the figure, the relative filtered set size is bounded by 17.5%. This is a significant reduction, enabling more advanced location similarity computations using the same resources.

Algorithm 13: Location based prefiltering

```
Data: candidate set ($C$) and query location (qlat, qlng) associated with
      the identification request
Result: Filtered candidate set $R(\Gamma)$
lat_low, lng_low, lat_high, lng_high ← approximate a circle of radius
  30km located at position (qlat, qlng) with a rectangle;
candidates ← get users with a known last location within the rectangle
  (lat_low, lng_low, lat_high, lng_high) using the R-tree implementation;
return candidates;
```

The second phase builds on the suggested approach of using estimated density distributions in order to perform a ranking of the candidates. The distributions

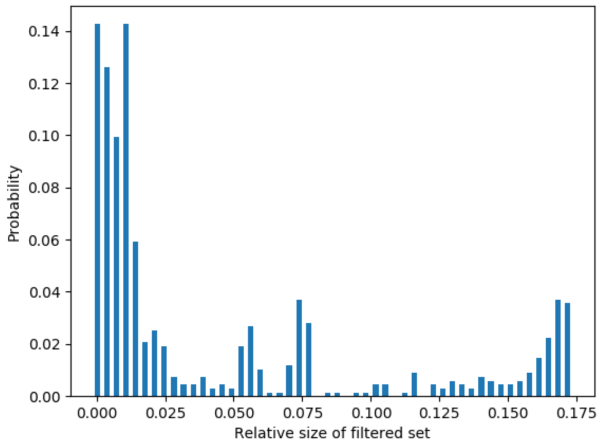
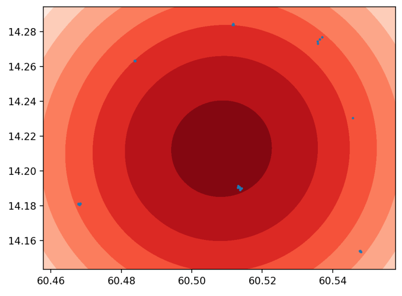


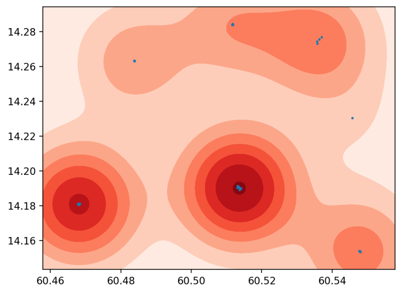
Figure 6.6: Illustrating the relative set size after simple location filter

are estimated by using a Density Kernel from the machine learning library Scikit-learn [24]. The algorithm works by estimating a density distribution from the location history for each candidate which then is used to obtain an approximated probability of each candidate appearing at the query location. These probabilities are then used in a simple ranking procedure, as can be seen in algorithm 14. One drawback with using a kernel is that they require some tuning of the bandwidth parameter in order to achieve optimal performance. One common solution to the bandwidth problem is to perform a grid search over the hyper-parameter space, which basically means that a subset of the possible parameters are tested in order to find a good set of hyper-parameters. This approach is feasible in our case since the hyper-parameter space is one dimensional and the simulator is able to provide an accuracy measurement for different hyper-parameters.

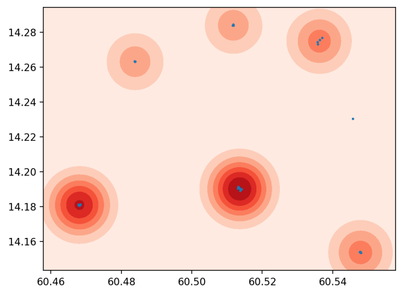
Figure 6.7 illustrates how the selected bandwidth affects the estimated density distribution for a user. The blue points in the figure corresponds to previous locations of the users and a dark red color corresponds to a higher density than a brighter color. It can be seen in the figure that a large bandwidth tries to capture the users location history on a inter-cluster level more than a small bandwidth. A bandwidth of 0.001 is clearly not optimal since it fails to capture intra-cluster information. A bandwidth of 0.0002 has the advantage of being able to capture both intra- and inter cluster information. The bandwidths 0.0001 and 0.00001 are more discriminating and may be able to provide a more optimal accuracy assuming that users are close to their significant places. The used value for the ranking process is obtained by obtaining the value from the density distribution at the query location, which means that a candidate will be given a higher score if the query location is close to some of the candidates clusters. The kernel size was selected to 0.00001 since it produced the best accuracy.



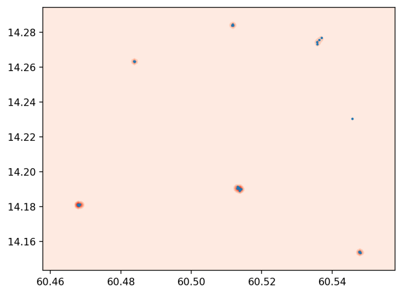
(a) Kernel with a bandwidth of 0.001



(b) Kernel with a bandwidth of 0.0002



(c) Kernel with a bandwidth of 0.0001



(d) Kernel with a bandwidth of 0.00001

Figure 6.7: Illustration of different kernel bandwidths

Algorithm 14: Location based filtering

```
Data: Pre-filtered candidate set ($C$) and query location (qlat, qlng)
      associated with the identification request
Result: Filtered candidate set $R(\Gamma)$
candidate_scores ← [];
for cand in C do
    cand_kernel ← get location kernel associated with cand;
    score ← use cand_kernel to extract score of the query location;
    append (cand, score) to candidate_scores;
end
sort candidate_scores by scores;
filtered_cands ← extract candidates from candidate_scores;
return filtered_cands;
```

Results

Figure 6.8 shows that the accuracy between 50.000 and 200.000 users. There exists just a small difference between, implying that the algorithm is able to perform well for even 200.000 enrolled users. Figure 6.9 shows the accuracy as the number of enrolled users grows. As expected there is a slight decrease in accuracy as the users increases. These results where collected assuming a location history of at least 30 points per user. Figure 6.10 and 6.11 shows the accuracy collected with

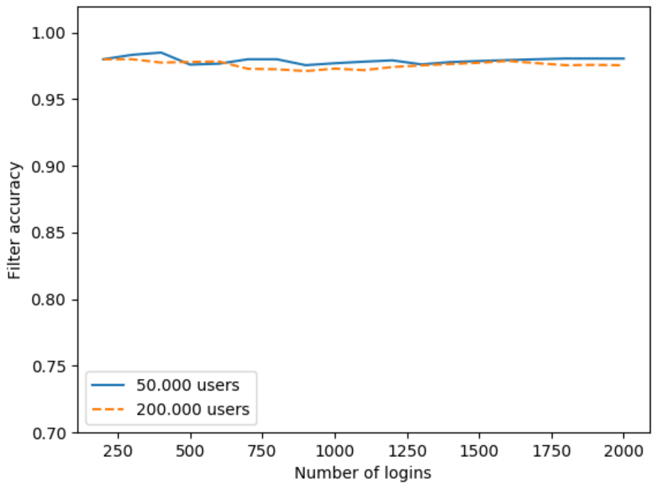


Figure 6.8: Top-50 accuracy with 50.000 and 200.000 enrolled users, using 30 location history points

a location history of 10 points. As seen in the figures, the accuracy has decreased

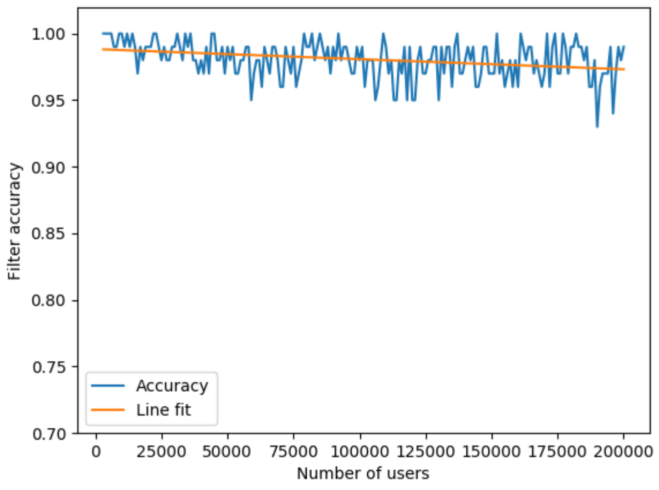


Figure 6.9: Top-50 accuracy with a growing database, using 30 location history points

compared to the results collected with a location history of 30 points. The model is now however not able to handle the different amount of users with the same accuracy.

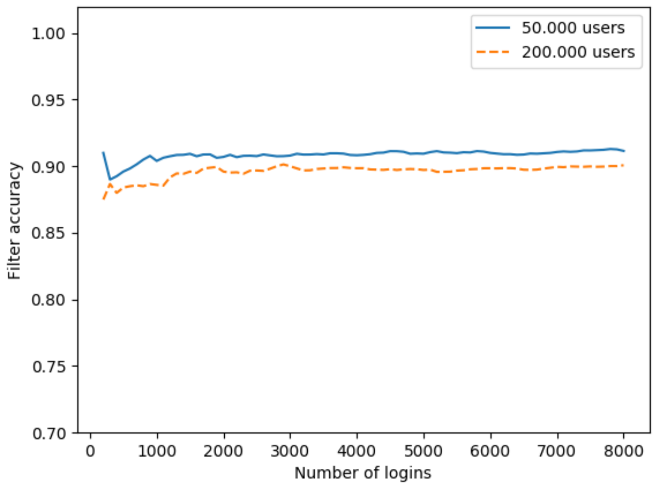


Figure 6.10: Top-50 accuracy with 50.000 and 200.000 enrolled users, using 10 location history points

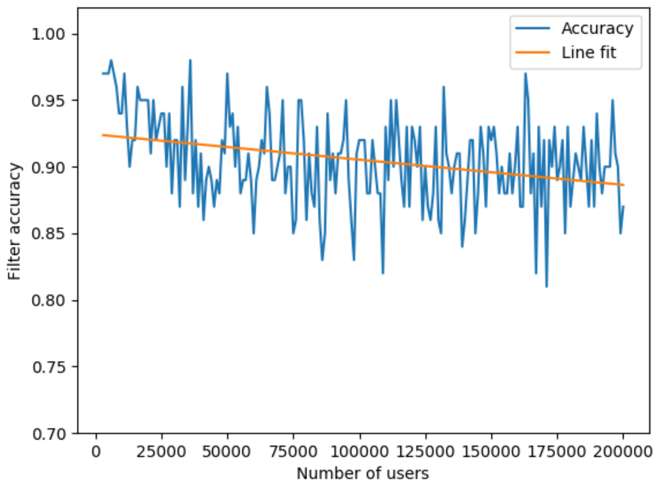


Figure 6.11: Top-50 accuracy with a growing database, using 10 location history points

It can be concluded by looking at the figures that location is an important metadata type to include in an identification procedure and should therefore be used whenever available. It should also be noted that the accuracy is dependent on length of the location history, leading to the conclusion that location should preferably be used together with some other metadata type in order to achieve a more reliable identification.

It shall be noted that these results depend on the simulation of locations associated with the identification requests, as described in section 5.1.4. In most cases, one of a users significant places with some noise is used as the query location, which favours the selected kernel bandwidth.

6.2 Multimodal filtering

6.2.1 Location and age based filtering

As concluded in the section 6.1.3, location information is not sufficiently effective when the location history of a user is sparse. To mitigate the accuracy loss for this scenario, a combination with another metadata type could be used. Age is a metadata type with relatively low privacy concerns when stored in the cloud. Using age alone would not result in efficient filtering as there exists to few different values in order to filter out a large number of enrolled users. However when used in conjunction with location information, the discriminating power of the two may result in an accuracy boost.

Filter implementation

The implementation consists of the three phases, rough location lookup, age filtering and location filtering, as can be seen in algorithm 15. The first phase uses the R-tree algorithm as described in section 6.1.3 in order to efficiently reduce the number of candidates before more costly computations is carried out. The second phase performs age based filtering by removing candidates with an age difference larger than 1 of the user associated with the identification request. The age based filtering can be seen in algorithm 16. The final phase is performed by using the density based filtering algorithm, as described in section 6.1.3.

Algorithm 15: Location and age based filtering

```
Data: Candidate set ($C$), query location (qlat, qlng) and query age (qage)
      associated with the identification request
Result: Filtered candidate set $R(\Gamma)$
candidates $\leftarrow$ algorithm-13($C$, qlat, qlng);
candidates $\leftarrow$ algorithm-16(candidates, qage);
return algorithm-14(candidates, qlat, qlng);
```

Algorithm 16: Age based filtering

```
Data: Candidate set ($C$) and query age (qage) associated with the
      identification request
Result: Filtered candidate set $R(\Gamma)$
candidates $\leftarrow$ candidates from $C$ with an age within 1 year from the query
  age (qage);
return candidates;
```

Result

As seen in figure 6.12 there exists an expected difference in accuracy between 50.000 and 200.000 number of enrolled users. The difference is expected as the combination may fail to identify the correct user if the user uses the service in a densely populated area where it is likely that a lot of people share the same age. Figure 6.13 shows a decrease in accuracy as the number of enrolled users increases. This result verifies the accuracy difference seen in figure 6.12.

6.2.2 Age and name based filtering

As seen in the results in 6.1.2, the discriminating power of a name may fail to capture the genuine user if the number of enrolled users is large. However if an additional metadata type is used together with name an increase in efficiency is possible. The idea is to use the two manually collected metadata types age and name together in order to increase the probability of finding the genuine user in a large set of enrolled users.

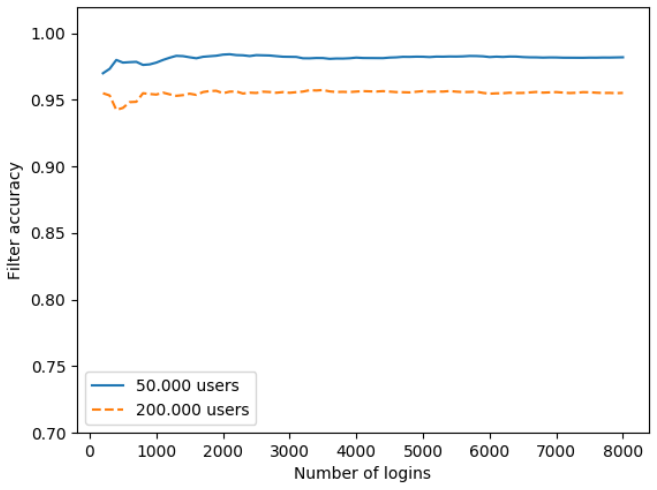


Figure 6.12: Top-50 accuracy with 50.000 and 200.000 enrolled users, using age and location (10 history points)

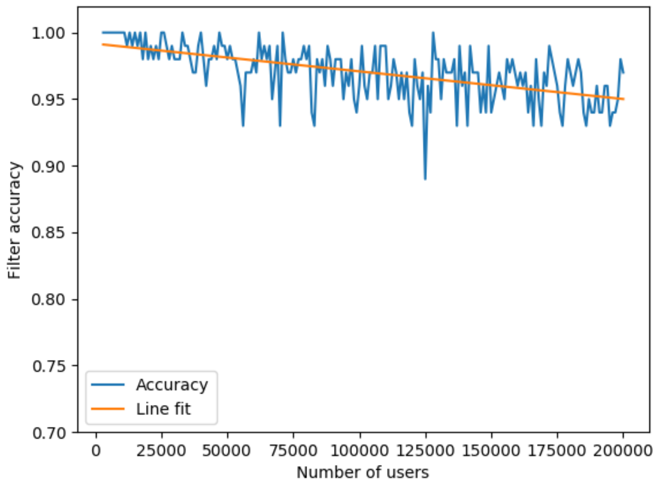


Figure 6.13: Top-50 accuracy with a growing database, using age and location (10 history points)

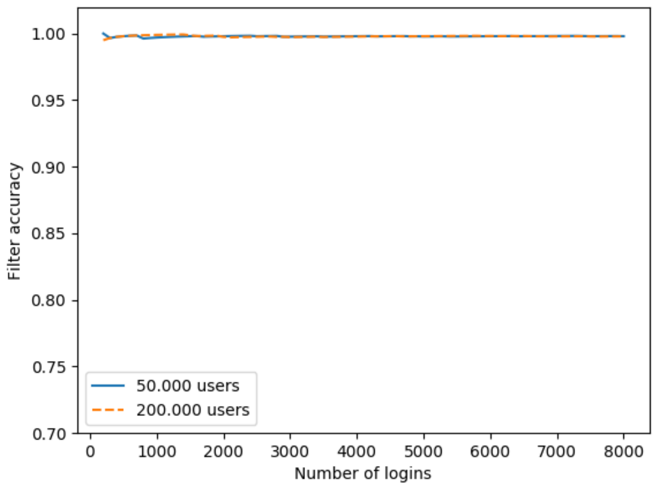


Figure 6.14: Top-50 accuracy with 50.000 and 200.000 enrolled users, using name and age

Filter implementation

The implementation is done using the cascade filtering technique, as discussed in section 4.2.2. As shown in algorithm 17, the candidates are first filtered on their age as this is a efficient lookup in the database. The candidates that are within 1 year away from the query age are then used in a name comparison and a similarity score is calculated for each name. The algorithm returns the sorted similarity score vector in descending order.

Algorithm 17: Age and name based filtering

```
Data: Candidate set ($C$), age (qage) and first and surname (first_name,
      last_name) associated with the identification request
Result: Filtered candidate set $R(\Gamma)$
age_candidates $\leftarrow$ algorithm-16(candidates, qage);
name_scores $\leftarrow$ algorithm-12(age_candidates, first_name, last_name);
return sorted name_scores vector;
```

Result

The results seen in figure 6.14 are expected as there is an added layer of filtering upon the results that can be seen in figure 6.1.2. By cascading age together with name, the worst case uniqueness is lowered by the factor of the worst case uniqueness of age. The result in figure 6.15 is also expected as there is close to no difference in the accuracy between 50.000 and 200.000 enrolled users.

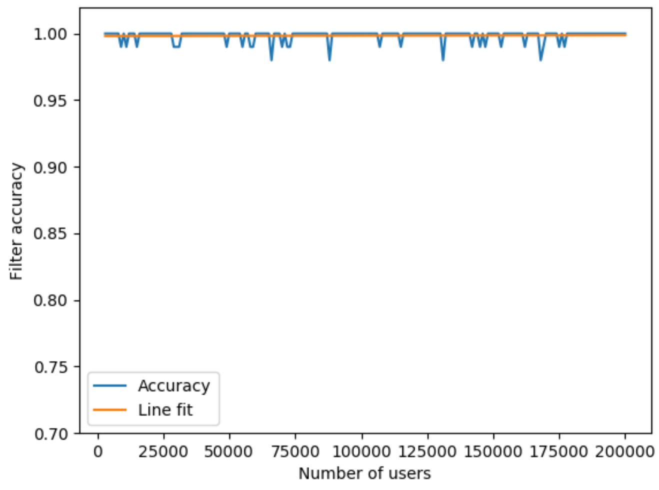


Figure 6.15: Top-50 accuracy with a growing database, using name and age

6.2.3 Location and name based filtering

The previous filtering combination combined age and name in order to achieve better accuracy than the name filter was able to achieve on its own. This filtering combination is another alternative and uses location information instead of age information. This combination has the advantage that locations can be collected automatically and can therefore be considered to be more user friendly.

Filter implementation

The filtering algorithm as seen in algorithm 18, builds on the idea of applying different filtering algorithms in a sequential order. The first phase of the algorithm consists of utilizing the location based preprocessing algorithm, as described in section 5.1.4, in order to efficiently reduce the number of candidates in an early stage. The next phase consists of applying the name filter, as described in section 5.1.3. This phase produces one score for each candidate which is saved for later use. This procedure is then repeated for the density based algorithm, as described in section 5.1.4. The next step is then to normalize the scores by using a normalization procedure, as can be seen in algorithm 19. This procedure is performed in order to make sure that the scores are within the same range. The normalized scores are then fused for each candidate by using a summation in order to obtain one score for each candidate. The last step of the algorithm is then to rank the candidates by their fused score in order to obtain a ranking of the candidates.

Algorithm 18: Location and name based filtering

```
Data: Candidate set ($C$), name (first_name, last_name) and query
      location (qlat, qlng) associated with the identification request
Result: Filtered candidate set $R(\Gamma)$
candidates ← algorithm-13($C$, qlat, qlng);
name_candidates ← algorithm-12($C$, first_name, last_name);
location_candidates ← algorithm-14(candidates, qlat, qlng);
name_scores ← extract scores from name_candidates;
location_scores ← extract scores from location_candidates;
name_scores ← algorithm-19(name_scores);
location_scores ← algorithm-19(location_scores);
candidate_scores ← [];
for cand in candidates do
    cand_loc_score ← extract score of cand from location_scores;
    cand_name_score ← extract score of cand from name_scores;
    score ← cand_loc_score + cand_name_score;
    append (cand, score) candidate_scores;
end
sort candidate_scores by score;
candidates ← extract candidates from candidate_scores;
return candidates;
```

Algorithm 19: Score normalization method

```
Data: Scores (scores) to normalize
Result: Normalized scores
mean ← compute mean of scores;
std ← compute std of scores;
scores ← [];
for score in scores do
    score ← $\frac{score-mean}{std}$;
end
return scores;
```

Result

It can be seen by looking at figure 6.16 and 6.17, that a combination of name and location information is able to provide excellent accuracy. The accuracy is comparable with the accuracy result of the previous filter but has the advantage of only requiring manual name collection. The filter is also effective for small as well as large candidate sets.

6.2.4 Location, name and age based filtering

This filter combines previous filtering implementations in an attempt to reach perfect accuracy. The filter uses location, name and age information associated with the identification request.

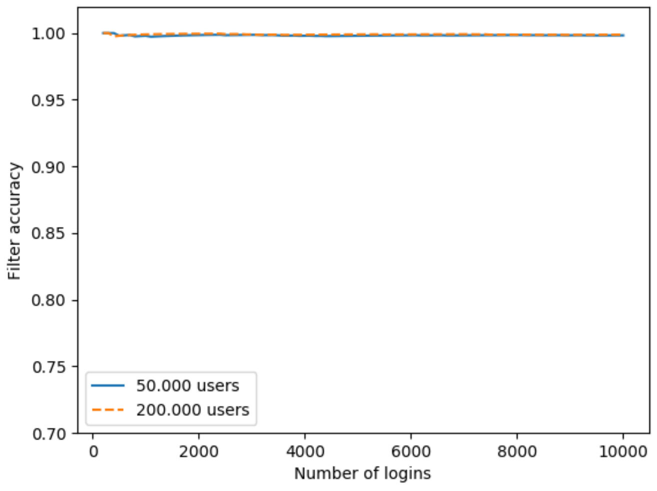


Figure 6.16: Top-50 accuracy with 50.000 and 200.000 enrolled users, using age and location

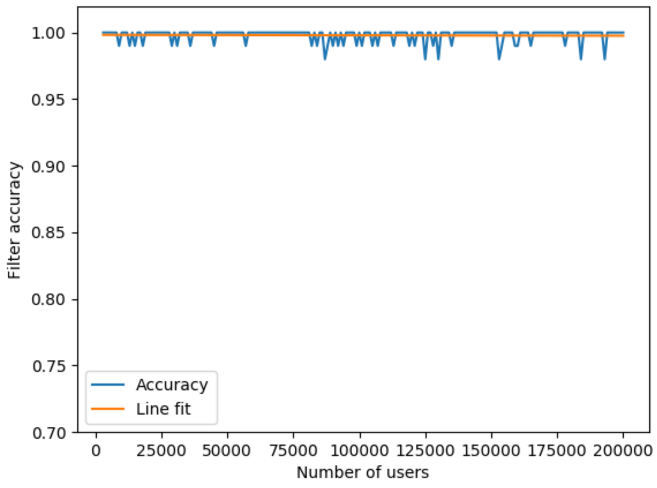


Figure 6.17: Top-50 accuracy with a growing database, using age and location

Filter implementation

The filter implementation uses previously implemented filters in order to successfully combine the three metadata types, as can be seen in algorithm 20. The first step of the algorithm is to utilize the location based prefiltering in order to efficiently reduce the number of candidates. The next step is to use the age filter in order to remove candidates with a different age before lastly combining name and location information.

```
Algorithm 20: Location, name and age based filtering
  Data: Candidate set ($C$), name (first_name, last_name) and query
        location (qlat, qlng) associated with the identification request
  Result: Filtered candidate set $R(\Gamma)$
  candidates ← algorithm-13($C$, qlat, qlng);
  candidates ← algorithm-16(candidates, qage);
  return algorithm-18(candidates, first_name, last_name, qlat, qlng);
```

Result

The results seen in figure 6.18 shows a very minor increase in accuracy when compared to previous multimodal filtering techniques. The accuracy is approximated to 0.9988 and there is close to no difference between 50.000 and 200.000 enrolled users. Figure 6.19 shows that this combination would be effective for a very large number of enrolled users.

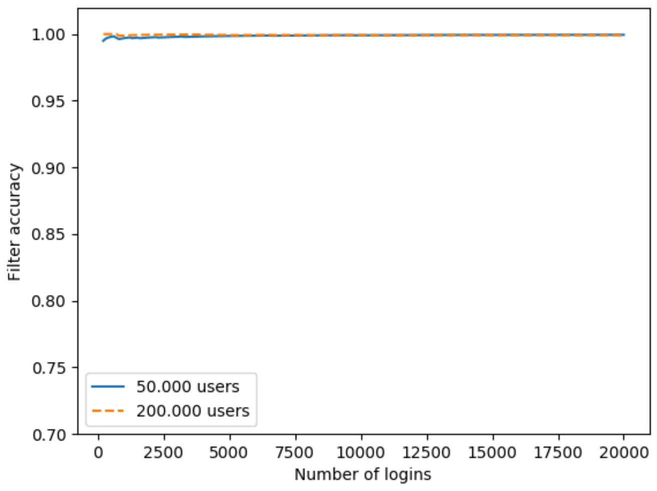


Figure 6.18: Top-50 accuracy with 50.000 and 200.000 enrolled users, using name, location (10 history points) and age

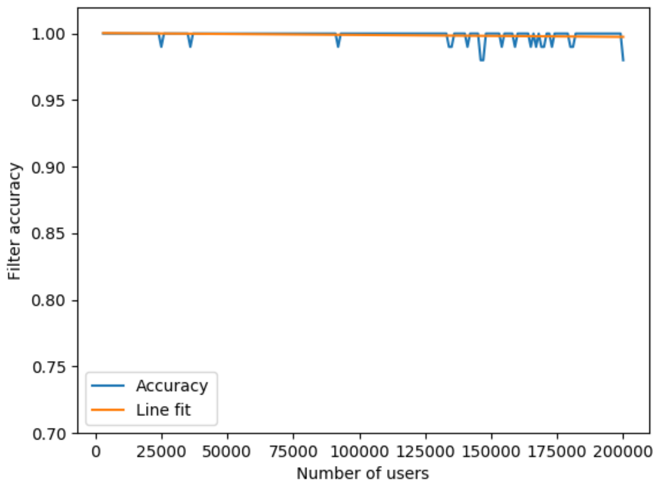


Figure 6.19: Top-50 accuracy with a growing database, using name, location (10 history points) and age

6.3 Result overview

An overview of the accuracy of all discussed uni- and multimodal filtering techniques is shown in table 6.3. The table shows accuracies for both 50000 and 200000 number of enrolled users.

Table 6.3: Result overview for different filter combinations

Filter	**50.000**	**200.000**
Device ID	0.8044	0.8024
Name	0.9920	0.9490
Location (30 points)	0.9805	0.9755
Location (10 points)	0.9115	0.8964
Location & Age	0.9819	0.9553
Location & Name	0.9988	0.9985
Location & Age & Name	0.9995	0.9992
Age & Name	0.9980	0.9979

Chapter 7

Security evaluation

The main purpose of the proposed system is to perform a reliable identification given a large number of enrolled users. Two crucial features of an ideal system is to only allow genuine users to identify successfully and to only reject imposters during identification. The ambition is therefore to come as close as possible to an ideal system by making sure that the FAR and FRR is as small as possible. This target together with the proposed data collection motivates a security evaluation done on the entire system in order to fully understand potential security threats. Figure 7.1 illustrates the proposed architecture and can be seen to consist of three subsystems. Each subsystem is connected to another subsystem over the Internet where the communication link as well as the individual subsystems could be a target for a malicious attack. Several different security threats together with their implications and mitigation strategies will be discussed in the following sections.

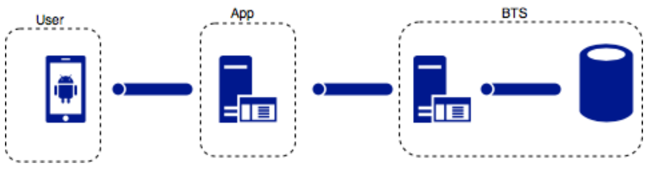


Figure 7.1: Illustration of the proposed system architecture

7.1 Unauthorized identification

One of the most serious security threats is unauthorized successful identifications by imposter users. Whether it being on purpose or not, the consequences could be devastating to the affected genuine user. An occurrence of such an attack would make the purpose of the system void and would negatively impact the trustworthiness of the system and the service provider.

The performance of identification systems are commonly measured in FAR and FRR, where the goal is to keep both as low as possible, as discussed in section 2.5. These measurements are affected by the different subsystems and it is therefore an requirement that each subsystem performs as good as possible in order to ensure the performance of the entire system. As the subsystems are used in cascade, it should be noted that the performance of the entire system is bounded by the worst performance of any individual subsystem.

The first subsystem in the identification procedure is the biometric sensor, creating a template from a scanned fingerprint. The sensors used in the system will have different FAR and FRR values affecting the performance of the identification. The next subsystem is the application, responsible for collecting and relaying metadata to the BTS. The application can lower the FAR and FRR values by including additional metadata in the identification request sent to the BTS. The final subsystem is the BTS, responsible for filtering and matching of biometric templates. The identification request is first handled by the filtering procedure, where metadata from the request is used to filter candidates before biometric matching is carried out. The filtering procedure is responsible to relay a ranked candidate set of at most 50 candidates to the biometric matcher. The performance of the BTS subsystem is therefore affected by both the filtering procedure as well as the biometric matcher. The performance of the biometric matcher is not considered further and an emphasis will be placed on the filtering procedure.

A common attack on an identification systems is to try several different combinations of credentials until a successful combination is found. In the case of the proposed system it would mean that different metadata types in conjunction with different templates could be tried in order to gain unauthorized access. The goal with such an attack is either to identify as a random or specific enrolled user. Based on the assumption that the device ID is protected using digital signatures, it can be assumed that any biometric template that is generated by a trusted device gets verified before the identification request reaches the filtering procedure. Thus making it harder for adversaries as they would have to breach the physical sensor and modify the inner workings before being able to send spoofed templates.

7.1.1 Random identification attack

The random identification attack is carried about by the procedure described below:

1. Acquire fingerprint template representing an unenrolled users fingerprint.
2. Generate some metadata combination
3. Generate an identification request from the generated metadata and the fingerprint template
4. Send identification request to BTS.
5. In case of an unsuccessful request go back to step 2.

The goal of this procedure is to be identified as a random user, thus gaining control over the users account. The above attack exploits the fact that the biometric matcher has a non zero FAR. At the time of writing, a typical FAR of a fingerprint for a small sensor range between $\frac{1}{200000}$ and $\frac{1}{50000}$, thus making it possible for an imposter to identify as someone else if enough template combinations are tried.

The protection against this attack should be composed of several mitigation strategies in order to build a strong defence. One possible mitigation strategy is to impose rate limits, limiting the number of identification requests that could be sent from a specific device ID over some time period. This limit should be individual and dynamically updated as different device ID's can exhibit different behaviours. This means that one compromised sensor can be blocked if malicious behavior is detected.

Another mitigation strategy is to implement a CAPTCHA challenge-response test in case that anomalies in requests are detected [59]. If a device ID or a specific user consecutively identifies unsuccessfully, a CAPTCHA challenge is required to be answered. If an answer is missing or several incorrect responses are given, a possible attack is unmasked and countermeasures can be taken. This prevention is only taken against extreme cases and it would therefore not affect normal usage.

A third mitigation strategy could be to use previous statistics gathered in order to build a classifier, potentially classifying incoming identification requests as malicious if they exhibit some learned patterns [60]. This strategy builds on the assumption that representative statistics are available and it is therefore a good idea to record data from successful identification requests as well as unsuccessful identification requests. An alternative approach could be to use an online learning algorithm, as suggested by Lampesberger et al. [61]. This approach still requires a teacher, providing some level of feedback in order to work. The advantage with this approach is that the feedback can come incrementally, not requiring a large training set in order to be deployed.

A final mitigation strategy could be to require several metadata types in the filtering process. This mitigation is based on the fact that more requests will be needed in order to try all combinations. It should however be noted that this strategy could place unnecessary burden on the users.

7.1.2 Specific identification attack

This attack is mitigated by verifying the biometric templates before they reach the filtering procedure, making it hard to gain access to a specific user by trying different templates. All discussed mitigation strategies in the previous section could also be deployed in order to further protect against this attack. E.g. if a specific user has several consecutive unsuccessful identification attempts, he/she could be temporarily banned and notified about the potential attack. After some intervention by the genuine user, the ban can be lifted.

7.2 Leak

It is assumed in a leak attack that an adversary has gained access to all data recorded in the BTS database by some means. It will not be discussed further how such an access can be obtained since it depends on the environment of the deployed system.

It is furthermore assumed that the leaked data consists of raw data in an unencrypted form, allowing an adversary to infer patterns. Some possible attack scenarios as well as their implications is discussed below.

- **User registrations:** The database contains information about all enrolled users and what services they are registered to. This information can be used to track peoples habits as well as enabling possible social engineering attacks where an adversary can contact the services impersonating the genuine affected user and perform malicious acts.
- **Location patterns:** The raw location data contains much information which could be used in order to infer user patterns. One example of an attack could be to extract secret locations often visited by a celebrity.
- **Information extraction:** As the BTS may contain sensitive personal data, this data may be used to breach the privacy of users. One attack may be to use collected metadata in order to blackmail and another attack may be to extract certain user behaviour believed to be sensitive.

The protection against a leak attack should render a leaked BTS database useless to an adversary. A common strategy to achieve this goal is to encrypt all data with a securely stored secret key before storing it in the database. By doing so, any leaked data will appear as nonsense to an adversary. This approach relies on that the decryption key itself is not leaked, otherwise the protection is void.

It is also important to consider the physical security of the servers by making sure that only authorized personnel are granted access. There should also be security protocols in place in order to make sure that authorized personnel are unable to steal data by physical means, such as stealing hard drives.

A final mitigation strategy is to transform sensitive data before it is stored in the database. By doing so, even if the data is leaked in clear text it is hard for an adversary to make sense out of it. As location data is considered very sensitive information, chapter 8 is dedicated to provide detailed explanations of threats and transformations of location data in order to protect the users integrity.

7.3 Eavesdropper

An eavesdropping attack aims to attack the communication links between the different subsystems seen in figure 7.1. The attack is based on listening to data flow between the different subsystems without modifying the contents of any packet.

The consequences are the same as the leak scenario. However the complete database it not available, instead the attacks relies on continuous collection of data which requires some time before malicious acts can be carried out.

One common protection against the eavesdropper is to add encryption on the communication links. The Transport Layer Security (TLS) encryption protocol is very common and could be used in order to encapsulate the request when sent over the communication links [62]. In the scenario that TLS is broken or compromised, an additional layer of encryption can be added on the data in the identification requests. One example of another protection strategy is to use object security. Object security is a technique where the objects are encrypted on the application layer before they are sent on a communication link between parties. One example of a standardized object security technique is JOSE which transforms the data into a JSON object that is then signed and encrypted before it is relayed [63]. By encrypting the data with a public key of a trusted service provider it certifies that only the trusted service provider can decrypt the data with the private key, ensuring that only cipher text is revealed to eavesdroppers.

Chapter 8

Location privacy

As concluded in section 5.1.4, location information can be used to efficiently track and identify users. It has been shown by several studies that some users have privacy concerns when sharing data with services when not fully aware of how the data is used and if it is shared with other parties [64, 65]. Especially location sharing has been shown to be sensitive and it is therefore motivated to conduct further investigations in order to decide if location can be used without breaching the integrity of the users, while still providing sufficient filtering performance.

Some of the discussed protection enhancing techniques in this chapter has been implemented and tested and some techniques has only been evaluated and discussed from a more theoretical viewpoint.

8.1 Attack scenarios

In order to implement mitigation techniques protecting against potential privacy breaches, one must understand that there are different type of breaches that can occur from the users perspective. One possible scenario is that the information stored on the trusted service is leaked and another scenario could be that the trusted service missuses the stored data and shares it without the users consent. A detailed description of possible mitigations of these risks can be seen in the following subsections.

8.1.1 Protection against leaked data

Consider the scenario where the trusted service has been breached and the database has been leaked. Even though an adversary is now in possession of data of all of the enrolled users, it should not be able to make sense of this data. It is very important that it is infeasible to extract the exact location of specific users as this is a serious integrity breach. Two different countermeasures against this threat is discussed below. These countermeasures can be used in conjunction in order to provide greater protection than using only one of them.

One solution would be to encrypt the location (and all other data) in the database. In case of a security breach, the data appears as nonsense to an adversary. This protection relies on the encryption key being securely stored. If the encryption key is also leaked, this protection is void.

The other solution would be to transform location points received in successful identification requests before storing them in the database. Therefore in the case of a breach, the coordinates are not the real coordinates of any user, just pseudo-coordinates. For this solution to work, there needs to exist a location obfuscation function implemented in a secure environment in the trusted service. The security of this solution relies on the inner workings of the obfuscation function to be unknown to everyone other than the trusted service.

8.1.2 Protection against the BTS

Consider the scenario where the user does not fully trust the service provider. A user may not want to blindly rely on that the privacy preserving promises by a service provider are carried out fully. To mitigate this, some noise injection or transformation of the users position can be carried out before the data is sent to the trusted service. Ardagna et al. [66] discusses different transformation techniques where the user can decide to what grade the true coordinate is masked before sending a pseudo-coordinate to the service provider. This would let the users control the privacy level themselves, giving the user more control of the exactness of the information he/she shares. The user would have to be notified that a greater privacy level would decrease the performance of the system.

Another approach could be to shift the position with some delta distance on the application side, before sending the coordinates to the service provider. This approach relies on the transformation being deterministic and individual to each user, otherwise transformation patterns can emerge over time.

8.2 Protection implementation

Taking the previous discussion about protection techniques into consideration, it can be concluded that an optimal system should combine the different techniques in order to maintain the highest level of user privacy. Figure 8.1 illustrates how these techniques can be combined together for the scenario at hand. The client side uses obfuscation techniques with a adjustable privacy level in order to adapt to different user preferences. This obfuscation will inject uncertainties into the location information provided to the BTS in order to protect the true location. The server side will use the previously discussed techniques by first applying a obfuscation transformation before storing an encrypted representation in the BTS database. The rest of this section will be devoted to describing the implementation in more detail.

8.2.1 Client side

As discussed in the beginning of this chapter, there was two possible approaches to client based location protection, namely noise injection or transforming the co-

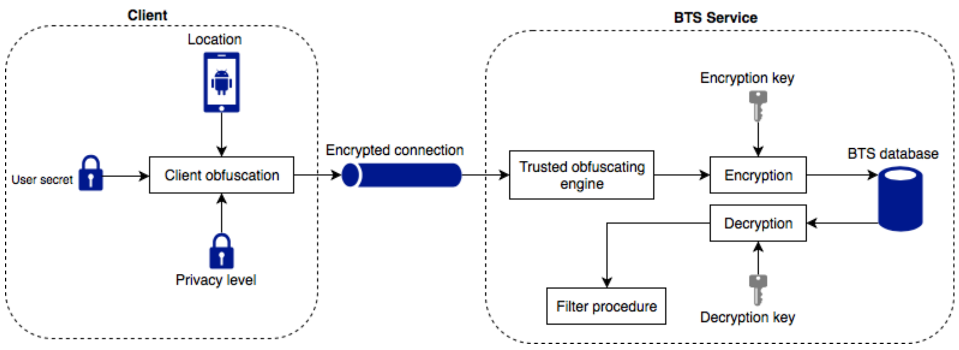


Figure 8.1: Overview of the proposed location protection

ordinate in a deterministic manor. An advantage with the noise based approach is that the transformation does not follow a deterministic pattern, instead the position is moved randomly. The noise injections makes it harder for an adversary to extract the exact location from leaked data. This approach however has the disadvantage of reducing the performance of location based filtering since the individual locations sent to the service provider has randomly added noise, making it harder for the algorithm to infer patterns that can be used for identification.

Another alternative approach is to move the positions in an deterministic manner, preserving the accuracy of the location based filter. The accuracy can be maintained since the transformations can be based on user and location specific information. As the user-location specific transformation is derived from a secret, the privacy of the users will be preserved to some extent even if the algorithm is leaked as the secret is needed in order to fully understand the used transformation.

Both of these approaches provides some level of protection against the service provider as well as other potential adversaries. The first approach a drawback of leaking spatial information to some extent since the clusters will be roughly at their original location. The deterministic approach has the advantage of moving the clusters, therefore making it harder to find out the true positions of the clusters. It should however be noted that the deterministic approach still leaks structural information about the users clusters. The implementations of the two approaches will be described in more detail below.

8.2.2 Noise based obfuscation

The noise based obfuscation technique builds on the idea of reducing the probability of an adversary being able to find out the true location of a user. One solution suggested by Ardagna et al. [66] is to apply different operators in an sequential manner, where each operator performs some kind of transformation on the the original location. Figure 8.2 and 8.3 illustrates two of the operators used in their paper, where the first figure illustrates the expand operator and the second figure illustrates the shift operator. The expand operator simply expands the

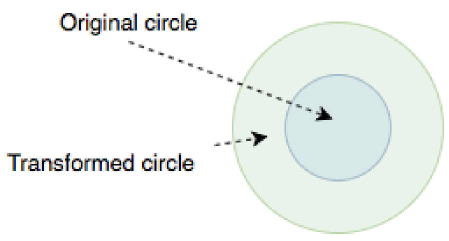


Figure 8.2: Illustration of the expand operator

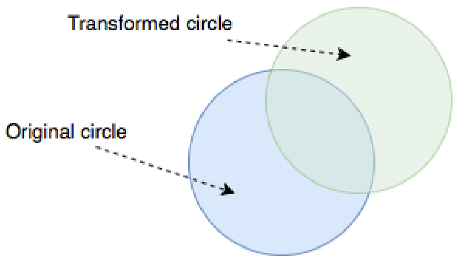


Figure 8.3: Illustration of the shift operator

uncertainty area of the measurement and the shift area simply moves the uncertainty area. The blue circles in the figure corresponds to the original uncertainty area before the transformation is applied and the green circles corresponds to the transformed uncertainty area. According to the authors of the discussed paper, the radius of the original uncertainty area can be provided by the hardware and the parameters of the transformation can be computed using formulas derived in their paper, taking the users privacy preferences into account. The obfuscated location is then generated by selecting a arbitrary location within the transformed uncertainty area.

Implementation

The implementation of the expand and shift based obfuscation algorithm can be seen in algorithm 21. The algorithm begins with computing lambda, which can be seen as a measurement of how much noise that needs to be added in order to provide the desired privacy level. The next step of the algorithm is then to compute the current privacy level in order to decide how aggressive the operators should be in their position transformation. The current privacy level is denoted as r_init in the algorithm. The next step is then to compute the desired privacy level r_final before applying the expand operator. The last step of the algorithm is then to apply the shift operator in order to obtain the final position. The exact details of the formulas can be found in the paper by Ardagna et al. [66].

Algorithm 21: Obfuscation algorithm

```
Data: Location (lat, lng) and obfuscation level R_min
Result: Obfuscated location (obf_lat, obf_lng)
lambda ← compute_lambda(r_meas, r_min);
r_init ← compute_initial_r(r_opt, r_meas);
r_final ← compute_final_r(r_init, l);
radius ← apply_enlarge(r_meas, r_init, r_final);
lat, lng ← apply_shift(radius, r_init, r_final, lat, lng);
lat, lng ← random position on disk of radius $r$ with center (lat, lng);
return lat, lng;
```

Result

Table 8.1 shows measured accuracies for different R_{min} values. When these results are compared to the non-transformed location accuracies seen in table 6.3, it can be concluded that a R_{min} value of more than 100 meters will significantly decrease the accuracy of the filter. It can also be concluded that the accuracy drop is more significant with more location history. It is believed that the reason for this is that with a larger location history more random noise is added to the history, resulting in more spread of the location points, false clusters and overlap with other users location points.

Table 8.1: Accuracies for different R_{min} values

Filter	$R_{min} = 0.05$	$R_{min} = 0.1$	$R_{min} = 0.2$	$R_{min} = 0.5$
Location (30 pts)	0.9749	0.9760	0.9013	0.6296
Location (10 pts)	0.9013	0.9014	0.8826	0.7354

8.2.3 Shifting implementation

This approach shifts the query position before it is sent to the service provider. The shift distance and angle is calculated with a secret derived from both location and user specific information. How this secret can be calculated will be discussed section 8.2.4. Figure 8.4 illustrates the fact that the shift should not be shared for all users, instead different users uses different shift parameters. The pseudo-code can be seen in algorithm 22.

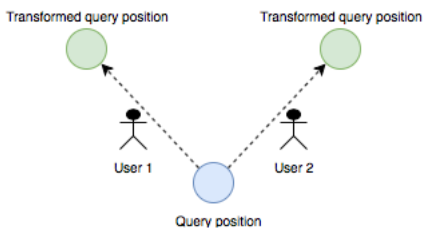


Figure 8.4: Illustration of the shifting approach

Result

Table 8.2 illustrates the accuracy preserving feature of the proposed solution. It can be seen by looking at the table that the deterministic approach provides better filter performance while at the same time providing privacy protection. This improvement comes from the fact that the deterministic approach moves users in random directions, actually making densely populated areas a little less dense

Algorithm 22: Position shift algorithm

```
Data: Query position (lat, lng), secret number (nbr)
Result: Shifted query position
distance $\in [2, 20] \leftarrow$ distance in km generated from nbr;
brng $\in [0, 2\pi] \leftarrow$ angle generated from nbr;
newlat, newlng $\leftarrow$ move lat, lng with distance and brng;
return newlat, newlng;
```

Table 8.2: Filter accuracy for location with and without shifting

Filter	**50.000**	**200.000**
Original Location (30 points)	0.9805	0.9755
Shifted Location (30 points)	0.9922	0.9862
Original location (10 points)	0.9115	0.8964
Shifted location (10 points)	0.9132	0.8998

which leads to better filtering performance. These results were obtained by assuming that the key derivation algorithm generates one deterministic key for each user. It is furthermore assumed that the key derivation is able to generate the correct key in all instances. This assumption is unrealistic and additional testing should be conducted when a concrete algorithm has been implemented.

8.2.4 Key derivation

This section will describe how the secret keys in the deterministic approach can be generated. In order to mitigate that all positions for a specific user gets shifted in the same way, location data will be used in conjunction with user specific information in order to shift the query positions in a non-deterministic way from an adversaries point of view. Figure 8.5 illustrates a suggested approach where two separate keys are used to derive a secret key. The key generation has the following requirements in order to be secure and provide acceptable usability:

- **Deterministic:** The position information should be used together with user specific information in a deterministic way. It should therefore be possible to reconstruct a secret by providing the same user specific information together with a position sufficiently close. The exact meaning of sufficiently close will be discussed later in this section.
- **Secure:** The entropy of the key space should be high in order to make a brute force attack hard.
- **User friendly:** The key generation should be user friendly, meaning that the user interaction should be minimal.

The rest of this section will present suggested approaches of user specified and location based key generation as well as fusion of the generated keys.

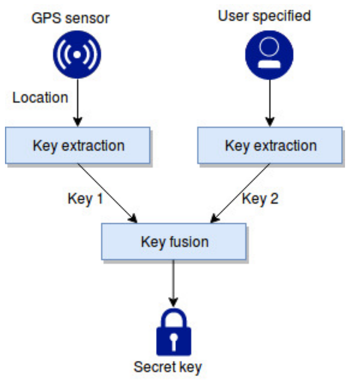


Figure 8.5: Illustrating the key derivation steps

User dependent key derivation

There exist both manually and automatically collected information that could be used to derive a user specific secret. Some examples of manual collection may be to use graphical patterns available on most mobile devices as well as pass phrases or alike. However these approaches are deemed not suitable as they counteract the usability aspect of using biometric information to identify. Instead a focus will be laid on automatic key generation using fingerprint biometrics.

The general idea behind biometric key generation is to generate the same key as long as the biometric sample comes from the same individual. This is a hard task since biometric samples comes with different amount of noise depending on the biometric source [67]. The biometric source can be of different kinds, e.g fingerprint or iris biometrics. From a usability perspective it would be advantageous to use the same biometric source that is used in the identification procedure in order to minimize the user interaction. An optimal key generation algorithm should be able to produce a unique key for each individual as well as producing a deterministic key for each individual. This problem is a reformulation of the identification problem, thus being a challenging task and a solution to this problem would also solve the entire identification problem, but no such algorithm exists to the best of our knowledge. Instead it has been deemed that it is more important to generate a key space of sufficient entropy in order to provide additional privacy protection. As long as the key generation is deterministic, it is okay if some users happens to share the same biometric key. The key generation discussion will be focused on using fingerprints as the biometric source, however it should be noted that there exists similar techniques for other sources.

The fingerprint based key generation typically works by first using a biometric sensor in order to scan a fingerprint, as seen in figure 8.6. The scanned image will then be enhanced using different techniques in order to remove deficiencies, allowing a more consistent representation of the same fingerprint, thus improving the performance of the key generation. The next step is to extract different features from the enhanced image that could be used for a reliable key generation. It is crucial that the selected features are stable in order to produce a stable key generation. The last step is to use the extracted features together with other potential parameters in order to generate a biometric key. The rest of this section will discuss previous work discussing biometric key generation. Most of the presented techniques follows the general structure as described and it will be pointed out if some technique deviates from this structure.

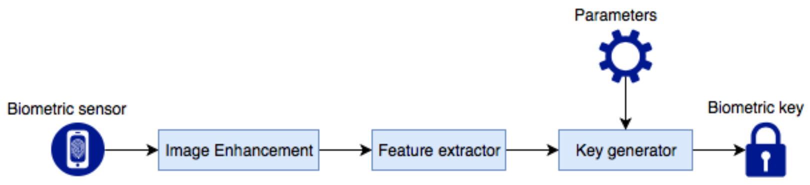


Figure 8.6: Illustration of key generation using fingerprint biometrics

One approach suggested by Panchal et al. [68] builds on applying a series of pre-processing steps before extracting the minutiae set of the scanned fingerprint. The pre-processing stage is an important part of the algorithm, trying to make the key generation more stable. The first step of the pre-processing step is to align the scanned fingerprint image, taking into account that the scanned fingerprints may be rotated. The next step is to select which region of the aligned image to use for minutiae extraction. One requirement on the selected region is that it should have a high probability of occurring in all readings of the same fingerprint. The final step of the pre-processing step is then to extract the minutiae set of the selected region.

The next step is to generate the four feature sets M_a, M_b, M_c, M_d from the extracted minutiae set. Each feature set is generated according to some rule, trying to generate stable features. The features are also constructed in such a way that they are different for different fingerprints. The exact details can be found in the article.

The next step is to generate one Trellis[1] for each of the four feature sets. Each Trellis will then produce one corresponding binary string. These strings are then used to compute the two values g and s, according to formulas in the article. These two values are then used to generate the biometric key BK according to formula 8.1, where H can be any suitable secure hash function.

$$BK = H(\ g \mid\mid s\ ) \tag{8.1}$$

The authors claim an estimated success rate of about 95.12% meaning that the algorithm generated the same key for the fingerprint with a probability of around 95%. This means that this algorithm could be a suitable candidate for a future implementation. It is also possible to include additional information in equation 8.1, in order to make the key generation application dependant. One disadvantage

[1]https://en.wikipedia.org/wiki/Trellis_(graph)

with including application dependant information is that the location history will be somewhat useless for cross-application identification.

Fuzzy extractors are another approach to the problem of using noisy data to generate stable keys [69]. Fuzzy extractors requires an additional enrollment procedure to take place, where a biometric template is used as input to a generator function which then outputs a secret key and an auxiliary helper string. The main disadvantage with fuzzy extractors is that the auxiliary helper string is required to be present during an identification request, which implies that the helper string must be stored and retrieved from somewhere. As this would add additional user interaction in order to retrieve the correct auxiliary string, it is not considered further.

Location dependent key generation

The location will be used in order to provide an additional layer of security, generating different keys depending on the location of the identification request. Figure 8.7 illustrated the proposed key generation where positions sufficiently far apart generates different keys. This problem can be formalized to a key generation problem that has been studied by different researches, suggesting different approaches. One approach suggested by Liao and Chao enables positions sufficiently close to generate the same key [70]. The major advantage with their method is that it enables fine tuning of the proposed algorithm in order to control what sufficiently close is. This is a hard problem since the sensitivity affects the entropy of the generated key space, thus also affecting the provided security. The sensitivity also affects the filtering performance since a very sensitive key generation could lead to close points being moved in such a way that their spatial relation is changed in a non-deterministic way. One solution to this problem could be to consider using some other kind of key generation by using hand drawn boundaries in order to avoid placing boundaries in populated areas. This approach could potentially mitigate a potential accuracy drop.

This algorithm has a security problem since the area dependent keys can be considered to be known by an adversary. This fact makes it easy to extract the used location key for an individual if the adversary knows where the person is likely to appear. The security of this approach is therefore strongly correlated with the selected region size as well anonymity level of other recorded metadata. Thus it is evident that the location based key derivation should be used together with some other key derivation algorithm in order to provide sufficient protection.

Combined key derivation strategy

The last step is to use a key derivation function that combines the two keys in order to provide user and location dependant keys, as seen in algorithm 23. Algorithm 23 shows the suggested approach where HKDF [71] is used as the key derivation function, using the the two keys as input. However in theory, any secure one way function would suffice.

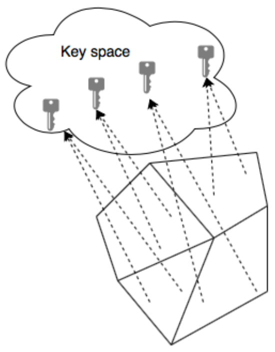


Figure 8.7: Illustration of key generation using location

Algorithm 23: Combined key derivation strategy

```
Data: Template (template) and location (loc) associated with the
      identification request
Result: User and location dependent key (key)
user_key ← generate key from template;
loc_key ← generate key from loc;
return HKDF(user_key, loc_key);
```

8.2.5 Server side

The service provider could implement an analogous protection scheme as the deterministic approach discussed in section 8.2.3. The primary problem to be solved by a concrete implementation would be how to know which transformation that should be applied to a position associated with an identification request. One solution would be to apply the same transformation to all positions, but that would reduce the security of the algorithm. It is also possible to add further noise to the location information on the server side, but that would reduce the filtering performance and should therefore be avoided.

The service provider should also encrypt the location information with an encryption scheme in order to protect against a leaked database. The encryption keys should be very well protected in order for the encryption to be meaningful.

8.3 Security evaluation

The protection provided by the two approaches is limited by the fact that both approaches are client based. This means that all clients will have access to the inner workings of the protection algorithm. This makes it possible for an adversary to reverse engineer the implementation by examining in the application code in order to gain knowledge about the protection strategy. It should therefore be assumed that the implementation is public knowledge and an ideal implementation should still be able to provide some level of privacy if the code is leaked. It is also assumed that an adversary has access to all location information stored within the BTS database allowing patterns to be extracted. A third assumption is that the adversary is aware of the accuracy provided by the location sensing technology.

8.3.1 Deterministic approach

The privacy provided by the deterministic approach is limited by the fact that the coordinates are only shifted in a deterministic direction for each user. It is possible for the service provider (or any other third party), with a history of shifted points, to extract movement patterns if some more information about a user is known. E.g. if a name is paired with a location history, a possible attack scenario could be:

1. Extract home address of all persons with a specific users full name in a large bounding area centered around a coordinate stored for that user.

2. Calculate the distance vector from the home address and the coordinate used in step 1.

3. Try every different distance vectors to shift all coordinates in the specific users location history.

4. If some of the transformed coordinates seem to be placed at public places, such as schools/work places, training facilities or bars, it is possible that the reverse shift vector for that said user has been found. This vector can then be used in the future to deobfuscate all locations sent to the service provider for that specific user.

The complexity of the proposed attack heavily depends on the distance interval that each position could be shifted as longer distances would require a larger bounding area to be used. This means that the positions should be moved as much as possible in order to increase the computational complexity required to carry out the discussed attack. The entropy of the secret number used in algorithm 22 also affects the complexity of an attack, since a higher entropy would make it harder to brute force the number.

A possible countermeasure against this attack scenario would be to separate the location information from any other information about users, e.g. name and age. By separating the information into different databases, an adversary has to breach both databases in order to perform the proposed attack. However this separation only protects against adversaries and not against malicious service providers. It should however be noted that the proposed attack can be performed without using name information. The only difference is that the complexity would be higher as all persons within the bounding area needs to be considered when calculating all the different distance vectors in step 2.

8.3.2 Noise injection

Following the assumptions made in the introduction to this chapter, the obfuscation operators used to transform the locations are assumed to be known to an adversary. According to a study by Ardagna et al. [72], it has been shown that the combination of shift and expand operators provide the highest amount of protection against adversaries. The adversaries had a maximum of approximately 42% success rate when deobfuscating the transformed area by 10% and a success rate of a merely 12% with an deobfuscation level of 70%. However those results are based on an adversary model where the adversary is only in possession of one single query location point. This security evaluation will expand the adversary model specified in the study, assuming that the obfuscated area is known as the adversary is in possession of location history, allowing a more sophisticated attack.

The privacy provided by the noise based approach is strongly correlated with the selected R_{min} value. A large R_{min} value will provide stronger privacy than a small value but at the same time affect the filter accuracy negatively. Looking at table 8.1, it can be concluded that the R_{min} value needs to be $100m$ or less in order to maintain filtering accuracy. However an obfuscated area with such a limited size can only offer very limited privacy protection. Figure 8.8 shows a fraction of a

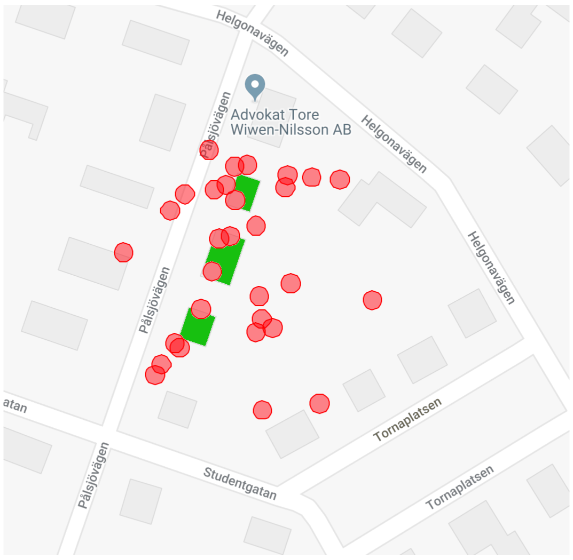


Figure 8.8: Location history for a specific location of a user with $R_{min} = 100$

specific users location history. It can be seen from the figure that with a large set of location points, it is easy to see patterns and extract the probable location of the user. Looking at the figure it seams reasonable to assume that the user lives in some of the green colored houses. It can therefore be concluded that using noise injection is not enough to protect against an adversary, instead it must be used in conjunction with some other technique in order to provide good enough privacy protection.

Chapter 9

Discussion

The discussion will be divided into four smaller parts, each discussing some significant component of the conducted work. The discussion will follow the chronological order of the thesis, starting with a discussion about the developed simulator, followed by a discussion about filtering results of both uni- and multimodal filtering implementations, followed by a discussion concerning the security evaluation before finishing the chapter with a discussion about the current state of the implemented system and its deployability status.

9.1 Simulator discussion

The first major decision in the thesis work was how to evaluate the performance of the final implementation in order to get an objective measurement. Several possible approaches was suggested before finally deciding on developing a simulator in order to allow objective measurements. The reason for choosing the simulator based approach was that it was the only viable option available that enabled investigations into different metadata types to be conducted in a flexible and extensible manor.

The largest concern when choosing the simulator based approach was that the simulator would be unable to correctly simulate real world dynamics of a deployed system. This concern was addressed by using statistics gathered from official sources to the highest degree possible. The flexibility of the simulator also allows different parameters to be used in order to analyze the performance impact of different usage scenarios.

The implemented simulator assumes that there exists an independence between the different metadata types that are simulated. This is not the case in the real world since there exists certain dependencies between metadata types. One example of a dependant pair is name and age. The name distribution differs between ages but this has not been taken into consideration in the simulator as a viable source of statistics for this was not found.

The simulated types was selected by considering several important aspects, such as availability, uniqueness and usability. The selected metadata types represents both manually and automatically collected metadata types that are deemed to be available for users using mobile devices. It should however be noted that the simulator is constructed in such a way that new metadata types could easily be integrated and tested. Any new metadata type should use distributions based on real life statistics in order to provide correct results when used in the simulation. The following sections will discuss the simulation of the selected metadata types in more detail.

9.1.1 Device ID discussion

The simulation of device ID is not based on gathered statistics since no such statistics was found during the implementation. Instead it was built with a worse case scenario in mind. It was decided that 20% should be more than enough to cover the cases where users try to identify with brand new device IDs. It is believed that this probability will be much smaller in a deployed system, making device ID based filtering even more efficient.

9.1.2 Age discussion

The age simulation is built on statistics gathered from SCB making it realistic. It should however be noted that this statistics are based on the Swedish population and it could be the case that the age distribution of other countries could affect the performance measurements slightly.

9.1.3 Name discussion

The name distribution is based on statistics gathered from SCB in order to approximate the actual name distribution seen in the Swedish population. The statistics consists of the 500 most common first and surnames, approximating the actual distribution to great length. One should however keep in mind that the name distribution differs between countries and further investigations could be conducted into how the name distribution affects the filtering performance.

The probabilities of different types of typing errors was extracted from a study by Baba and Suzuki [41]. This study did however not provide probability of typing errors being performed and this probability was set with a worst case in mind. It is believed that most users are capable of writing their own name with just a few typing errors at most, but the simulator generates typing errors with a probability of 33% per character. The simulator does not take spelling errors into consideration as the Jaro Winkler string distance measurement used in comparisons gives equal penalties for typing errors compared to spelling errors.

9.1.4 Location discussion

As discussed in section 5.1.4, the location simulation is built on demographic statistics gathered from SCB in order to properly simulate the population distribution throughout Sweden. Looking back at figure 5.8 together with gathered statistics, it can be argued that the simulator correctly simulates the population distribution on a large scale. The more fine grained location simulation is built on studies showing that a user trends to have a number of significant locations. The simulator uses this information in order to generate query location points around the users significant places. This is a limitation as it fails to simulate login attempts of users moving between their significant places. This is a probable scenario in a deployed system and a possible improvement could be to integrate this movement and run tests on the scenario.

9.2 Filtering discussion

It was decided at an early stage to include both unimodal as well as multimodal filters in order to first obtain a baseline that could be used as a starting point. The multimodal filters could then be evaluated against the baseline in order to investigate the potential accuracy improvement.

9.2.1 Unimodal filtering

The performance of the unimodal filters differs relatively much compared to the multimodal filters, as can be seen in table 6.3. The best accuracy for 50.000 enrolled users was 99.20% and it was obtained by the name based filter. At 200.000 enrolled users the best accuracy was 97.55% and it was obtained by the location based filtering. The reason for this is that a name is unique enough with a smaller set of enrolled users but as the set grows, the accuracy drops as more and more users share the same name. The performance of location based filtering is not affected much by the increase of users as the uniqueness of users clusters are still enough to not significantly affect the performance. However it is believed that the accuracy will decline as the number of users grow much larger.

The advantage with unimodal filters is that they require less information than the multimodal filters, but by observing the obtained accuracies it seems like a good idea to use a multimodal filter in order to obtain a lower FRR.

9.2.2 Multimodal filtering

By observing the performance of the multimodal filters it can be concluded that multimodal filtering is superior to unimodal filtering. This performance gain comes with the cost of requiring additional metadata to be collected during an identification.

From the multimodal filtering approaches, only the combination of location and age fails to satisfy performance requirements. It can be seen from the accuracy results that there exists too many people sharing the same age in a densely populated area, making the age and location filter combination fall short.

The best combination was obtained by fusing location, age and name information. The obtained accuracy was 99.92%, meaning that only 0.08% gets wrongly rejected when trying to identify themselves using the provided metadata. This is quite good and should be strongly considered because of the superior performance. Table 9.1 shows examples of metadata where a user gets wrongly rejected. As can be seen in the table the supplied names contains so many spelling errors that it could be considered to be a user error.

Table 9.1: Example of metadata from wrongly rejected requests

	Stored metadata	**Query metadata**
Name:	Felix Olsson	Fsoi Lpdon
Age:	47	47
Name:	Ulla Jansson	Uk Jansslb
Age:	71	70
Name:	Stephan Axelsson	Ssphah Qls0
Age:	68	68
Name:	Jari Fredriksson	Har8 Fdeddikewon
Age:	75	76
Name:	Joel Andersson	Jp Ajdetsson
Age:	34	34
Name:	Ebba Björk	Rgbz Gjörik
Age:	73	73
Name:	Jonas Englund	Nw Ehgluhd
Age:	27	26

9.3 Security discussion

The discussed attacks were selected by considering different attack scenarios believed to be relevant and cover all subsystems of the identification system. The conducted security evaluation was focused on evaluating the security threats exploiting the parts relevant to metadata information, filtering and transmission. There might exist additional attacks targeting other parts of the deployed systems. However these were not considered since they require more detailed knowledge of the deployed system. The exact architecture of the deployed system is not considered in this report and it is therefore unfeasible to perform such an investigation. Relevant security threats found during the security evaluation have been presented together with possible mitigation strategies. Some mitigation strategies builds on several suggested approaches that could be used in combination in order to achieve a strong protection.

The conducted investigation into location privacy serves as a template study, illustrating the research approach that should be used for any other privacy investigation. The suggested approach is to estimate the privacy concerns by reading or performing surveys in order to decide if the privacy concerns are high enough to motivate further investigation. The next step is to develop protection strategies by using previous work as a starting point. The final step is to evaluate how the developed protection affects the performance of the system in order to evaluate the applicability of the implemented protection.

9.3.1 Location security

The implemented privacy protection techniques for location has been deemed to protect the users privacy to some extent. However there are still concerns for when an adversary is in possession of location history. Then there exists a small probability that an adversary is able to perform a brute force attack, as described in section 8.3.1. The discussed countermeasures protects against such an attack to a certain degree, however if all data is leaked, the proposed attack is still possible to perform.

9.3.2 Key derivation

The shift based privacy preserving technique could potentially have a negative impact on the filtering performance as the technique builds on transforming the query positions which is used during filtering. This observation leads to the conclusion that care must be taken to make sure that the shifting approach preserves the relation between different positions. The shifting approach works by using a derived key in order to move positions in a deterministic way and the problem of construction a reliable shifting algorithm can therefore be formulated as two semi-independent problems. The first problem is to construct an algorithm able to transform all positions in the same deterministic way given the same key. The second problem is to have a stable and reliable key generation, able to produce equal keys between readings for individual users. One of the suggested approaches to the location based key derivation was to divide the earth into different cells where each cell was assigned a key. One drawback with this approach is that two positions only a few meters apart may generate different keys, if the positions are separated by a boundary. This fact leads to the conclusion that the boundaries should be constructed in such a way that this risk is minimized. The user dependant key derivation has not been implemented but an approach based biometric based key derivation was made. Because of the fuzzy nature of biometric templates, even if they originate from the same biometric source, there exists a need to further investigate if the suggested approach is viable for a large number of enrolled users.

The key derivation performance is limited by the performance of the location as well as biometric based key derivation since the key fusion is influenced by both. This means that their individual performance needs to be acceptable in order for the fused key derivation to have an acceptable performance. The performance of the biometric based key derivation is estimated to be around 95.12% and it can therefore be concluded that the performance of the fused key generation is at most 95.12%. It can also be concluded that a concrete implementation will lower the performance somewhat, motivating additional experimentation in order to optimize performance.

9.4 Deployability

An ideal deployable system should fulfill the following properties; stable, secure and user friendly. The filter implementation is a back end system, implying that there is close to no user interaction. The only thing in the filter implementation that affects the user friendliness is the amount of metadata required in the filtering process. As it has been shown in the results, there exists a need of two or three metadata types if the device ID based filtering fails in order to achieve a reliable identification. It can be argued that this is still user friendly since this is a rare event only occurring if the user uses new devices when identifying. It can also happen if more than 50 persons are using a specific device, but it is believed that this is also a rare event and a potential solution could be to use a multimodal metadata filter if this is the case. The rest of this discussion will focus on the stability and security of the implemented filtering solution.

9.4.1 Stability

The stability of the implementation was evaluated by extensive testing under different usage scenarios. The obtained results shows that most multimodal filters are stable, performs well and could be used in a deployed system, as concluded in section 9.2.2. The stability of the implementation is deemed stable for a deployed system, however there exists additional work that should be carried out before a potential deployment. This additional work will be described in chapter 10.

9.4.2 Security

The security from a FAR and FRR standpoint are within an acceptable range and it is therefore deemed that the identification security is enough. The privacy protection in case of a leak is however somewhat lacking for other metadata types other than location. The only protection against a privacy breach is case of a database leak is that the BTS database is encrypted. Further protection strategies should therefore be investigated before a potential deployment could be carried out.

9.4.3 Integration

In order to integrate the developed filtering solution, a database containing recorded metadata as well as identification requests containing query metadata and templates needs to be implemented. The identification request will be sent to the filter from the component responsible for receiving identification requests, as seen in figure 9.1. The filter implementation will then relay a set of possible candidates to the biometric matcher which is assumed to be an existing component.

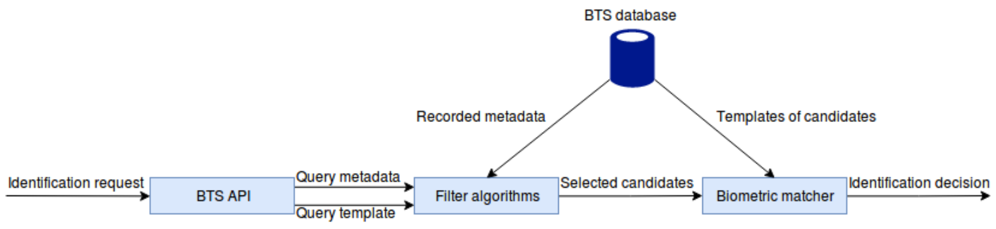


Figure 9.1: Illustrating integration of a filter component

Chapter 10

Future work

The thesis have presented a functional simulator of metadata and several filter implementations. The conducted work has raised many new relevant questions as well as answering the questions presented in section 1.3. The following sections presents future work believed to be relevant for a future deployment.

10.1 Improving the simulator

The current implementation of the simulator does not support dependencies between metadata types. It is believed to be very interesting to simulate these dependencies and how they affect the filtering performance. To enable such a simulation, real life statistics would have to be extracted that show signs of these dependencies. A possible starting point could be to gather data from a register of all the citizens in a country and extract distributions from said data.

The location simulation could also be expanded, allowing for movement between clusters. It is possible that the bandwidth of the kernel used in the location filter would have to be increased in order to perform better for inter-cluster movement.

10.2 Simulation of more metadata types

This thesis have focused on a few selected metadata types. As mentioned before, there is no limitation to using only these in a filtering procedure, however statistics about new metadata types would have to be extracted in order to simulate real life dynamics. From a usability perspective it would be favourable to focus on automatically collected metadata, e.g. IP address, location trace, gait trace and phone number.

10.3 Automatic detection of anomalies

Another potentially interesting research direction is automatic detection of anomalies. Previous work within this area has been conducted by several researchers where they are trying to classify a http request as either malicious or legitimate

[60, 61]. This approach could be useful as a protection mechanism in order to uncover potential attacks against the identification system. The research would have to consider both how to perform the classification in terms of features and accuracy as well as how to integrate the classification into a deployed system. One concern is that the FRR can increase if the classifier is too aggressive in its predictions. It could therefore be an idea to include a human operator in the process.

10.4 Dynamic filters

A possible improvement to current filter implementation could be to implement dynamic filters. E.g. when it is noted that the query comes from a sparsely populated area, a less sophisticated filter combination could be used compared to when a query location comes from a densely populated area. This is believed to increase the overall performance as the execution time would be slightly lower. One starting point could be to investigate the performance of the location filter based on the population density on the query position.

10.5 Distributed computing

A deployed system is believed to be used by many people which requires more computing power than could be provided by a single computer. The system will also accumulate data over time and it is therefore motivated to consider using distributed computing in order to provide necessary robustness and performance. This will require a distributed database instead of a file based database, as currently used by the implementation.

Distributed computing can provide speedup for the unimodal filters by partitioning the candidate set, enabling Map-Reduce to be applied. Each partition can be assigned to a specific server responsible for producing scores for the assigned partition. The partitioned scores can then be combined in order to obtain the final candidate ranking.

The multimodal filters can be re-implemented in an similar manor by distributing the tasks of performing filtering on different metadata types to different servers, as seen in figure 10.1. Each server will produce a ranking of the candidates which is then sent to a server responsible for producing a final ranking of the candidates.

10.6 Improved key generation

The key generation consists of two sub generators where each generator is responsible for producing a key. Both generators can be improved in order to produce a higher level of privacy meanwhile maintaining a high filtering performance.

Following the discussion in section 9.3.2, there were some certain points that

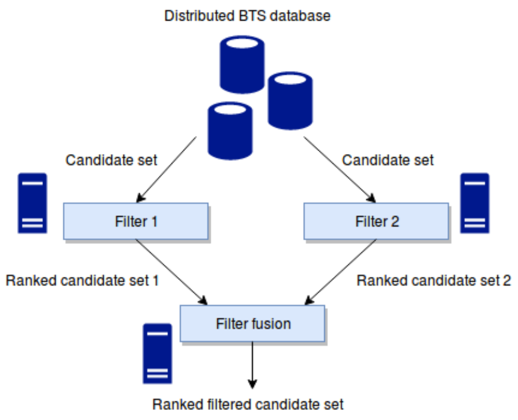


Figure 10.1: Illustrating distributed computing for multimodal filtering

needed to be addressed before an implementation of fingerprint key derivation for this thesis' purpose can be done.

The location based key generation can be improved by using clever boundaries in order to minimize the possibility of someone being on a boundary. This approach is expected to perform more optimally than the proposed solution. Further work would be needed in order to determine the feasibility of such an approach from both a privacy as well as a performance point of view. Another approach could be to detect when a position is close to a boundary and use the key of the neighbouring cell as well as the key of the enclosing cell. This would result in two different points that could be sent to the BTS which would be responsible for including both positions into the filtering process. It may be possible to select the most probable point but further work should be conducted in order to estimate the performance impact of such a method.

Chapter 11

Conclusions

The main goal with this thesis was to investigate if biometric identification could be used with a large number of enrolled users. Previous work has mainly dealt with biometric authentication, which is a more simple procedure compared to identification. The chosen approach for this thesis was to investigate the possibility of performing a reliable identification by using metadata in a filtering process before biometric matching is carried out. It was believed at a fairly early stage that filtering would increase the performance of the system and focus was therefore placed on trying to measure and improve the filtering performance.

Results show that filtering on unimodal metadata is not enough to provide sufficient performance for a large number of enrolled users. However by combining different types of metadata, significant performance improvements was seen. It has been concluded that two different types of metadata would be sufficient to provide acceptable performance for most cases but in very densely populated areas, three metadata types could be required. The most important conclusion is therefore that it is possible to perform metadata with sufficient filtering performance by integrating several different metadata types into the filtering procedure.

The security of the proposed system was also discussed and it was concluded that the security of the system could be improved by implementing different mitigation strategies. These strategies should be implemented before the system is deployed in order to protect against breaches and other security threats. The thesis also discussed potential strategies that should be used in order to protect against privacy breaches, originating from both the service provider as well as an outsider. These strategies was not implemented fully but could work as an starting point for further investigations.

The developed simulator together with the presented theory and results is a solid base for further investigations into additional metadata types. All further investigations should consider the proposed strategy where data is collected from trustworthy sources and used in the simulation in order to predict the performance in a deployed system more accurately.

Looking back at the questions formulated in the introduction to this thesis, it can be concluded that all questions have been answered.

References

[1] J. E. Weber, D. Guster, P. Safonov, and M. B. Schmidt, "Weak password security: An empirical study," *Information Security Journal: A Global Perspective*, vol. 17, no. 1, pp. 45–54, March 2008.

[2] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang, "The tangled web of password reuse," in *21st Annual Network and Distributed System Security Symposium*, February 2014.

[3] S. Z. S. Idrus, E. Cherrier, C. Rosenberger, and J.-J. Schwartzmann, "A review on authentication methods," *Australian Journal of Basic and Applied Sciences*, vol. 7, pp. 95–107, June 2013.

[4] A. Ross and A. K. Jain, "Multimodal biometrics: An overview," in *12th European Signal Processing Conference*, pp. 1221–1224, September 2004.

[5] M. O. Derawi, D. Gafurov, R. Larsen, C. Busch, and P. Bours, "Fusion of gait and fingerprint for user authentication on mobile devices," in *2010 2nd International Workshop on Security and Communication Networks (IWSCN)*, May 2010.

[6] F. Wang and J. Han, "Multimodal biometric authentication based on score level fusion using support vector machine," *Opto-Electronics Review*, vol. 17, no. 1, pp. 59–64, March 2009.

[7] K. Vishi and V. Mavroeidis, "An evaluation of score level fusion approaches for fingerprint and finger-vein biometrics," in *10th Norwegian Information Security Conference (NISK 2017)*, pp. 124–134, November 2017.

[8] A. K. Jain, A. Ross, and S. Karthikeyan, "An introduction to biometric recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, pp. 4–20, January 2004.

[9] A. K. Jain, J. Feng, and N. K. Karthikeyan, "Fingerprint matching," *Computer*, vol. 43, no. 2, pp. 36–44, February 2010.

[10] C. Böhm, I. Färber, S. Fries, U. Korte, J. Merkle, A. Oswald, T. Seidl, B. Wackersreuther, and P. Wackersreuther, "Efficient database techniques for identification with fuzzy vault templates," in *Proceedings of the Special Interest Group on Biometrics and Electronic Signatures*, pp. 115–126, January 2011.

[11] C. Gehrmann, “Fpc cloud technology project ii, metadata filtering techniques and strategies,” Fingerprint Cards AB, Tech. Rep., November 2017.

[12] A. K. Jain, F. Patrick, and A. A. Ross, *Handbook of Biometrics.* Springer, 2008.

[13] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2006.

[14] “Facial recognition systems metadata usage,” Facial Recognition Scientific Working Group, Tech. Rep., ver. 1.0, 2014.

[15] S. J. Russell and P. Norvig, *Artificial Intelligence - A Modern Approach - Third Edition.* Prentice hall, 2010.

[16] T. D. Pigott, “A review of methods for missing data,” *Educational Research and Evaluation*, vol. 7, no. 4, pp. 353–383, August 2001.

[17] N. Jaques, S. Taylor, A. Sano, and R. Picard, “Multimodal autoencoder: A deep learning approach to filling in missing sensor data and enabling better mood prediction,” in *Seventh International Conference on Affective Computing and Intelligent Interaction*, pp. 202–208, October 2017.

[18] G. V. Bard, “Spelling-error tolerant, order-independent pass-phrases via the damerau-levenshtein string-edit distance metric,” in *Proceedings of the Fifth Australasian Symposium on ACSW Frontiers*, vol. 68, pp. 117–124, January 2007.

[19] W. E. Winkler, “String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage,” *Proceedings of the Section on Survey Research*, pp. 354–359, January 1990.

[20] G. V. Brummelen, *Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry.* Princeton University Press, 2013.

[21] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, “Locality-sensitive hashing scheme based on p-stable distributions,” in *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, pp. 253–262, June 2004.

[22] P. Spanger, K. Takahiro, and T. Takenobu, “Titch: Attribute selection based on discrimination power and frequency,” in *Proceedings of UCNLG+MT: Language Generation and Machine Translation*, pp. 98–100, January 2008.

[23] M. Czajkowski, M. Grzes, and M. Kretowski, “Multi-test decision tree and its application to microarray data classification,” vol. 61, pp. 35–44, May 2014.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, October 2011.

[25] F. Chollet *et al.*, “Keras,” https://keras.io, 2015.

[26] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303–314, December 1989.

[27] J. A. Hartigan, *Clustering Algorithms.* John Wiley & Sons, Inc., 1975.

[28] J. Reddy and B. Kavitha, “Clustering the mixed numerical and categorical dataset using similarity weight and filter method,” *International Journal of Database Theory and Application*, vol. 5, no. 1, pp. 121–134, April 2012.

[29] A. Sayers, Y. Ben-Shlomo, A. W. Blom, and F. Steele, “Probabilistic record linkage,” *International Journal of Epidemiology*, vol. 45, no. 3, pp. 954–964, June 2016.

[30] J. de Bruin, “Probabilistic record linkage with the fellegi and sunter framework,” Master’s thesis, Delft University of Technology, October 2015.

[31] I. P. Fellegi and A. B. Sunter, “A theory for record linkage,” *Journal of the American Statistical Association*, vol. 64, pp. 1183–1210, December 1969.

[32] J. L. Bentley, “Multidimensional binary search trees used for associative searching,” *Commun. ACM*, vol. 18, no. 9, pp. 509–517, September 1975.

[33] E. W. Bai, “Big data: The curse of dimensionality in modeling,” in *Proceedings of the 33rd Chinese Control Conference*, pp. 6–13, July 2014.

[34] I. G. Damousis and S. Argyropoulos, “Four machine learning algorithms for biometrics fusion: A comparative study,” *Applied Computational Intelligence and Soft Computing*, vol. 2012, pp. 1–7, January 2012.

[35] P. Billingsley, *Convergence of probability measures*, 2nd ed., ser. Wiley Series in Probability and Statistics: Probability and Statistics. New York: John Wiley & Sons Inc., 1999.

[36] R. Kaur and A. Kaur, “Digital signature,” in *2012 International Conference on Computing Sciences*, pp. 295–301, September 2012.

[37] A. Martin and I. Martinovic, “Security and privacy impacts of a unique personal identifier,” *Working Paper Series*, no. 4, April 2016.

[38] SCB. (2017) Medelfolkmängd (efter födelseår) efter region, ålder och kön. År 2017. http://www.statistikdatabasen.scb.se/pxweb/sv/ssd/START__BE__BE0101__BE0101D/MedelfolkFodelsear/?rxid059feb7c-7c33-4103-88e0-a91d9db9d207. Accessed April 2018.

[39] ——. (2017) Förnamn med minst 10 bärare bland folkbokförda 31 december 2017. http://www.statistikdatabasen.scb.se/pxweb/sv/ssd/START__BE__BE0001__BE0001G/BE0001FNamn10/?rxidf45f90b6-7345-4877-ba25-9b43e6c6e299. Accessed April 2018.

[40] ——. (2017) Efternamn med minst 10 bärare bland folkbokförda 31 december 2017. http://www.statistikdatabasen.scb.se/pxweb/sv/ssd/START__BE__BE0001__BE0001G/BE0001ENamn10/?rxidf45f90b6-7345-4877-ba25-9b43e6c6e299. Accessed April 2018.

[41] Y. Baba and H. Suzuki, “How are spelling errors generated and corrected?: A study of corrected and uncorrected spelling errors using keystroke logs,” in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers*, vol. 2, pp. 373–377, January 2012.

[42] V. Martínez and F.-L. Álvarez, "State of the art in similarity preserving hashing functions," in *The 2014 International Conference on Security and Management (SAM'14), Worldcomp 2014*, pp. 139–145, July 2014.

[43] S. Myerscough, B. Lowe, and F. Alpert, "Willingness to provide personal information online: The role of perceived privacy risk, privacy statements and brand strength," *Journal of Website Promotion*, vol. 2, no. 1-2, pp. 115–140, April 2006.

[44] A. Bhattacharya and S. K. Das, "Lezi-update: An information-theoretic framework for personal mobility tracking in pcs networks," *Wireless Networks*, vol. 8, no. 2/3, pp. 121–135, March 2002.

[45] B. Shaw, J. Shea, S. Sinha, and A. Hogue, "Learning to rank for spatiotemporal search," in *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, ser. WSDM '13, pp. 717–726, February 2013.

[46] F. M. Naini, J. Unnikrishnan, P. Thiran, and M. Vetterli, "Where you are is who you are: User identification by matching statistics," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 2, pp. 358–372, February 2016.

[47] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel, "Unique in the crowd: The privacy bounds of human mobility," *Scientific Reports*, vol. 3, no. 1, March 2013.

[48] L. Rossi, J. Walker, and M. Musolesi, "Spatio-temporal techniques for user identification by means of GPS mobility data," *EPJ Data Science*, vol. 4, no. 1, August 2015.

[49] SCB. (2017) Befolkningstäthet (invånare per kvadratkilometer) per tätort 2017. http://www.statistikdatabasen.scb.se/pxweb/sv/ssd/START__BE__BE0101__BE0101C/BeftathetkvkmT/?rxid78383e5a-efb2-407c-817f-37939848a29d. Accessed April 2018.

[50] M. Kihl, P. Ödling, C. Lagerstedt, and A. Aurelius, "Traffic analysis and characterization of internet user behavior," in *International Congress on Ultra Modern Telecommunications and Control Systems*, pp. 224–231, October 2010.

[51] A. Noulas, S. Scellato, C. Mascolo, and M. Pontil, "An empirical study of geographic user activity patterns in foursquare," pp. 70–573, January 2011.

[52] S. Isaacman, R. Becker, R. Cáceres, S. Kobourov, M. Martonosi, J. Rowland, and A. Varshavsky, "Identifying important places in people's lives from cellular network data," in *Pervasive Computing*, ser. Pervasive'11, K. Lyons, J. Hightower, and E. M. Huang, Eds., pp. 133–151, June 2011.

[53] Z. Yang, J. Yu, and M. Kitsuregawa, "Fast algorithms for top-k approximate string matching," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, vol. 3, pp. 1467–1473, January 2010.

[54] H. Zhang, Z. Chen, Z. Liu, Y. Zhu, and C. Wu, "Location prediction based on transition probability matrices constructing from sequential rules for spatial-temporal k-anonymity dataset," *PLOS ONE*, vol. 11, no. 8, pp. 1–22, August 2016.

[55] M. Umair, W. S. Kim, B. C. Choi, and S. Y. Jung, "Discovering personal places from location traces," in *16th International Conference on Advanced Communication Technology*, pp. 709–713, February 2014.

[56] D. Zhao, J. Ma, X. Wang, and X. Tian, "Personalized location anonymity - a kernel density estimation approach," in *Web-Age Information Management*, pp. 52–64, June 2016.

[57] A. Guttman, "R-trees: A dynamic index structure for spatial searching," *SIGMOD Rec.*, vol. 14, no. 2, pp. 47–57, June 1984.

[58] R. Baca, *R-tree example*, 2010. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/thumb/6/6f/R-tree.svg/560px-R-tree.svg.png

[59] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford, "Captcha: Using hard ai problems for security." in *EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 2656, pp. 294–311, May 2003.

[60] L. Machlica, K. Bartos, and M. Sofka, "Learning detectors of malicious web requests for intrusion detection in network traffic," *CoRR*, vol. abs/1702.02530, 2017.

[61] H. Lampesberger, P. Winter, M. Zeilinger, and E. Hermann, "An on-line learning statistical model to detect malicious web requests," in *Security and Privacy in Communication Networks*, M. Rajarajan, F. Piper, H. Wang, and G. Kesidis, Eds., pp. 19–38, September 2012.

[62] T. Dierks and E. Rescorla, "The transport layer security (tls) protocol version 1.2," RFC 5246, August 2008.

[63] M. Miller, "Examples of protecting content using json object signing and encryption (jose)," Internet Requests for Comments, RFC Editor, RFC 7520, May 2015.

[64] I. Shklovski, S. Mainwaring, H. Hrund Skúladóttir, and H. Borgthorsson, "Leakiness and creepiness in app space: Perceptions of privacy and mobile app use mobile," in *Conference on Human Factors in Computing Systems - Proceedings*, pp. 2347–2356, April 2014.

[65] A. P. Felt, S. Egelman, and D. Wagner, "I've got 99 problems, but vibration ain't one: A survey of smartphone users' concerns," in *Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, pp. 33–44, October 2012.

[66] C. A. Ardagna, M. Cremonini, E. Damiani, S. D. C. di Vimercati, and P. Samarati, "Location privacy protection through obfuscation-based techniques," in *Proceedings of the 21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, pp. 47–60, July 2007.

[67] C. Herder, B. Fuller, M. van Dijk, and S. Devadas, "Public key cryptosystems with noisy secret keys," *IACR Cryptology ePrint Archive*, vol. 2017, pp. 210–246, February 2017.

[68] G. Panchal, D. Samanta, and S. Barman, "Biometric-based cryptography for digital content protection without any key storage," *Multimedia Tools and Applications*, pp. 1–22, March 2017.

[69] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM J. Comput.*, vol. 38, no. 1, pp. 97–139, March 2008.

[70] H.-C. Liao and Y. H. Chao, "A new data encryption algorithm based on the location of mobile users," *Information Technology Journal*, vol. 7, no. 1, pp. 63–69, January 2008.

[71] H. Krawczyk and P. Eronen, "Hmac-based extract-and-expand key derivation function (hkdf)," Internet Requests for Comments, RFC Editor, RFC 5869, May 2010.

[72] C. Ardagna, M. Cremonini, S. De Capitani di Vimercati, and P. Samarati, "An obfuscation-based approach for protecting location privacy," vol. 8, pp. 13–27, March 2011.