Enhancing Privacy in Location Based Services

Caroline Holmberg

Abstract- **A great concern of many today is that they no longer have control over their own private data online. As soon as an internet connection is established and the user goes online an exchange of information takes place. This information can then be used for directed commercials or to make a profile of the users habits and interests. The project will implement a new method to prevent this and produce a working app. The focus will be on protecting privacy while using location based services. These services in particular have the opportunity to collect information about a users whereabouts and other sensitive data. To protect user integrity, this project will use a decentralized system involving peer to peer communication. This can also come with a lot of problems since many devices and nodes are involved and some of them might be mischievous. Therefore, user accountability, that is, a users responsibility over the material posted, must be established. This is accomplished by saving all users identities at a long term certification authority. With the ticket the long term certification authority gives them in return the users can then get a pseudonym to apply in contact with other devices and the location based service. The result of the project indicates that this method works well to protect user privacy in contact with location based services.**

I. INTRODUCTION

Many mobile services used today collects information that over time can threaten user privacy. A possible solution is to collect information from neighbors that are interested in the same information instead of going directly to the service. One study into this technique is described in [1]. The goal of this project will be to implement the theory described in said paper and produce a working app using the program Android studio. In short, the theory proposes a decentralized system where the users interact in a peer to peer network and mainly get the information from other users. All identities of the members in the peer to peer network are stored with a long term authentication authority. This authority in turn gives the users an anonymous ticket they can use to apply for a pseudonym at a pseudonymous authentication authority. This pseudonym will then be used to safely and anonymously interact with other users and the location based server. It is necessary to store user identity to achieve non repudiation and prevent misbehaving peers. The detection and removal of misbehaving peers will however not be implemented in this project. The architecture is visualized in figure 1.

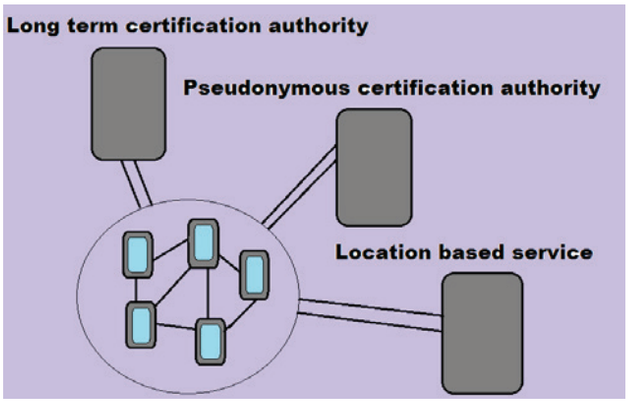

Figure 1. system structure

None of the components in the system will be able to track a users activity. The long term certification authority does not know which pseudonym the user has been given and the pseudonymous certification authority does not know the identity of the users. When a request for information is made, the cache of the users' own device will be checked. If the requested information is not found the question will go out to neighboring users. If they do not have the information the next step will be to go to the location based service. When the requested information has been gained it will be saved in the cache memory for later use. Due to the time restraints some adjustments will be made. The focus will be on creating a secure connection between to peers and exchange points if interest between them.

II. SECURITY REQUIREMENTS

The security requirements that needs to be taken into account during this project are:

II.A. Authentication and confidentiality

Authentication means that it is possible to verify that the information comes from a reliable source. This requirement is met by the use of a long term authentication authority. The user remains anonymous in contact with other devices and servers but since the information is recorded by the long term authentication authority it is still possible to verify that the user is a legitimate participant in the network. Confidentiality, which means that only the intended receivers can read the messages is achieved since only users registered

by the long term authentication authority can partake in the information exchange.

II.B. Non-repudiation and accountability

These basically interprets as a proof of origin of the data. The users can secure that the device they are interacting with is indeed a verified member of the network. It must be possible to trace the sender and this person will be held responsible for any mischievous activity from their device. This can be done by using the information saved by the long term authentication authority. As mention before though, this project will not include any punishments for such nodes.

II.C. Anonymity and unlinkability

The users' identity will not be traceable by the location based server or the peers in the peer to peer network. This is accomplished by the use of pseudonyms and anonymous tickets.

III. TOOLS AND PRELIMINARIES

III.A. Android studio

Android studio is the platform on which the project is going to be done [2]. It is the official developing environment for Android apps. Here it is possible to both build the app and simulate it on virtual devices. The language used in the program is java.

III.B. Peer to peer communication

This is a method that allows nearby devices to communicate with each other. Using this, the user hardly ever has to interact with the service that would otherwise have collected their information. This also has the advantage that even if a few nodes malfunction the network as a whole will not be effected. Peer to peer communication can be established in a number of ways. In this project unicast communication will be used. This means that there will be one sender and one receiver. Only two devices will interact with each other. When applied in real life multicast communication would be used. It means that there are one or more senders and multiple receivers. One device can thus interact with many others. The communication part will be achieved by using secure socket layer(SSL) sockets, thus creating a secure connection.

III.C. SSL sockets

A socket is the endpoint of a two-way communication link. It is specified by an IP-address and a port number. Every connection is uniquely identified by their end points; thus it is possible to have multiple connections at once. Sockets are used to enable two apps on different devices to communicate with each other [3].

SSL is a security protocol that establishes a secure link between devices or servers. A protocol is a set of rules that regulates communication between different network devices. Thus it is not necessary to specify exactly what has to be done with the keys and certificates provided. They just have to be uploaded and then the programming necessary to use them is already contained in the SSL protocol. To implement this socket, OpenSSL and spongy castle is used. Spongy castle is a cryptographic library and OpenSSL a software library. OpenSSL is the program used to create the keys and certificates which is then used to create the SSL socket.

III.D. Keys

A key is basically a sequence of numbers and letters. They are used for authentication and encryption. In this project RSA keys will be used. RSA is made up of the initial letters of the inventors surnames, Rivest, Shamir and Adleman [4]. Each user has a private and a public key. In RSA both keys can be used to encrypt. Everyone can see the public key while the private key is visible only to the user. When sending a message, the user can sign with the private key and then anyone can decipher it with the users' public key. Thus, anything signed with the private key can be deciphered with the public key and vice versa. But it has to be the same key pair. User A can for example not use their public key to decipher anything signed with user Bs' private key. Only As' public key can be used to decipher A's private key.

To safely send a message two properties have to be fulfilled. Confidentiality, only the intended receiver can read the message. Authentication, ensures that the message comes from a reliable source. This does not necessarily mean that the identity of the sender is known but it confirms that the sender is the owner of the public key that they use. Combined with certificates, which contain the public key and a signature from a reliable authentication source, confirms that the person is a trusted participant of the network. To achieve confidentiality, the sender encrypts the message with the intended receiver's public key. Thus, only that receiver can decrypt the message using their private key. This is shown in figure 2.

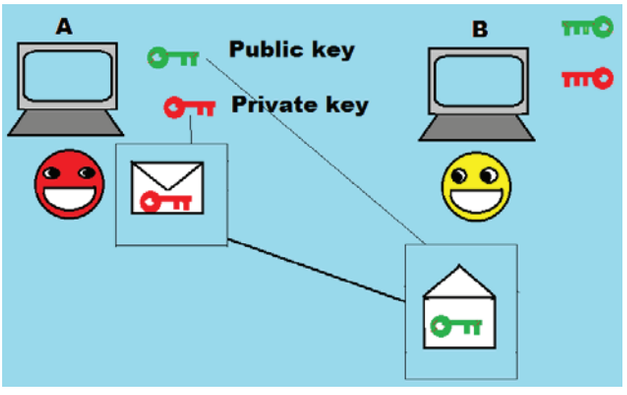

Figure 2. To achieve authentication, A signs with her private key

To achieve authentication, the user signs the message with their private key. Only doing this though creates a problem since everyone can decrypt it using the senders public key. To solve this problem, the user first encrypts the message with their private key, and then with the intended receivers public key. Then, the receiver can use their private key, followed by the senders' public key to decrypt the message. This is shown in figure 3.

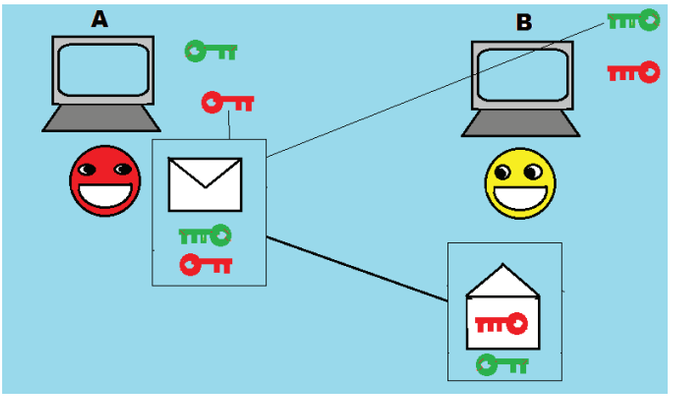

Figure 3. To achieve authentication and confidentiality, A encrypts with her private key and Bs' public key

E. Key store

This is a storage for cryptographic keys and certificates. There are three types of entries in a key store. The private and public key pair, a secret key that will prevent unauthorized access and a public key certificate that can be used to authenticate other parties [7].

F. Certificates

Certificates works as on online ID-card. They make sure that the users or servers in a connection are certified by whichever authority that signed it. The certificate contains the public key and a signature from the authentication agency. Thus, by confirming authentication through the use of keys the ownership of the certificate is also confirmed.

III.G. Google API

API stands for application-programming interface. It is a collection of programming instructions that makes it easier to access a web based service. Developers can use this particular API to add maps to web applications or apps [5].

III.H. Cryptographic libraries

A cryptographic library is a collection of algorithms and APIs used to secure systems and create keys. An algorithm is a set of steps for a computer program to accomplish a task. The library used in this project is Spongy castle [6].

III.I. Centralized systems

Centralized systems mean that most of the computing is done at the same place. To protect privacy in a system like this an anonymizer is used. This is a device that mixes up queries from the clients before sending it to the server, thus stopping the server from acquiring private information.

III.J. Decentralized systems

Decentralized systems mean that most of the computing is done at a different place before it will be sent to a central server. Sometimes, like in the case of peer to peer communication, a central server might not at all be necessary. This almost completely cut out the risk of a central server collecting information but will instead introduce the risk of losing information to misbehaving peers.

IV. IMPLEMENTATION

The app was made in Android studio. When first starting a project it is possible to choose a number of base apps. The base app chosen was one that uses maps. The default setting produces a picture of a map with a marker set in Sidney, Australia. To be able to use this map it was necessary to get a key from the Google API. This key is then added to the Android manifest. When this is done it was necessary to log in to a google account. Now the map is ready to be used. A zoom function was added to make it easier to do close ups. The next step was to create a text file with the type, coordinates, name and addresses of different places that a user could be interested in. This file was then placed on the SD card of one device and a file with slightly different content was placed on the SD card of the other device. The format was as follows:

Cinema, 59.33466639999999, 18.062829899999997, Sergel bio, Hötorget

Restaurant, 59.3353515, 18.06237669999996, Kungshallen, Kungsgatan 44

The next function added was a window that would pop up when the user long clicked the map. In this window, the user can write what type of place the program should search for as seen in figure 4, for example, restaurant. It was necessary to specify that whether a letter was capitalized or not would not make a difference. Otherwise the program would have returned a result when searching for Restaurant while it would have found nothing searching for restaurant.

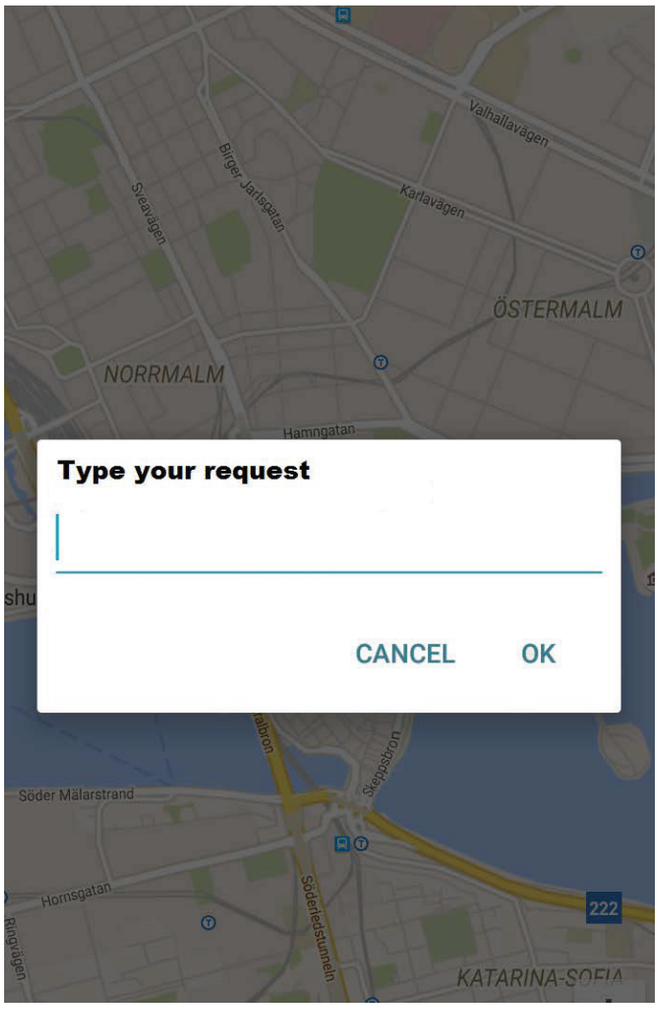

Figure 4. Popup that accepts client requests

The program then goes through the first column, type, in the list of places in the file and if the type matched restaurant, it would pick it out. Then the program would check if the found restaurant is within one kilometer radius. If it is, the place would be showed on the map with a marker as seen in figure 5. These steps would then be repeated with all the items in the list.

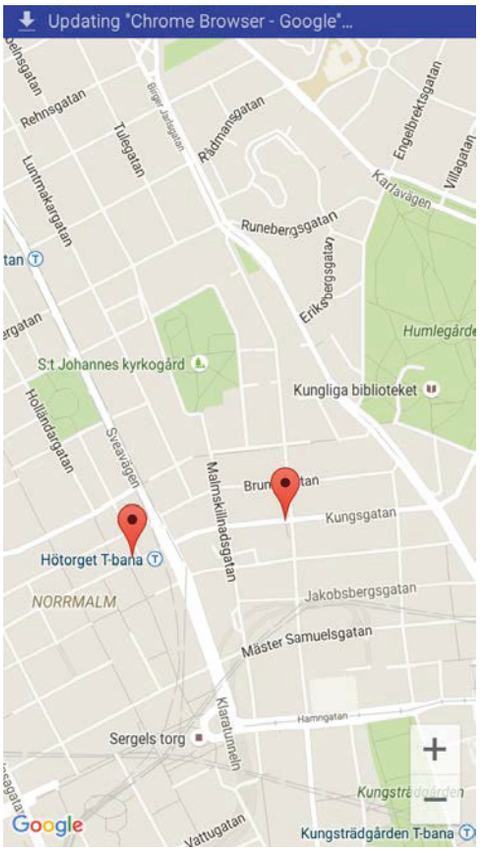

Figure 5. Markers shows the location of nearby restaurants

The next part is to make two devices communicate with each other to receive information the users' device does not already have. This is implemented using SSL sockets. These sockets are used in the project because they provide a secure transmission of messages. The transmission takes place in parts, beginning with a call for socket. If one device calls another, what will happen on the receiver side is bind, listen and accept. Bind establishes an address for the communication in point and listen and accept prepares the socket for communication. Then the calls send and receive will be used for as long as the communication takes place. Both sides will end with close [8]. Code wise the creation of a SSL socket must be done in two parts. A receiving socket and a sending socket. Then the functions these sockets will perform must be specified. In this case, the sending socket will take the users' input and send it to the other device as a stream. The receiving socket will take this stream and turn it back into a string. Then it will check the points of interest file in the SD memory. If a result is found it will check that it is located in the desired range. If so, it will convert the result into a stream and send it back to the device that made the request. To be able to use sockets it is necessary to create a key store and a trust store. The key store contains the public and the private key and the trust store contains the certificate.

This is done by using OpenSSL. To create the keys, the following code is used [9]:

openssl genrsa -des3 -out Alice_key.pem 2048

openssl genrsa -des3 -out Bob_key.pem 2048

This creates two keys; Alice_key and Bob_key. Secondly, the certificates need to be created. This is done with the code:

openssl req -new -x509 -key Alice_key.pem -out client.pem -days 365

openssl req -new -x509 -key Bob_key.pem -out server.pem -days 365

These certificates are created in Pem.format and valid for 365 days. To be able to use the keys and certificates it is necessary to create a truststore. This will contain the certificates the device should trust. So Alices' certificate needs to be in Bobs' truststore and Bobs' certificate needs to be in Alices' truststore in order for their devices to be able to interact with each other. This code creates Alices truststore and imports Bobs certificate into it:

keytool –importcert -trustcacerts –keystore Alicetruststore.bks –storetype bks –storepass <truststore_password> -file Bob.pem -provider org.spongyycastle.jce.provider.SpongyCastleProvider –providerpath <path_to_bcprov_jar>

Similarly, this code creates Bobs truststore and imports Alices certificate into it:

keytool –importcert -trustcacerts –keystore Bobtruststore.jks –storetype jks –storepass <server_truststore_password> -file Alice.pem

Since Javas keytool doesn't allow an already created key to be imported into a keystore it is necessary to combine the private key and the certificate. This is done with the following code:

openssl pkcs12 –export –inkey Alice_key.pem –in Alice.pem –out Alice.p12

openssl pkcs12 –export –inkey Bob_key.pem –in Bob.pem –out Bob.p12

This can then be put into keystores and imported.

keytool –importkeystore –srckeystore Alice.p12 –srcstoretype pkcs12 –destkeystore Alicekeystore.bks –deststoretype bks –provider org.spongyycastle.jce.provider.SpongyCastleProvider –providerpath <path_to_bcprov_jar>

keytool –importkeystore –srckeystore Bob.p12 –srcstoretype pkcs12 –destkeystore Bobkeystore.jks –deststoretype jks

The final files that are imported are, Alicetruststore, Alicekeystore, Bobtruststore and Bobkeystore. The stores are then placed onto the devices SD card among the text file with points of interest. In this project the certificates are self-signed. Since a certificate is used to authenticate that the users are legitimate participants of the peer to peer conversation it will not pose a security threat. The security provider is Spongy castle. Therefore, it is necessary to download spongy castles' core and prov jars. These will then be placed into library. The other library needed is CVS reader that is used to read the lines in the points of interest file. The full code for the project is placed in the Appendix.

V.FUTURE WORK

There are still a lot of work that has to be done in order to fully implement the theory described in the beginning. Instead of using a file with points of interest the devices should interact with a real location based server. The program should also receive the certificates from certification authorities instead of using the self-created ones. These certificate authorities are already available. A few more changes that should me made are to have the results gained from peers or the location based server saved onto the file.

VI. CONCLUSIONS

The goal of this project was to implement an app which protects user privacy while using location-based services. This has in parts been reached. Due to time restraints certificates have been self-created and there have only been two phones involved. The security aspects have on the other hand been met. Authentication was accomplished by using the self-created certificates and keys. By showing the certificate to the other device the user proves that they are who they claim to be and that they are legitimate members of the peer to peer network. Confidentiality is also achieved since only this particular certificate will be valid in the information exchange. Non-repudiation and accountability was achieved by using the private key to sign messages. Anonymity and unlinkability was however not achieved since no pseudonymous certification authority was involved.

VII. ACKNOWLEDGEMENTS

I would like to thank Hongyu Jin and Mohammad Khodaei for their extensive help and support throughout the project.

VIII. REFERENCES

[1] H. Jin and P. Papadimitratos. “Resilient Collaborative Privacy for Location-Based Services,” Springer Secure IT Systems, *NordSec conference*, pp. 47-63, October 2015

[2] "Download Android Studio and SDK Tools | Android Developers," May, 2016. [Online]. Available: http://developer.android.com/sdk/index.html.

[3] ”What is a socket?,” May, 2016. [Online]. Available: https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html.

[4] M. Rouse.”RSA algorithm,” May, 2016. [Online]. Available: http://searchsecurity.techtarget.com/definition/RSA

[5] ”Google maps API,” May, 2016. [Online]. Available: https://developers.google.com/maps/.

[6] ”Spongy castle,” May, 2016. [Online]. Available: https://rtyley.github.io/spongycastle/

[7] ”Class keystore,” May, 2016. [Online]. Available: https://docs.oracle.com/javase/7/docs/api/java/security/KeyStore.html.

[8] M. Rouse.”Network socket,” May, 2016. [Online].Available: http://whatis.techtarget.com/definition/sockets

[9] M.Krantz. “Creating self-signed certificates for use on Android,” May, 2016. [Online]. Available: http://callistaenterprise.se/blogg/teknik/2011/11/24/creating-self-signed-certificates-for-use-on-android/