LOCATION-BASED ACCESS CONTROL AND TRUST IN LOCATION ESTIMATION

A Thesis

by

Berker Soyluoğlu

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Engineering

in the
Department of Computer Engineering

Özyeğin University
January 2017

LOCATION-BASED ACCESS CONTROL AND TRUST IN LOCATION ESTIMATION

Approved by:

Associate Professor Murat Şensoy
Department of Computer Engineering
Özyeğin University

Assistant Professor Barış Aktemur
Department of Computer Engineering
Özyeğin University

Professor Pınar Yolum
Department of Computer Engineering
Boğaziçi University

Date Approved: 7 December 2016

To my family and friends

ABSTRACT

With the introduction of touchscreen smartphones our way of life changed. Indie developers are in a rush to find the next big thing. Available sensors and the powerful mobile operating systems made location aware applications more and more important. Access to information now depends on users location and we need to make sure the access is granted for the right reasons to the right user. However as the influence of location increased users started searching for ways to cheat the system. There are applications and real life examples that show once the data is highly restricted with the users location, users try to find ways to access that information. In this thesis I'm going to provide extensions to a well known Access Control Markup Language XACML, that allows us to evaluate geospatial access control policies. Also by adding global trust algorithms to the system to avoid fabricated claims made by malicious users we can make the overall system more robust.

ÖZETÇE

Dokunmatik ekranlı akıllı telefonlar yaşam şeklimizi ciddi şekilde değiştirmeyi başardı. Bağımsız yazılımcılar bir sonraki büyük uygulamayı üretmek için yarış içerisine girdiler. Güçlü mobil işletim sistemleri ve cihazlarda bulunan kuvvetli sensörler sayesinde konum servislerini kullanan uygulamalar her geçen gün daha da fazla önem kazanmaya başladı. Konum artık kullanıcı adı ve şifre gibi bilgiye olan erişimi engellemek için kullanılmaya başlandı. Bilgiye erişimde konum bilgisinin etkisinin artması kullanıcıları sistemi kandırma yolları arayışına itti. Bu arayışın başarılı sonuçlarını gerçek uygulamalar üzerinde gerçek örneklerde görmemiz mümkün. Benim bu tezdeki amacım zaten kullanımda olan yetki kontrolü işaretleme dili XACML'e eklentiler yaparak bunu mekansal kuralları da değerlendirebilecek hale getirmek. Ayni zamanda global güven algoritmalarını da kullanarak sahte konum yollayan kullanıcıları izole etmektir.

ACKNOWLEDGEMENTS

There are so many people to thank to it is not possible to provide a comprehensive list of names but I will try my best.

Firstly I would like to thank to my graduate advisor Murat Şensoy. None of this would be possible without his guidance, motivation, patience and knowledge. He has been a great advisor from the start of my studies to the very end. I am grateful to have him as an advisor and can not thank him enough for everything that he has done for me over the years.

Secondly I would like to thank to Saritha Arunkumar for sharing her expertise and guiding me during my research.

I would like to thank to my family this would not be possible without their support, patience and finally for their motivational speeches over the years for me to try harder.

I would like to thank to my supervisors Korhan Kocabıyık and Dağhan Aytaç in Fiba Holding and Rıfat Burak Ulutoprak and Kerem Tiryakioğlu in Vektör Telekom. This would not be possible without their support.

I would also like to thank to all of my Professors both undergraduate and graduate for their knowledge, help and patience.

I would like to thank to all of my friends who have been with me over the years and kept me sane during both my undergraduate and graduate studies. I would like to extend that thanks to Bahar Yılmaz for her help proof reading. Murad Ersoy for his motivation and understanding. Emre Göynügür for his knowledge, motivation and for countless trips to buy coffee in between classes. Finally Mehmet Ahat for his motivational speech where he jokingly said that I could not finish my graduate studies.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Since the release of first generation touchscreen smartphones in 2007, our understanding of a smartphone has shifted drastically. A phone is no longer a device that we can make calls, send short messages and play the occasional snake game with. It is now a device that is competing with our laptops, the average user is now purchasing tablets, phablets instead of PCs[1]. The improvement of mobile data technologies meant that we can be connected whenever wherever. The introduction of advanced mobile operating systems on these powerful devices meant more developers were free to take advantage of this immense market. If we were to take a look at the numbers of largest online stores dedicated for smartphones we can see how big the market is and how developers are filling the gaps. Apple announced that there are over 2 million apps available in the AppStore and those apps have been downloaded 130 billion times since the first launch of the AppStore in 2008[2][3]. Even though Google hasn't announced any numbers like Apple, analytics companies like Statista suggests that there are even more apps on Google Play [4][5].

Compared to a laptop a smartphone has many built-in sensors which makes it a better platform for various apps. My focus will the built-in GPS which is available on almost every smartphone on the market as of today. Companies are building services, applications that require the existence of a built-in GPS. Some even depend on the fact that the user is being honest about his/her whereabouts because their monetization strategy depends on it. Companies that provide online streaming services, companies that sends drivers to pick you up, companies that play matchmakers or simply companies that are hosting games. All of which depend on the user being

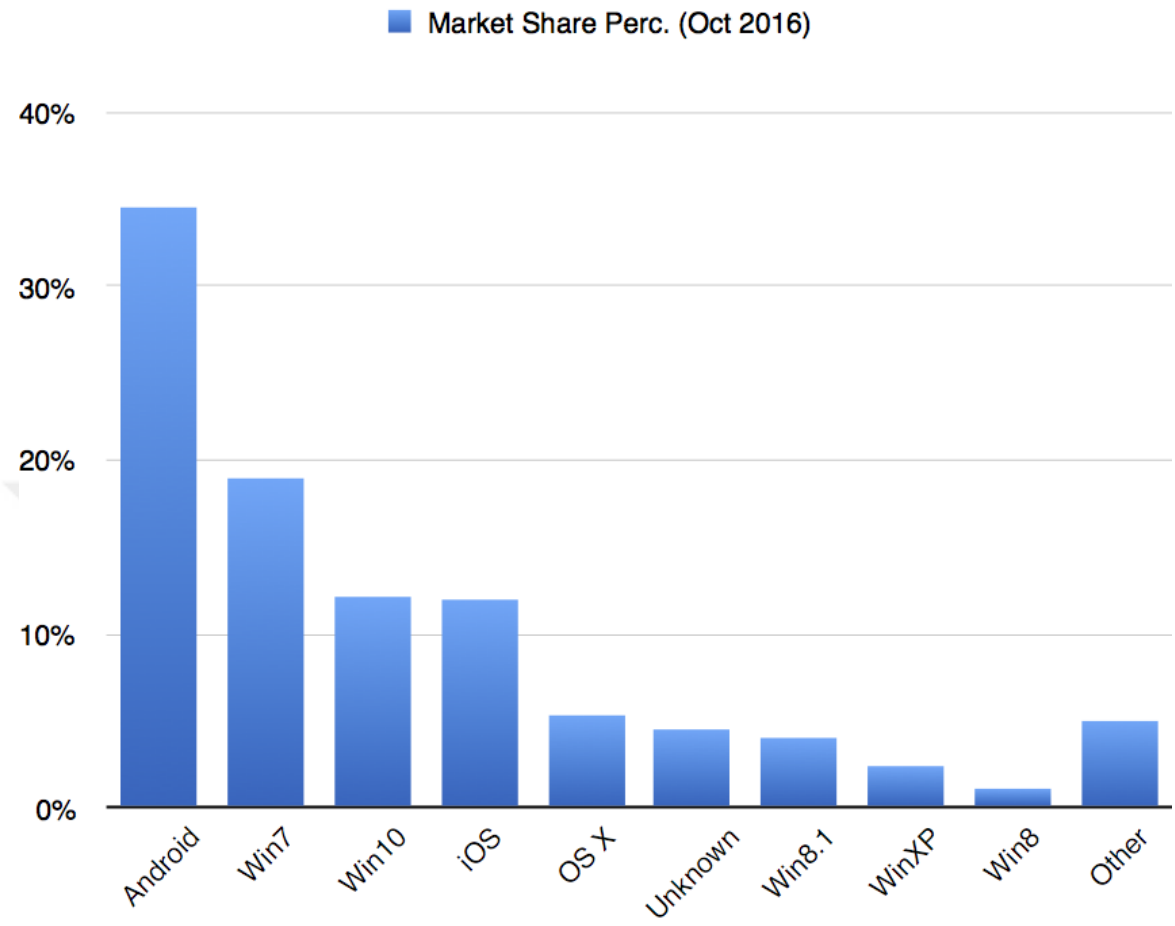

Figure 1: Market share of operating systems as of Oct. 2016

honest.

There are countless examples of real life applications that require our location to work. They all make use of built-in GPSes that are available in our mobile devices and provide us with a service and/or data depending on our location. However even though GPSes are easy to use and they do provide an accurate reading of the user's latitude and longitude we should keep in mind that this data is generated on the client-side and sent to the server. What we should also know is that the user is not always honest and reliable. The user can do anything and everything to provide you with the data you need in order to access the information he/she wants. There are countless examples of how users are breaking the systems that depend on location for

their own gain.

Netflix one of the largest online streaming service provider has made it's content available to 130 more countries in the first quarter of 2016 and right away had issues with Hollywood about their content licenses. Users started connecting to the servers using VPNs to hide their true locations. After which Netflix blocked VPNs from accessing the service which lead to many unhappy customers and probably cancellations of some of the subscriptions[6].

Uber one of the largest transportation company, allows users to request rides for a specific trip. The price of that trip depends on the number of request made from that area i.e. surges. Users have figured out that these surges are usually limited to a street so they started requesting the ride to an other street nearby and once the driver calls the user, they correct the address and avoid paying the full fare of that trip[7]. There are even some cases where a user's account is hacked and the hacker piggybacks of that user to get free rides by requesting rides that have nothing to with the original user's location[8].

Tinder is one of the largest matchmaking application having more than 1 million paid customers as of May 2016[9]. The app matches people depending on their likes and their location. But if you are traveling and want to match with somebody far away from your current location you need to become a premium member and pay a monthly subscription fee. There are forum entries in which people discuss how to overcome this restriction.

Pokémon Go largest location-based augmented reality game ever released as of October 2016. The game was released in July 2016. Founder and CEO of Niantic Labs. John Hanke announced that Pokémon Go has been downloaded 500 million times and people walked 4.6 billion kilometers playing the game as of September 7, 2016[10]. However due to the game's logic and some security flaws people managed to forge GPS locations and managed to trick the system[11]. Chart 2 shows Google

search trends for keywords “fake gps”, “gps spoofing” and “Pokémon GO”. We can clearly see that these search trends are closely related. The initial peak on all three graphs are just around the time when Pokémon GO was released. Once people started hacking the system Niantic released an update changing the game dynamics while also fixing the bug that allowed users to locate arbitrary Pokémons. This led to honest players demanding refunds since the application no longer provided the “nearby Pokémon” feature. These requests were eventually fulfilled by Google and Apple based on their store policies[12].

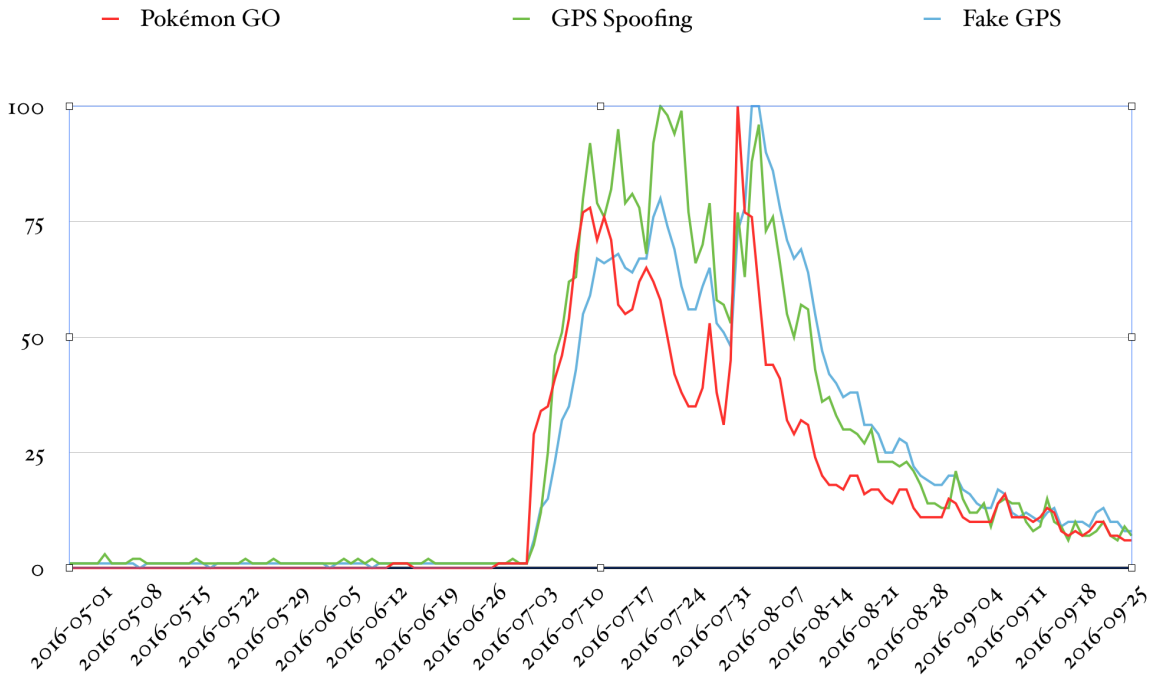

Figure 2: Google trends for Pokémon GO, GPS Spoofing and Fake GPS.

Our lives are now revolving around information, retrieving that information safely is crucial. We are used to providing credentials to access a piece of information like usernames, passwords, one time passwords etc. The examples mentioned above point to a new credential that we are providing to the services we are using. Our access is determined based on our location. There are other ways to retrieve user's location

which will be mentioned in the background section of this thesis, but none of which provide the accuracy or the ease of use of a GPS signal.

In this thesis I will provide an implementation to the geospatial access control markup language specification provided by OGC. Which allows us to decouple the decision making logic that grants the access to the data from the overall application. We can clearly see the need to secure the data using location information. However since such language can not determine if the data provided is trustworthy I will be adding global trust algorithms to increase the security and to minimize these kinds of attacks and filter out malicious users which will in turn allow us to build and maintain an application that is built for change, robust and trustworthy. The rest of this thesis is organized as follows. In the background section, I will briefly go through some of the current methods imposed to retrieve user's location and how we can use policy languages to separate access control decision making from the application logic. I will also discuss current work around this field while briefly introducing the technologies used while implementing this thesis.

In the following chapters I will gradually build up to the latest version of the algorithm and the path it took. I will discuss my previous work with Arunkumar regarding this issue and move on to the final implementation. In the results section I will provide you with results generated using the algorithm and discuss them and their meanings.

CHAPTER II

BACKGROUND

In the introduction section I've laid the motivation for this specific research. In this section I'm going to introduce the technologies used in this thesis. I will go over possible methods that can be used to retrieve user's location while mentioning the drawbacks of these methods. I will also go over previous work done around this topic. Following those I will be mentioning my previous work with Arunkumar et. al. in which we tackled the question where the data requested can only be accessed if the requester is in the desired place. We tried to find a way to evaluate geographical policies allowing us to restrict user access depending on the location where the request was issued[13].

Our understanding of regular old username and password credentials are coming to end. The applications we are using in our daily life require our location to work and depending on the location we provide they serve us with a services and/or data. It is important to secure this data and make sure the right user has access at the right time and at the right location.

When it comes to access control, previous work and research points to XACML (eXtensive Access Control Markup Language), it is an externalized, standardized, policy based, attribute based and dynamic authorization system[14]. Research on XACML shows it can be used to solve several issues and improve security of the applications. Arunkumar et al. also targeted this issue in their previous work and proposed a new architecture that introduced XACML to the mobile devices[15].

However for us to evaluate a policy that depends on the requesters current location we have to provide that location. There are several ways that we can use to retrieve

location information from the user those are GPS, Mobile Phone Tracking, LAN Reporting and Geographic IP lookup. All of which have a real life applications associated with them. But we should keep in mind, since we are gathering this location information from the client side, it is possible to change this data before it reaches to our servers.

2.1 Location Retrieval

2.1.1 GPS

Global Positioning System was developed by the U.S. Military for military use only. However it is now publicly available to any civilian with a GPS receiver. The system has three segments, space segment that has 28 satellites which orbit around the Earth. At any given time there exists a clear line of sight from at least 4 of these satellites to the user segment. The user segment is defined as the receivers that we use to navigate. Final segment is the control segment which has ground stations and checks if the satellites are working correctly. GPS receivers gather the data transmitted from the satellites and calculates it's location using the triangulation method[16].

Almost all smartphones currently available in the market have their own GPS receivers built-in. There is even wearable technology that incorporate a built-in GPS[17]. However since these devices are open to development there are ways to spoof the GPS data generated within the system. Even some operating systems provide ways to fake the GPS signal for debugging purposes[18].

2.1.2 Mobile Phone Tracking

Mobile phone tracking is the way of finding the mobile phones location by the means of cell towers. Several methods are used while calculating the location of the user. One of these methods incorporate knowing the connected Base Transceiver Station (BTS), knowing the time advance which is the round trip delay between the device and the connected BTS and knowing the signal strength of other BTSs in the vicinity.

With these values it is possible to calculate an estimate of the location of the device. It is possible to narrow the the estimate significantly using these methods[19].

However some of the currently available smartphone operating systems do not provide an interface to gather such information. Even if they do many of the hardware vendors do not provide such functionality[20] [21]. Requesting this data from the carriers creates huge privacy issues and it is nearly impossible to get this information, showcased by Malte Spitz at a Ted talk in 2012. He showcases how accurate these readings can be and how his life was retained by Deutsche Telekom[22].

2.1.3 LAN Reporting

LAN Reporting is a concept which tries to estimate the location of the device using WLAN routers. It uses the same ideas used to locate a mobile phones using BTSs. But in a smaller scale and indoors. It uses signal strengths to estimate distance from the routers and using a predefined set of data that was created during the configuration of the system it tries to estimate the location of the device within the premises[23].

The system has some drawbacks; if the user is indoors there are many obstacles that block the signal from reaching the receiver which creates the requirement of a configuration data. On the other hand the approach is useful only when navigating indoors and it will be difficult to scale this concept and make it available outdoors.

2.1.4 Geographic IP Lookup

This process maps your IP to a region, even though it will not always reflect your exact location it will provide a ballpark. How this works is IANA (Internet Assigned Numbers Authority) allocates large IP ranges to RIRs (Regional Internet Registry) which in turn allocate IP addresses to ISPs (Internet Service Provider) and those are assigned to the end users. Every RIR maintains a whois server to hold which IP is assigned to which ISP and in some cases to which end user. Services who provide

this lookup basically does a mining of these whois records[24].

This method is used mainly on websites which try to provide regional information and/or data. There are services that allow users to change their geographic signature by allowing them to connect through their servers. Users use these proxies to avoid restrictions imposed to their network or to view content that is not available in their geographical region. There are lists that provide the IPs of these service providers but the use of a proxy is not limited to registered service providers. Tor network allows any user to become a proxy for any user that uses the Tor network. Thus when the location of the user is queried by Geographic IP Lookup the system can only see the last node that made the request in this case the proxies location will be returned[25].

Using one or more of these techniques the application can improve its security to avoid forged locations. Developers can also add simple anti-spoofing strategies to avoid malicious users. Adding speed limitations to an application that requires the user to be walking, adding a history of locations to block any jumps that are not possible in real life scenarios. Using map APIs to check if the given location is on a road to an application that requires the user to be driving.

However all of the above methods still suffer from basic attacks, to name a few cross-site request forgery and man-in-the-middle. If we were to reverse engineer the requests that were made to the server, we could forge requests and send those to the servers to request a data which we are not entitled to. Pokémon GO suffered from these attacks users were able to reverse engineer requests made by the smartphones and create their own and send any location details they desired[26]. There are even extremely creative hacks where the user places the phone in a radio-frequency-shielded box, and generate fake GPS signals via the help of a signal generator[11].

2.2 GeoSpatial Access Control for Mobile Devices

In our previous work with Arunkumar we've explored possible ideas on how we can achieve a trustworthy API. We focused on XACML and its uses. XACML (eXtensive Access Control Markup Language) is a specification put forth by OASIS (Organization for the Advancement of Structured Information Standards). It is a markup language that creates ways to express business requirements and ways to create policies that can change as the requirements evolve. The abstraction created with the use of XACML policies between the access control mechanism and the overall application enables a whole new flexibility to the developers. Allows applications that are build for change. XACML describes both a policy and an access control decision request/response language. The policy language is used to describe general access control requirements and has standard extension points. The request response language lets you form a query to ask whether or not a given action should be allowed. The response always includes an answer about whether the request should be allowed[27][13].

Since the work done requires geographic functions to be evaluated. We should know if a given point is located within some predefined boundaries. XACML alone isn't enough. This is where GeoXACML comes in to work. GeoXACML is an implementation specification put forth by OGC (Open Geospatial Consortium). The specification builds on the work provided by OASIS and it adds the ability to evaluate geographic policies[28]. However at the time of this work we failed to find a functioning open implementation of the specification. To prove the concept I've implemented some of the necessary components that were stated in the specification which will be dealt with in the coming chapters of this thesis.

The focus is to use a policy language to evaluate the requests made by the user. Thus giving more flexibility to the framework. However this does not provide any security over GPS spoofing and the losses created to the companies. As far as XACML is concerned the data provided to it is authentic and trustworthy. The weakest link

in the system is the location and failing to authenticate this piece of information leads to the failure of the entire system. To do so we should isolate the user and the computation about the authenticity of the data should be done on the servers where the user presumably does not have an access to.

2.3 Proof in Location

Many have suggested proofs to be delivered to the servers those proofs consists of wireless access points, trusted devices placed by an authority or simply other devices that are using the same system. Even with these security measures there are still security vulnerabilities of these system such as colluding malicious users that engage in a wormhole attacks. Saroiu et al. have pointed out that the next "killer app" is location[29]. Their work is somewhat similar to Singh et al. where APs are used for location proofs instead of determining location of a device indoors. Lou et al. tried to solve this problem using cryptography and placing trusted devices to avoid collusion attacks[30].

Other work presented from here on focuses on how other users can be used to provide a proof of location. The use of Bluetooth enabled devices and a proof server that receive data from devices using the framework.

Hasan et al. focused on path and history critical scenarios in their works published in 2011 and in 2016. Their first work makes use of hash chains to prove the order of any arbitrary sub-sequence of the location history created by the user overtime. They introduce a witness endorsed scheme for generating collusion-resistant location proofs. The work they provide comes with a proof of concept Android Application[31]. Their second work WORAL improves upon this by adding trusted Rasberry Pis to improve collusion detection using them as Location Authorities. They also manage to show a ready to deploy framework and an application that shows how the system is easily deployable and usable[32].

Zhu et al. makes use of co-located Bluetooth enabled mobile devices which generate proofs for each other. On the other hand their work also focuses how user privacy is preserved by using periodically changed pseudonyms. They also focus on the fact that the framework is also easily deployable and usable[33].

Wang et al. propose entropy-based trust evaluation approach to detect fake proofs resulting from collusion attacks. Like Zhu et al. their work also uses co-located Bluetooth devices and uses them as proofs[34].

Arunkumar et al. suggested the use of global trust schemes to ensure the authenticity of the user. In their work Eigentrust and peer trust algorithms are suggested with the addition of Bluetooth reports collected from co-located devices. They provide experimental results to their approach by the means of open data sets collected by various providers[35].

2.4 Zero-Conf Networking

This thesis not only uses co-located Bluetooth devices it can also make use of any device that implements the Bonjour framework. The issue at hand is proven to have a significant impact on many applications currently used by millions of users. Rather than using vanilla Bluetooth implementation my approach is to use Zero-Configuration Networking to be more specific using one of the major implementations backed by Apple. Zero-Conf Networking is a specification introduced by The Internet Engineering Task Force that allows us to take two computers, and connect them with a crossover Ethernet cable, and have them communicate usefully, without using a techie friend to help us out[36] Zeroconf is not limited to networks with just two hosts it is scaleable and used seamlessly in many of the products we take for granted as of today. In 1980s Apple handled this protocol very well in a product they called AppleTalk. It is stated that if you took a group of Macs and connected them together with LocalTalk cabling, you had a working AppleTalk network, without any expert

intervention[36].

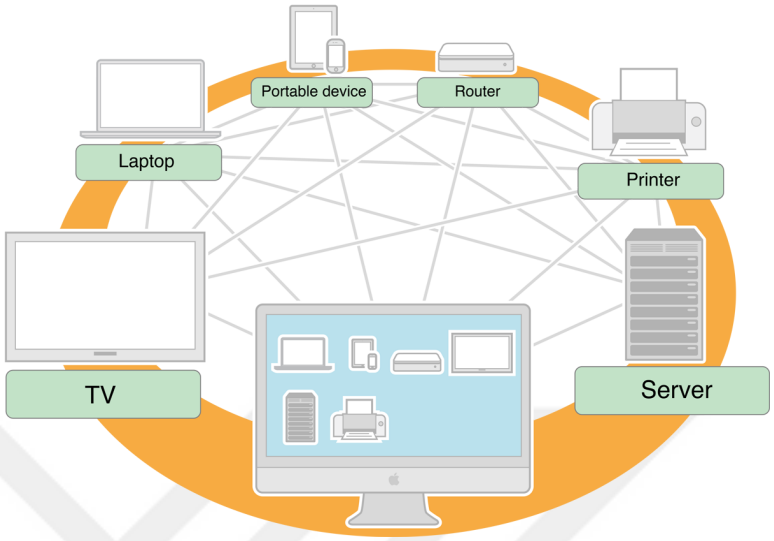

Figure 3: A Bonjour Network.

,

Bonjour on the other hand is the official implementation of Zero-Configuration Networking made available open source by Apple. Bonjour discovers other devices ranging from printers computers and mobile devices and the services offered by those devices using multicast Domain Name System. Currently all major mobile operating systems support multicast Domain Name System[37][38][39]. Bonjour is not only implemented in computers and mobile devices they are available in printers it also provides a programming interface that is accessible from Cocoa, Ruby, Python, and other languages[39].

Previously mentioned state of the art frameworks all tried to solve reliability by having co-located devices having more devices meant a better proof set. By making use of Bonjour a widely used Zero-Configuration Networking implementation proposed solution increases reliability of the overall system. Instead of using location authorities we can now make use of any stationary device that supports Bonjour like TVs, routers and printers.

This thesis uses a centralized version Eigentrust developed by Kamvar et al. to

overcome collusion by malicious peers as suggested by Arunkumar et. al. in their work[35]. Eigentrust presents a distributed and a secure method to compute global trust values, based on Power iteration[40]. The developer of the application which makes use of the framework then can decide if the trust value of that user is noteworthy enough to have access to that data.

A policy engine that makes it possible to write and evaluate geographic policies coupled with a global trust algorithm to avoid various attacks and a client-end that supports a wide range of devices the solution proposed in this thesis proves to be an easy to deploy and use collusion resistant framework for location sensitive applications. I will show that even if the user somehow generates the request and forges the location data, it is not possible for him/her to fool the system. Since the application requires proofs and makes use of global trust algorithms to avoid collaboration. This corroborating measure improves the systems overall reliability as the number of users increase.

CHAPTER III

LOCATION BASED ACCESS CONTROL USING GEOXACML

As the access of powerful smartphones and mobile internet got easier our needs shifted and created a more connected world. Location started playing a bigger role in the applications that we use daily. The need of location based access control grew. This need is proven by the real world applications mentioned in the introduction section of this thesis. All of those applications try to restrict user access depending on the location of the user. They all use location as a credential when accessing a piece of data.

In our work with Arunkumar about GeoSpatial Access Control for Mobile Devices[13]. We tackled that question and tried to find a solution to the problem where the user should be in a specific location to access a piece of data.

XACML is widely popular when it comes to a decoupled access control mechanism. It has also proved that is has many use cases[14][15]. However for our research XACML was not enough we needed to write policies that could evaluate GeoSpatial functions. OGC (Open Geospatial Consortium) proposes GeoXACML as a standard when it comes to geospatial policies. GeoXACML is an implementation specification. It extends XACML specifications to allow geographic policies to be implemented. Since at the time of this work there were no open implementations to GeoXACML I've implemented some of the functions to achieve a proof of concept application.

Our work also contains a proof of concept application that shows how this can be used in a health care environment. As patient data can be sensitive our application allows doctors and health care physicians to access patient data if and only if they

are located within the hospital they are working in. This work has been published and presented in the IEEE Region 10 Symposium (TENSYMP) on 2015[13].

This chapter will first cover some basic knowledge about XACML and GeoX-ACML, go over the proof of concept application finally discuss the overall architecture of the application.

3.1 XACML

XACML (eXtensive Access Control Markup Language) a specification put forth by OASIS (Organization for the Advancement of Structured Information Standards). It is a markup language that in turn creates an access control policy language. It defines a way to write policies and access requests and which then can be evaluated into an access control request result. In this work we've used the open implementation of Sun's XACML. All of the extensions mentioned to the implementation are done on Sun's XACML implementation[41].

The XACML architecture has 5 components and each has a unique job to fulfill those components and their tasks are,

1. PAP (Policy Administration Point)

   This is where all the policies are administered.

2. PDP (Policy Decision Point)

   The point where the policies are evaluated by the information gathered from PIP and PRP.

3. PEP (Policy Enforcement Point)

   The point where the users interact with. This is the point where the user makes his/her request and this is the point which returns the resource to the user if allowed by the PDP.

4. PIP (Policy Information Point)

   The point where the information is stored to evaluate the policies.

5. PRP (Policy Retrieval Point)

   The point where the policies are stored. It can be anything from a file system to a database.

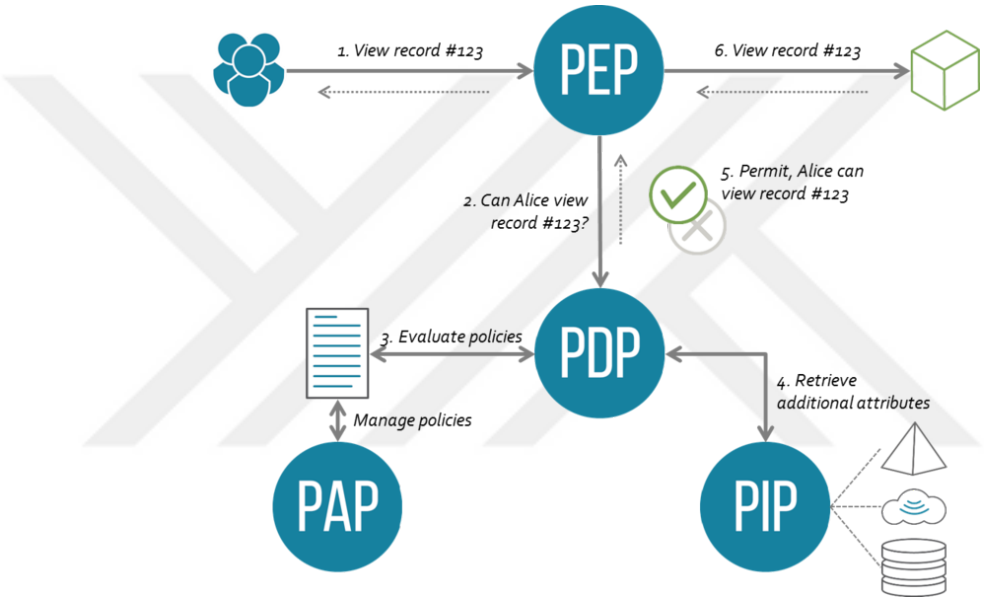

Figure 4: XACML Architecture, Flow,

The architecture and the authorization flow of XACML is shown in figure 4. This flow does not change with GeoXACML since the extensions made are within the nodes and they do not change the overall architecture of the policy language.

1. Depicts the request made by the user to the system through PEP.

2. PEP asks PDP if the user in question has access to the requested resource.

3. PDP pulls the policies that are required to evaluate the request made by the PEP.

4. PDP gathers information that are required to evaluated the policy retrieved from the PAP.

5. PDP returns the result to PEP.

6. PEP requests the resource that was originally requested by the user.

7. The resource is returned to the user.

```
1  <Policy PolicyId="ExamplePolicy"
2          RuleCombiningAlgId="permit-overrides"><!-- [aa] -->
3    <Target><!-- [ab] -->
4      <Subjects><!-- [ac] -->
5        <AnySubject/>
6      </Subjects>
7      <Resources><!-- [ad] -->
8        <Resource>
9          <ResourceMatch MatchId="anyURI-equal">
10           <AttributeValue DataType="anyURI">
11             student-list
12           </AttributeValue>
13           <ResourceAttributeDesignator DataType="anyURI"
14                                        AttributeId="resource-id"/>
15         </ResourceMatch>
16       </Resource>
17     </Resources>
18     <Actions><!-- [ae] -->
19       <AnyAction/>
20     </Actions>
21   </Target>
22   <Rule RuleId="ReadRule" Effect="Permit">
23     <Target><!-- [ab] -->
24       <Subjects><!-- [ac] -->
25         <AnySubject/>
26       </Subjects>
27       <Resources><!-- [ad] -->
28         <AnyResource/>
29       </Resources>
30       <Actions><!-- [ae] -->
31         <Action>
32           <ActionMatch MatchId="string-equal">
33             <AttributeValue DataType="string">
34               read
35             </AttributeValue>
36             <ActionAttributeDesignator DataType="string"
37                                        AttributeId="action-id"/>
38           </ActionMatch>
39         </Action>
40       </Actions>
41     </Target>
42     <Condition FunctionId="and"><!-- [af] -->
43       <Apply FunctionId="string-equal"><!-- [ag] -->
44         <Apply FunctionId="string-one-and-only">
45           <SubjectAttributeDesignator AttributeId="username"
46                                       DataType="string" />
47         </Apply>
48         <AttributeValue DataType="string">
49           Alice
50         </AttributeValue>
51       </Apply>
52     </Condition>
53     </Rule>
54   </Policy>
```

Figure 5: A Policy Example.

Figure 5 represents a policy example with only one rule and that rule has only one condition. A policy can contain more than one rule tag but only one target tag. Each rule evaluated returns it's Effect attribute when the evaluation is true. Tags subject, resources and actions located in lines 4, 7, 18, 24, 27 and 30 are used to match the policy to the request. Using these tags and evaluating only the policies that match the request improves the performance of XACML drastically[41]. Figure 6 includes a XACML request, if we were to match subject, resource and action tags

```
<Request>
  <Subject><!-- [ba] -->
    <Attribute AttributeId="username"
               DataType="string">
    <AttributeValue>
      Alice
    </AttributeValue>
  </Attribute>
  </Subject>
  <Resource><!-- [bb] -->
    <Attribute AttributeId="resource-id"
               DataType="anyURI">
    <AttributeValue>
      student-list
    </AttributeValue>
  </Attribute>
  </Resource>
  <Action><!-- [bc] -->
    <Attribute AttributeId="action-id"
               DataType="string">
    <AttributeValue>
      read
    </AttributeValue>
  </Attribute>
  </Action>
</Request>
```

Figure 6: A Policy Request Example.

we can say that this request will result in an evaluation of the previously discussed policy. Once the target, resource and action is matched the function is evaluated. This policy only requires that the username string value found in the subjects tag in the request must match Alice. If so this policy returns permit as a result. Given that there are no other policies in the system that matches the request the overall result will be permit. If we were to have other policies the results will be combined by the predefined policy combining algorithm.

3.2 GeoXACML

GeoXACML on the other hand is merely a set of extensions that allow the PEP to evaluate more functions and understand more objects that can be found in the request and/or policies added to the system. We will go over two attributes GeoPoint and GeoPolygon. GeoPoint describes a point on geographic coordinate system. GeoPolygon is a collection of GeoPoints that when combined creates a polygon. There are also function extensions made to the engine those are GeoContains and the required

one-and-only functions for previously mentioned attributes. Since all tags in the XACML requests are considered as bags, which can contain more than one element *-one-and-only functions are used to ensure that there exists only one value within the bag. These implementations are then injected to the engine on start-up seen on listing 3.1.

Listing 3.1: Engine Startup.

```
....
StandardAttributeFactory standardFactory =
        StandardAttributeFactory.getFactory();
final BaseAttributeFactory newFactory =
        new BaseAttributeFactory(standardFactory.getStandardDatatypes());

newFactory.addDatatype(GeoPoint.TYPE, new AttributeProxy() {
    public AttributeValue getInstance(Node root) throws Exception {
        return GeoPoint.getInstance(root);
    }
    public AttributeValue getInstance(String value) throws Exception {
        throw new RuntimeException("get instance with value is called
            geoPoint");
    }
});

newFactory.addDatatype(GeoPolygon.TYPE, new AttributeProxy() {
    public AttributeValue getInstance(Node root) throws Exception {
        return GeoPolygon.getInstance(root);
    }
    public AttributeValue getInstance(String value) throws Exception {
```

```
        throw new RuntimeException("get instance with value is called
            geoPolygon");
    }
});

AttributeFactory.setDefaultFactory(new AttributeFactoryProxy() {
    public AttributeFactory getFactory() {
        return newFactory;
    }
});

FunctionFactoryProxy proxy = StandardFunctionFactory.getNewFactoryProxy();
proxy.getTargetFactory().addFunction(new GeoContains());
proxy.getTargetFactory().addFunction(new GeoVerify());

proxy.getTargetFactory().addFunction(
        BagFunction.getOneAndOnlyInstance("geoPoint-one-and-only",
                GeoPoint.TYPE));

FunctionFactory.setDefaultFactory(proxy);
...
```

In figures 7 and 8 it is possible to see these extensions in action. If we were to go over the policy example and look at line 38 to the Condition tag. There exists the usage of geo-contains, geoPoint-one-and-only and geoPolygon. geoPoint-one-and-only is used to retrieve the attribute from the subject tag located in the request on line 3. If the engine encounters an object or a function that is unknown, that is not built-in, engine tries to locate it with the added AttributeFactoryProxy or the

FunctionFactoryProxy objects. To initiate the object engine passes the XML node object to the getInstance method of the related datatype or calls the constructor of the function. The creation of a GeoPoint object is simply the execution of the following code on listing 3.2. The node is passed in to the getInstance method which contains the string "41.028514,29.190435" the method parses the node and initiates the object.

For the case of geoPolygon (since polygons would be defined only on the policies) adding each point to the XACML file was going to make the file unreadable and prone to error our solution to the problem was to create a geoPolygon object that takes a string and that string maps to a list of coordinates inside the database and those points are then used to create the polygon. This is then accessed by the Policy Information Point once the policies are evaluating.

Listing 3.2: GeoPoint Implementation.

```
public class GeoPoint extends AttributeValue {
public static final String TYPE = "geoPoint";
public GeoPoint(double lat, double lon) throws URISyntaxException {
    super(new URI(TYPE));
    this.lat = lat;
    this.lon = lon;
}
public static AttributeValue getInstance(Node root) throws
    URISyntaxException {
    String point = root.getFirstChild().getNodeValue();
    String[] values = point.split(",");
    Double lat = Double.valueOf(values[0]);
    Double lon = Double.valueOf(values[1]);
    return new GeoPoint(lat, lon);
}
```

...

```
<Policy PolicyId="ExampleGeoPolicy"
        RuleCombiningAlgId="permit-overrides">
  <Target>
    <Subjects>
      <AnySubject/>
    </Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="anyURI-equal">
          <AttributeValue DataType="anyURI">student-list</AttributeValue>
          <ResourceAttributeDesignator DataType="anyURI"
                                       AttributeId="esource-id"/>
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions>
      <AnyAction/>
    </Actions>
  </Target>
  <Rule RuleId="ReadRule" Effect="Permit">
    <Target>
      <Subjects>
        <AnySubject/>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="string-equal">
            <AttributeValue DataType="string">read</AttributeValue>
            <ActionAttributeDesignator DataType="string"
                                       AttributeId="action-id"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
  <Condition FunctionId="and">
      <Apply FunctionId="geo-contains">
      <Apply FunctionId="geoPoint-one-and-only">
        <SubjectAttributeDesignator AttributeId="location" DataType="geoPoint" />
      </Apply>
      <AttributeValue DataType="geoPolygon">OzU_Campus</AttributeValue>
    </Apply>
  </Condition>
  </Rule>
</Policy>
```

Figure 7: A GeoPolicy Example.

```
<Request>
  <Subject>
    <Attribute AttributeId="location"
               DataType="geoPoint">
      <AttributeValue>41.032239,29.258025</AttributeValue>
    </Attribute>
  </Subject>
  <Resource>
    <Attribute AttributeId="resource-id"
               DataType="anyURI">
      <AttributeValue>student-list</AttributeValue>
    </Attribute>
  </Resource>
  <Action>
    <Attribute AttributeId="action-id"
               DataType="string">
      <AttributeValue>read</AttributeValue>
    </Attribute>
  </Action>
</Request>
```

Figure 8: A GeoPolicy Request Example.

Once we got the engine working we started the put pieces together to create a server that can serve us with the mock data and evaluate requests made by the proof of concept mobile application. The application architecture is shown in figure 9. The flow of the application starts with the request made by the mobile application. It creates a requests and sends it to the server. On the server side PEP interrupts the request and converts it to a XACML request and passes that to the PDP. PDP loads the required policies from PAP and evaluates the policies using the extension provided. While PDP is evaluating the policies it gathers the information needed to process the required policy from PIP like the previously mentioned GeoPolygon data.

Once PDP evaluation is completed it returns a result to PEP which can contain one of the following,

1. Permit, access is granted
2. Deny, access is denied.
3. Not Applicable, which means the request can not be answered by this service
4. Indeterminate, which means that an error has occurred PDP failed to come to a decision

PEP then depending on the result returned from the PDP accesses the data from the database and returns it to the client application.

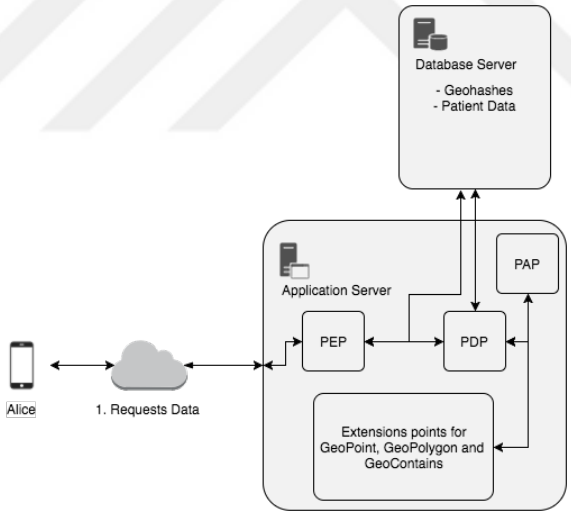

Figure 9: Geospatial access control framework architecture

3.3 Health Care Application

In our work with Arunkumar we've also implemented a proof of concept iOS application. In this scenario we simulate an application where doctors can access patient records from their mobile devices. Since patient data can be sensitive we've added some restrictions. Those restrictions only allowed doctors to access patient records

while they are working i.e. when they are at their office since it is unlikely for a health care physician to access patient records at his/her home. To achieve this we've written GeoXACML policies shown in figures 10 and 11. A simple run results in the screenshot provided in figure 12 if the result returned by the PDP is *Permit*.

```
<Request>
  <Subject>
    <Attribute AttributeId="doctor"
               DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>John</AttributeValue>
    </Attribute>
    <Attribute AttributeId="patient"
               DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>Alice</AttributeValue>
    </Attribute>
    <Attribute AttributeId="location"
               DataType="geoPoint">
      <AttributeValue>41.028514,29.190435</AttributeValue>
    </Attribute>
  </Subject>
  <Resource>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
               DataType="http://www.w3.org/2001/XMLSchema#anyURI">
      <AttributeValue>http://example.com/medical/reports/alice</AttributeValue>
    </Attribute>
  </Resource>
  <Action>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
               DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>read</AttributeValue>
    </Attribute>
  </Action>
</Request>
```

Figure 10: Policy Request for Healthcare Application.

```
<Policy PolicyId="ExamplePolicy" ...>
  <Target><AttributeValue>/medical/reports/*</AttributeValue></Target>
  <Rule RuleId="ReadRule" Effect="Permit">
    <Target>
      <Subjects>  <AnySubject/> </Subjects> <Resources> <AnyResource/></Resources>
      <Actions> <Action>...
          <AttributeValue DataType="...XMLSchema#string">read</AttributeValue>
      ...</Action>  </Actions>
    </Target>
  <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
        <SubjectAttributeDesignator AttributeId="doctor" DataType="...#string" />
      </Apply>
      <AttributeValue DataType="...XMLSchema#string">John</AttributeValue>
    </Apply>
    <Apply FunctionId="geo-contains">
      <Apply FunctionId="geoPoint-one-and-only">
        <SubjectAttributeDesignator AttributeId="location" DataType="geoPoint" />
      </Apply>
      <AttributeValue DataType="geoPolygon">
                    41.027514,29.189435;
                    41.029514,29.189435;
                    ...
      </AttributeValue>
    </Apply>
  </Condition>
  </Rule>
</Policy>
```

Figure 11: Policy for Healthcare Application

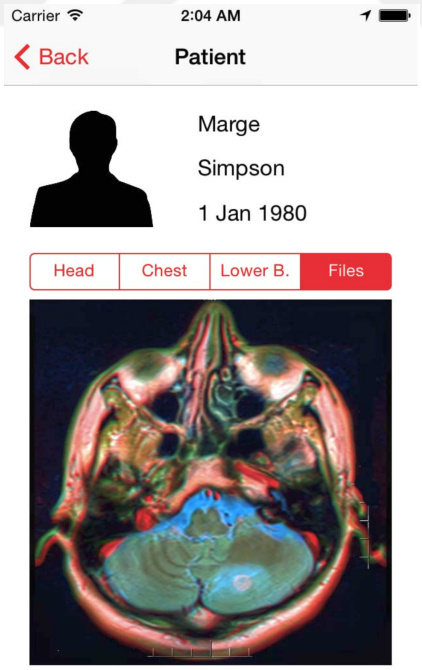

Figure 12: Application Screenshot

CHAPTER IV

GLOBAL TRUST ASSISTED LOCATION BASED ACCESS CONTROL USING GEOXACML

However the previously mentioned implementation only enables us to write applications that separate access control decision from the application logic. It allows us to make complicated decisions depending on the users' whereabouts but it fails to ensure that the data provided by the user is genuine and trustworthy. Everything presented up to this point will suffer the same attacks various production applications suffered as presented in the introduction section of this thesis. Even if we have means to write and evaluate geographical policies the access control provided by this mechanism will not work given that the user can send any information he/she desires. This might even be the reason why GeoXACML is not popular in enterprise applications and why we've failed to find an open and working implementation of it.

To improve this it is crucial for the system to decide if a given user is being honest or not. To do so we propose to use global trust metrics to assist the GeoXACML implementation that was presented in the earlier chapters. If the system somehow calculates and assigns trust values to the users we can avoid the attacks mentioned in the introduction chapter of this thesis, by deciding to serve the data depending on the neighboring users and how trustworthy they are and/or depending on the trustworthiness of the user requesting the data.

I've used Eigentrust algorithm in this thesis, it is possible to swap this algorithm and use others if needed. For us to use global trust metrics we need to create a feedback graph, a graph of users that are in the system, a graph that represents the interactions between the users. Interactions are defined as two devices being in close

proximity with one another. In the new and improved version of the application we require users to report their location with the devices that they sense nearby via Bluetooth and/or WiFi. Like many have mentioned It is possible to use other devices to prove that the device in question is indeed where it claims to be[31][32][34][35]. By making some changes it was possible to add this feature to the existing implementation.

4.1 Zero-Conf Networking

For us to provide proofs for co-located devices we should first discover co-located devices. All previous works mentioned has proof of concept implementations that use vanilla Bluetooth in this thesis I purpose the use of Bonjour framework an implementation of Zero-Conf Networking enabling us to use more devices like printers TVs if we were to improve on this implementation. Also it is now possible to use WiFi to discover services which rules out the case when the Bluetooth is turned off on some devices. If we were to consider the fact that in many cases users prefer to turn off their Bluetooth to preserve battery power.

To support this I've added some changes to the client application now it makes use of the Bonjour to find published services on a network. The application starts by broadcasting a service with the newly defined service name and a predefined service type. The app then shares the service name and GPS location with the server. This information is then used to provide a proof for this user and/or other users. The execution flow of the application can be summarized as;

1. User launches the application.

2. Application assigns an arbitrary name to the device.

3. Application shares the device name and GPS location with the Server.

4. Application registers a run loop to the network browser to identify all co-located devices and changes to those devices.

    4.1. Application sends changes detected on co-located devices i.e. a device is no longer available or a new device is discovered.

5. User logs in to the application.

    5.1. Application sends the name assigned to the device.

    5.2. Server associates the name assigned to the device with the user.

6. User navigates to a screen where a data request is performed.

7. Application sends the request accompanied with the list of co-located devices.

8. Server recalculates the trust values.

9. Server runs the policy engine and returns the requested data if it is allowed by the policy engine.

To publish a service NetServiceBrowser is instantiated with the values domain, type, name and port[42]. The type argument is the type of the service we are providing and it should be the same across all devices and it must be registered with the Internet Assigned Numbers Authority (IANA)[43]. If a domain and a port are not given NetServiceBrowser picks the best domain and port for us[42]. The example initiation can be seen on listing 4.1.

Listing 4.1: NSNetService alloc and init.

```
self.netService = [[NSNetService alloc] initWithDomain:@"local."
                                                  type: @kGeoClientType
                                                  name:deviceName port:0];
self.netService.includesPeerToPeer = YES;
[self.netService publishWithOptions:NSNetServiceListenForConnections];
```

The application also uses NetServiceBrowser to browse services to do so we have to assign a delegate to the *netService* object once we implement the delegate methods *netSerice* object starts calling our delegate methods in which we can implement the run logic of our application. For our needs we only need to implement two delegate methods 4.2

Listing 4.2: NSNetService delegate methods.

```
- (void)netServiceBrowser:(NSNetServiceBrowser *)browser
    didRemoveService:(NSNetService *)service moreComing:(BOOL)moreComing
- (void)netServiceBrowser:(NSNetServiceBrowser *)browser
    didFindService:(NSNetService *)service moreComing:(BOOL)moreComing
```

Once NetServiceBrowser calls our delegate methods we can then update our nearby devices list and update our proof by sending reports to the server like the one shown on listing 5.1. This triggers the calculation of the new trust values for all user.

4.2 Global Trust

Discovering devices is only the start, once each user transmits their findings to the server the server should now leverage this data and create a graph of all the users that are using the system. Having this information makes it possible for us to calculate a global trust value for each user. In this thesis I've used a centralized version of Eigentrust.

Eigentrust is an algorithm that calculates and assigns a global trust value for each peer located in a peer-to-peer file-sharing system. It does this using the history of uploads by that peer[40]. As Kamvar et al. mention the anonymous nature of these networks allows malicious users to spread inauthentic files to the system. Their goal

was to isolate these malicious peers from the network. The same open and anonymous nature can also be achieved in location sensitive applications. The increase of inauthentic users may put the overall system into jeopardy and cause authentic users to leave the system entirely.

In Eigentrust each user defines a trust value for each peer it interacts and assigns a local trust value to that peer. It is defined as the difference between satisfactory downloads and unsatisfactory downloads shown in equation 1. The s_{ij} value calculated here become the edge weight of the aforementioned user graph.

$$s_{ij} = \text{sat}(i, j) - \text{unsat}(i, j) \tag{1}$$

However in our case peers do not know anything about the peers they interact with. Also for the sake of the system our calculations are done in a central location. So the local trust values for each peer is calculated using the equation suggested by Arunkumar et al. in their previous work titled Global Attestation of Location in Mobile Devices shown on equation 2[35].

$$s_{ij} = n/(n + m) \tag{2}$$

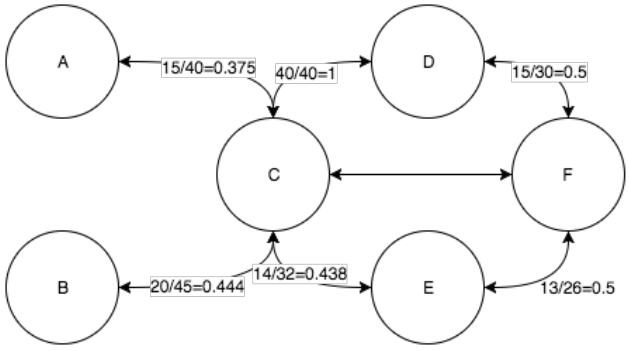

Figure 13: Feedback graph created by using the equation 2

s_{ij} is calculated using positive feedback represented by n and negative feedback

represented by m. Positive feedback is defined as, if two devices that claim to be co-located those two devices must include each other in their location reports. Negative feedback is defined as the opposite, if one of the devices fails to report the other even though they claim to be co-located. Once all these values for S are calculated global trust values can be calculated using the basic centralized Eigentrust algorithm show in equation 3[40]. An example user graph with the edge weights can be seen on figure 13. Once these changes are made the systems overall architecture takes the form depicted in figure 14.

$$
\begin{aligned}
\bar{t}_0 &= \bar{e} \\
&\boldsymbol{repeat} \\
\bar{t}^{(k+1)} &= C^T \bar{t}^{(k)} \\
\delta &= ||t^{(k+1)} - t^{(k)}|| \\
&\boldsymbol{until} \delta < \text{error}
\end{aligned}
\tag{3}
$$

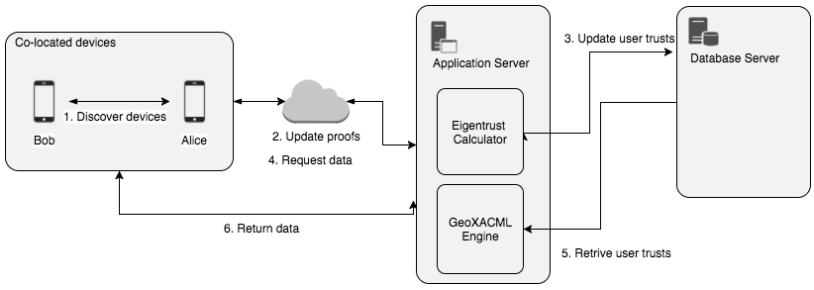

Figure 14: Application architecture with trust modules added.

CHAPTER V

RESULTS

To prove that the algorithm works as intended I've done extensive simulations with multiple values. I'm going to go over these results in this chapter. The experimentation simulates users wandering inside Ozyegin University Campus. First we need to define two separate regions where the users are spawned. First region is for the real users where the data access is granted. Second is for the malicious users where the data access is not granted. After all users are spawned in their regions the simulation starts to move the users within their regions. Each iteration of a simulation goes over every user and tries to move the user with a probability of p. p is defined in a way that it increases on each iteration which means as the user fails to move on iteration i, the probability of this user moving increasing as the time progresses. p is defined as $p = e^{-5/n}$. The movement of the user is defined as a reasonable walking distance so there are no teleportations in the system.

On each iteration we collect reports from all users those reports contains the location of the user and a list of nearby devices, a sample report can be seen in listing 5.1. The interaction between devices are generated depending on their proximity. If two devices are within a predefined distance away from each other the simulation fabricates an interaction between those two devices.

Listing 5.1: User Report Example.

```
{
   "lat":41.027514,
   "lng":29.189435,
   "nearbyDevices":["bob", "eve"]
```

```
}
```

On each iteration the simulation calculates the trust value for each user and the error for the overall system. Since we are aware of which users are honest and which are not, we can calculate the error by simply subtracting the calculated trust value from the expected trust value of the user in question. Simulation continues until the error converges to either 0 or 1 or until a predefined iteration count is reached.

The simulation is performed with two user densities. First a reasonably sparse environment where each user has at most two neighbors at a given time. That setup has 10 users and depending on the experiment 3, 4, 5, 8, 9 or 10 malicious users. Second a reasonably dense environment where each user has at most ten neighbors at a given time. That setup having 100 users and depending on the experiment 30, 40, 50, 80, 90 or 100 malicious users.

The regions used in the experiments stay the same allowing us to increase the density by only increasing the number of users participating in the simulation. In figures 15 and 16 it is shown how two densities can result if we were to plot the users initial locations given by the simulation. Green pins represents the users pink pins represents the malicious users and the purple pins represent the location which the malicious users claims to be. The images also contain a square area which represents the area in which data request is granted.

The experiment is repeated with collaborative and non-collaborative malicious users. A malicious user is defined as a user that is not located in the region where the data access is permitted and where all the users are located. The collaborative malicious users have a "spy" within the system and they are aware of all the changes that are made on the area that they claim to be in. This allows them to successfully report all users and their actions. I've tried to show that the system will provide reasonable results even when faced with a fully capable collaborative malicious user

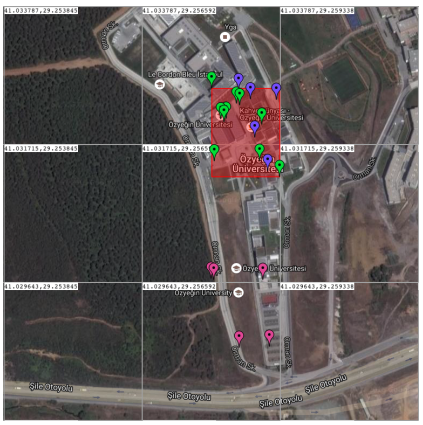

Figure 15: Possible user distribution with 10 users and 5 malicious users.

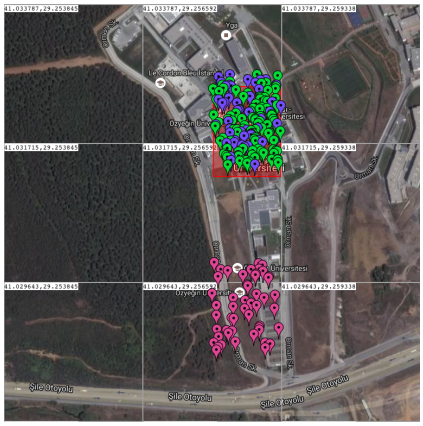

Figure 16: Possible user distribution with 100 users and 50 malicious users.

set. Non-collaborative user on the other hand only mocks a location and sends only the devices that are co-located with it.

So when a collaborative malicious user sends a status report it includes all users located where he/she is presumably located plus all the malicious users that claim to be on that location. This way malicious users increase their proof list and try to dominate the system.

Since in real life scenarios it is highly unlikely for all malicious users to collaborate all together and provide 100% true reports about the entire system. Same simulations are repeated with non-collaborative malicious users and we can see how the system quickly eliminates malicious users and provides a stable result in all cases where the users do not collaborate.

In the following charts we can clearly see that the Eigentrust algorithm quickly isolates malicious users even when they collaborate.

The charts below show how error progresses over time as the iteration increases. Error bars are defined as the standard deviation of error over several runs of the same simulation. Having small error bars means that the system was consistent over several runs. There are instances of the simulation where the values show that malicious users and normal users are head to head and the system fails to converge to a single group

of users, thus having large error bars. There are also some instances where malicious users overwhelm regular users and the error converges to one.

If we were to look at the graphs closer they can be divided in to three categories to be examined further.

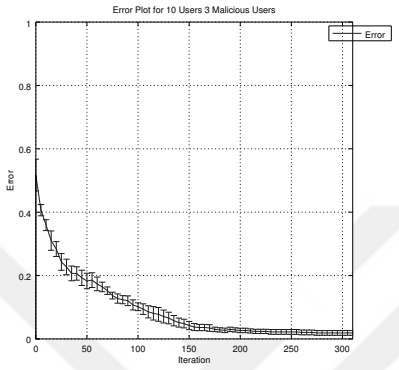

Figure 17: 10 Users and 3 Malicious Users

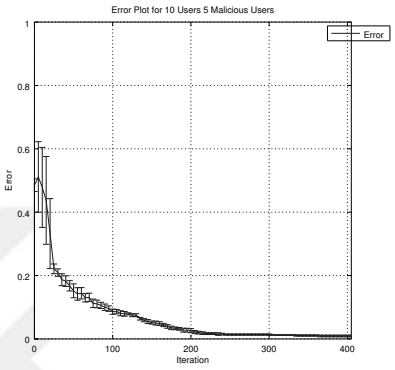

Figure 18: 10 Users and 5 Malicious Users

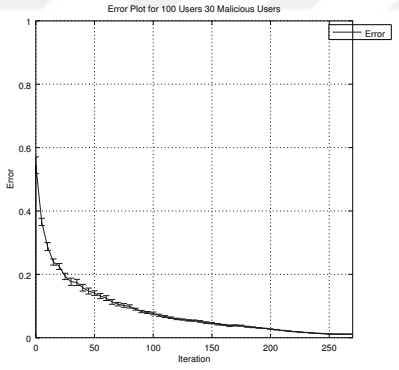

Figure 19: 100 Users and 30 Malicious Users

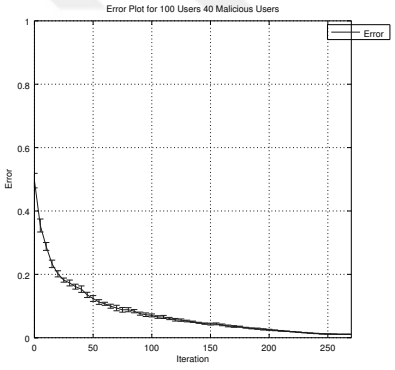

Figure 20: 100 Users and 40 Malicious Users

1. Algorithm successfully isolates malicious users within a reasonable amount of iterations. The error converges to 0. Examples of that can be seen in figures 17, 18, 19, 24 and in 21 for collaborative malicious users and in all figures that represent non-collaborative malicious users (30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41). For collaborative malicious users if we have a low ratio of collaborative

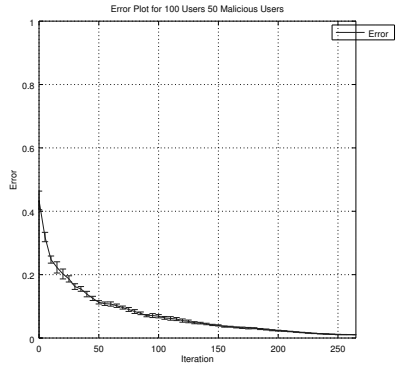

Figure 21: 100 Users and 50 Malicious Users

malicious to real users the system can successfully isolate collaborative malicious users. For non-collaborative malicious users since they do not provide any information about their claimed whereabouts system successfully isolates them even if we have a high ratio of non-collaborative malicious to real users.

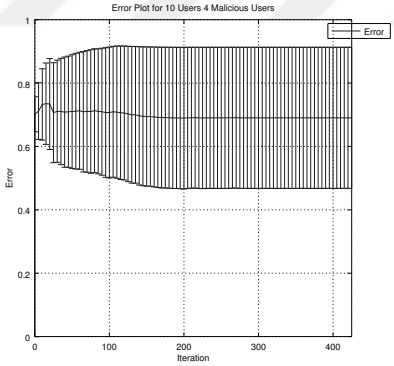

Figure 22: 10 Users and 4 Malicious Users

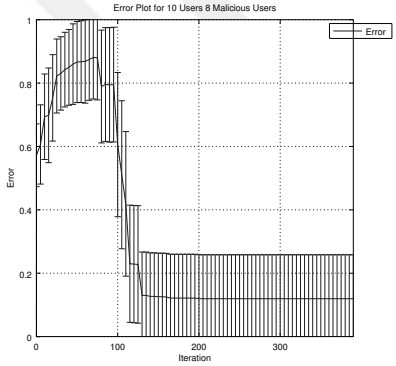

Figure 23: 10 Users and 8 Malicious Users

2. Algorithm fails to distinguish between malicious users and normal users which results in a large standard deviation of the error. Examples of that can be seen in figures 22 and 23. Figure 22 shows a system having only 10 real users and 4 collaborative malicious users which can only result in a very sparse feedback graph thus the algorithm fails to converge to 0 or 1. For figure 23 we can say that the system is still sparse however it is dense enough to provide a better result

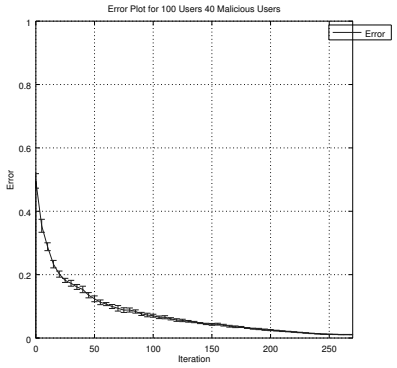

Figure 24: 100 Users and 40 Malicious Users

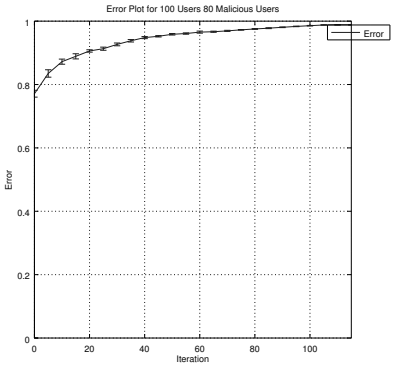

Figure 25: 100 Users and 80 Malicious Users

despite having the large error bars. We can compare these with their denser counter parts figures 24 and 25; it is clear that the same ratio of malicious users can converge to an error if we have a denser feedback graph.

3. Algorithm fails to distinguish real users and the error converges to 1, showing that the malicious users dominated the system and they achieved their goal by compromising the entire system. Examples of that can be seen in figures 25, 26, 27, 28 and 29. It is obvious that when there is a high ratio of collaborative malicious to real users the system is compromised and fails to distinguish between real and malicious users.

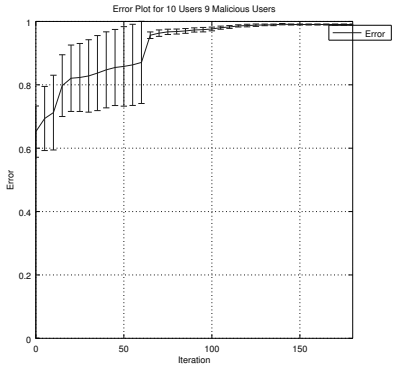

Figure 26: 10 Users and 9 Malicious Users

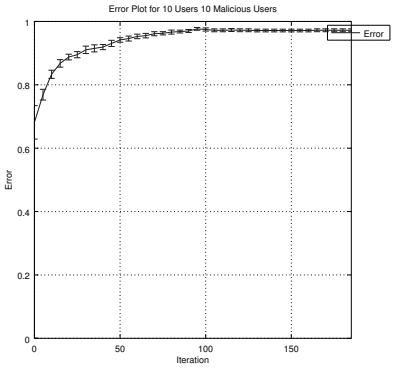

Figure 27: 10 Users and 10 Malicious Users

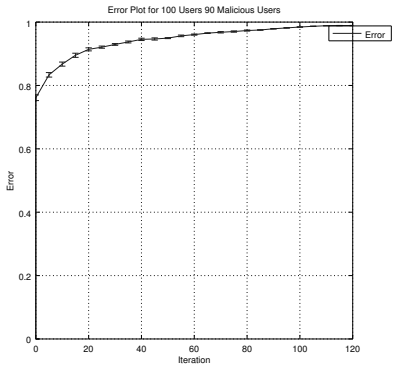

Figure 28: 100 Users and 90 Malicious Users

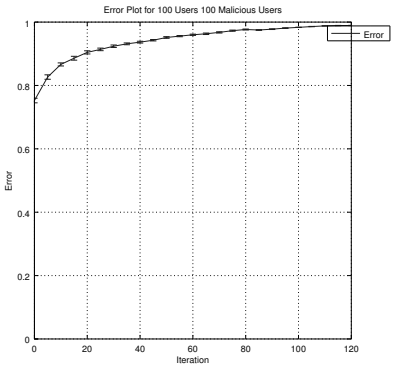

Figure 29: 100 Users and 100 Malicious Users

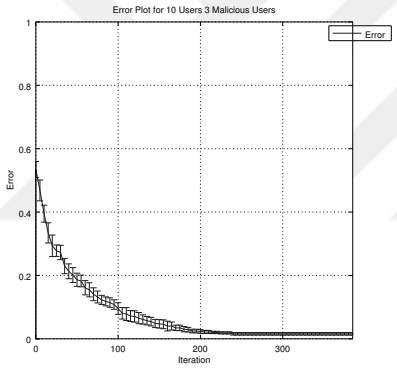

Figure 30: 10 Users and 3 Non-collaborative malicious Users

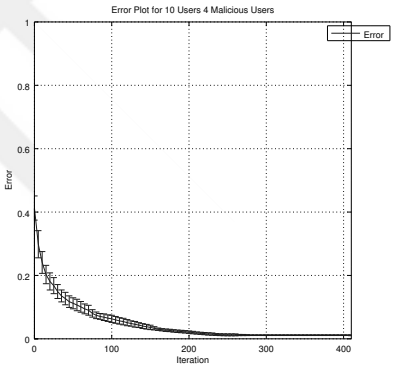

Figure 31: 10 Users and 4 Non-collaborative malicious Users

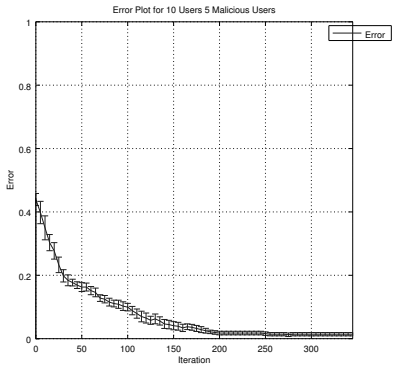

Figure 32: 10 Users and 5 Non-collaborative malicious User

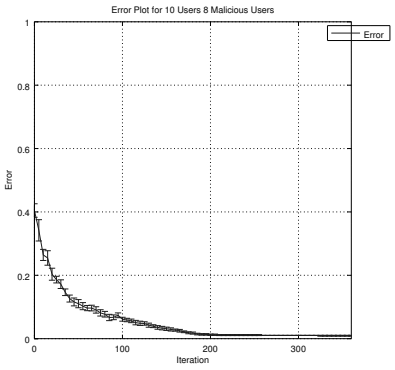

Figure 33: 10 Users and 8 Non-collaborative malicious Users

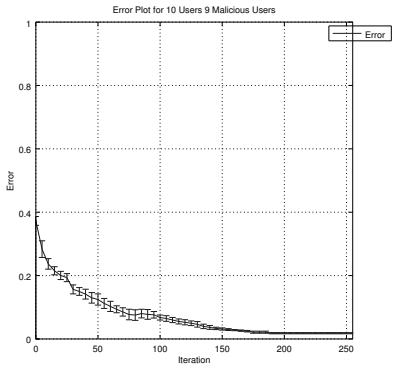

Figure 34: 10 Users and 9 Non-collaborative malicious User

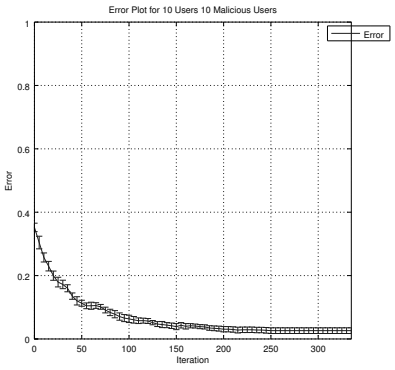

Figure 35: 10 Users and 10 Non-collaborative malicious Users

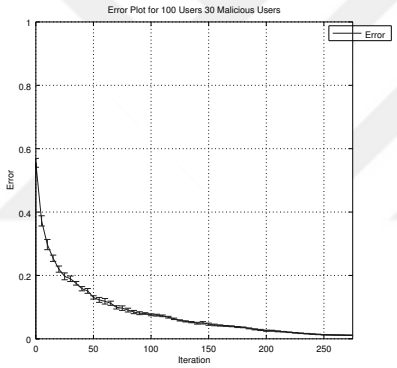

Figure 36: 100 Users and 30 Non-collaborative malicious Users

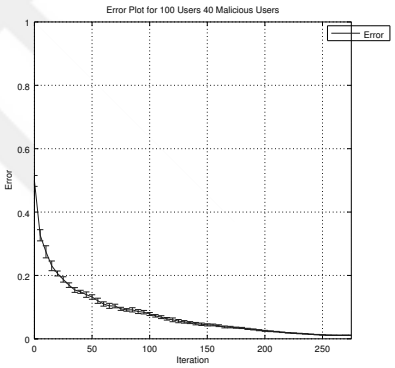

Figure 37: 100 Users and 40 Non-collaborative malicious Users

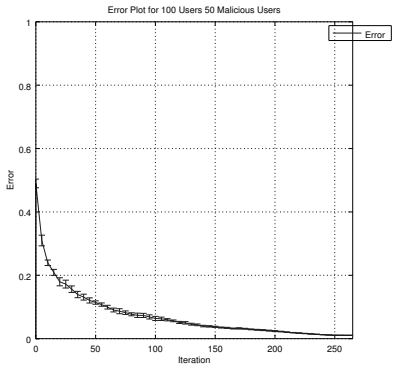

Figure 38: 100 Users and 50 Non-collaborative malicious User

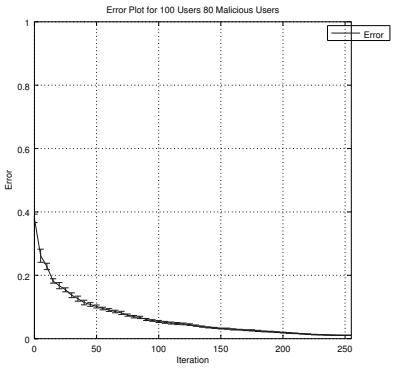

Figure 39: 100 Users and 80 Non-collaborative malicious Users

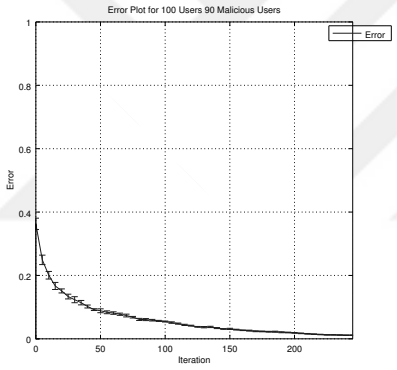

Figure 40: 100 Users and 90 Non-collaborative malicious User

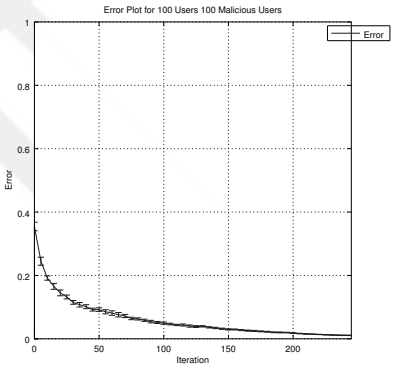

Figure 41: 100 Users and 100 Non-collaborative malicious Users

CHAPTER VI

CONCLUSION

The emergence of touch screen smartphones surely changed how we view information. Increased use of mobile internet made data accessible whenever we desired. We are now highly dependent on our smartphones. Artistic videos surface on Facebook and other social media about the intensive use of smartphones. This shift in our lives created an entire new market. GPS and location sensitive data took its place with this shift. Many of the applications we use in our daily lives depend on our location and many assume that this location information isn't compromised. Google search trends of fake gps and Pokémon GO shows how users can quickly pickup a habit when they desire something.

It is important for an application to secure the data it serves depending on it's user's location. It should do it without the need of location authorities and ways that will compromise user's privacy like mobile phone tracking. Users need to provide their location with their own decisions but the application is responsible to prove that the users are indeed telling the truth. As applications get more and more complicated the need for a geospatial access control language is inevitable. Using a vanilla Bluetooth implementation fails to become a reliable framework due to vendor strategies and user habits. We need an implementation that works on all devices without any compromise.

Using Bonjour allows us to use all devices that support Bonjour, which proves to be a long list of devices, as a proof provider. It is possible for a future work to learn printers, TVs, routers laptops and even servers co-located with the user and use those as a proof to the user in question. The combination of GeoXACML implementation

allows an abstraction and a framework that is built for change a framework that can be molded in to any business requirement that is faced.

Bibliography

[1] Statcounter, "Top operating systems 2016," 2016.

[2] J. Golson, "Apples app store now has over 2 million apps," 2016.

[3] K. Bell, "Google play now has more apps than apple's app store, report says," 2015.

[4] Statista, "Cumulative number of apps downloaded from the google play as of may 2016," 2016.

[5] Statista, "Number of apps available in leading app stores as of june 2016," 2016.

[6] J. Greenberg, "Netflixs vpn ban isnt good for anyoneespecially netflix," 2016.

[7] H. Campbell, "Top 10 ways that uber and lyft passengers are gaming the system (and how to prevent it)," 2014.

[8] R. Jones, "The uber scammers who take users for a (very expensive) ride," 2016.

[9] A. Gjorgievska, "Match investors swipe right on stock as tinder boosts growth," 2016.

[10] J. Hanke, "Apple special event. september 7, 2016.," 2016.

[11] C. Gartenberg, "This pokmon go gps hack is the most impressive yet," 2016.

[12] B. Gilbert, "Pokmon go players are getting refunds from apple and google for in-game purchases," 2016.

[13] S. Arunkumar, B. Soyluoglu, M. Sensoy, M. Srivatsa, and M. Rajarajan, "Geospatial access control for mobile devices," in *Region 10 Symposium (TENSYMP), 2015 IEEE*, pp. 86–89, IEEE, 2015.

[14] K. D. Silva, "Uncovering xacml to solve real world business use cases," 2015.

[15] S. Arunkumar and M. Rajarajan, "Healthcare data access control using xacml for handheld devices," in *Developments in E-systems Engineering (DESE), 2010*, pp. 35–38, IEEE, 2010.

[16] D. of Defense and G. NAVSTAR, "Global positioning system standard positioning service performance standard," 2008.

[17] J. Stables, "The best gps running watches," 2016.

[18] Andev, "Fake gps - fake location," 2008.

[19] S. Wang, J. Min, and B. K. Yi, "Location based services for mobiles: Technologies and standards," in *IEEE international conference on communication (ICC)*, pp. 35–38, 2008.

[20] https://code.google.com/u/115750341222356983405/, "Telephonymanager.getneighboringcellinfo() returns empty list on many samsung devices, including galaxy nexus," 2012.

[21] G. Inc., "Telephonymanager.getneighboringcellinfo() returns empty list on many samsung devices, including galaxy nexus."

[22] M. Spitz, "Your phone company is watching," 2012.

[23] A. Singh, *LOCUS: Wireless LAN location sensing*. PhD thesis, WORCESTER POLYTECHNIC INSTITUTE, 2004.

[24] Alnitak(http://stackoverflow.com/users/6782/alnitak), "How does geographic lookup by ip work?," 2008.

[25] "Tor: Overview."

[26] PokemonGoF(https://github.com/PokemonGoF), "Pokemongo-bot," 2016.

[27] O. Standard, "extensible access control markup language (xacml) version 2.0," 2005.

[28] J. Greenwood and A. Whiteside, "Ogc web services common standard," 2010.

[29] S. Saroiu and A. Wolman, "Enabling new mobile applications with location proofs," in *Proceedings of the 10th workshop on Mobile Computing Systems and Applications*, p. 3, ACM, 2009.

[30] W. Luo and U. Hengartner, "Veriplace: a privacy-aware location proof architecture," in *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 23–32, ACM, 2010.

[31] R. Hasan and R. Burns, "Where have you been? secure location provenance for mobile devices," *arXiv preprint arXiv:1107.1821*, 2011.

[32] R. Hasan, R. Khan, S. Zawoad, and M. M. Haque, "Woral: A witness oriented secure location provenance framework for mobile devices," *IEEE Transactions on Emerging Topics in Computing*, vol. 4, no. 1, pp. 128–141, 2016.

[33] Z. Zhu and G. Cao, "Toward privacy preserving and collusion resistance in a location proof updating system," *IEEE Transactions on Mobile Computing*, vol. 12, no. 1, pp. 51–64, 2013.

[34] X. Wang, J. Zhu, A. Pande, A. Raghuramu, P. Mohapatra, T. Abdelzaher, and R. Ganti, "Stamp: Ad hoc spatial-temporal provenance assurance for mobile users," in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, pp. 1–10, IEEE, 2013.

[35] Y. S. Krishna, V. Subrahmanyam, M. Zubair, and P. Rajalakshmi, "Tensymp 2015,"

[36] S. Cheshire, "Zero configuration networking (zeroconf)."

[37] A. Inc., "About bonjour," 2013.

[38] G. Inc., "Using network service discovery."

[39] A. Inc., "Bonjour for developers."

[40] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *Proceedings of the 12th international conference on World Wide Web*, pp. 640–651, ACM, 2003.

[41] I. Sun Microsystems, "Sun's xacml implementation."

[42] D. H. Steinberg and S. Cheshire, *Zero Configuration Networking: The Definitive Guide: The Definitive Guide*. " O'Reilly Media, Inc.", 2005.

[43] A. Inc., "Discovering and advertising network services."