UNIVERSITY OF GOTHENBURG

Constructive Research Study of Location-Proximity Privacy-Preserving Algorithm Capabilities as Part of a Mobile Application

Bachelor of Science Thesis Software Engineering and Management

SIMONAS STIRBYS
OMAR ABU NABAH

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, June 2016

Constructive Research Study of Location-Proximity Privacy-Preserving Algorithm Capabilities as Part of a Mobile Application

SIMONAS STIRBYS
OMAR ABU NABAH

Examiner: MIROSLAW STARON

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2016

Constructive Research Study of Location-Proximity Privacy-Preserving Algorithm Capabilities as Part of a Mobile Application

Simonas Stirbys
Department of computer Science and Engineering
Gothenburg University
Gothenburg, Sweden
simonas.stibyrs@yahoo.com

Omar Abu Nabah
Department of computer Science and Engineering
Gothenburg University
Gothenburg, Sweden
gusabunom@student.gu.se

***Abstract*—Location Based Services (LBS) have been responsible for several privacy breaches of their users in recent years and a number of researchers have put forward solutions towards making LBS more secure in regards to user location privacy. However, the proposed protocols were rarely tested in real conditions. This research study will create an artifact, an Android application that implements one of the state-of-the-art protocols for location privacy preservation, and test its performance to determine if the state of the protocols is at a stage where it can be implemented in real world applications. We have found that the performance of our artifact in short distances is comparable to real applications as it took 5 seconds to check proximity between users in 25 meter radius. However, the computation time increases drastically with increase in radius values and the protocol took 72 seconds to check proximity for 100 meter radius. Such time frames are insufficient as we believe most users are not willing to wait so long for an application to process their requests. Therefore we were unable to show that the state-of-the-art location privacy-preserving algorithms are ready for adaptation in real world applications, although certain modifications to coordinate precision could correct that.**

I. INTRODUCTION

Through using LBS in mobile applications users are able to accomplish a large variety of different tasks, such as planning a route from one location to another or obtaining information about entertainment venues in the vicinity. Applications such as these require the location of their user to provide functionality and are categorized as Location Based Services (LBS). By obtaining the location of their users these LBS applications are able to provide a personalized experience to their users, but unfortunately, at the same time endanger the confidentiality of the user's location. As a result, numerous solutions aiming to improve the security of mobile LBS applications have been created. This study aims to evaluate the effectiveness of these solutions by building a mobile application that incorporates them. The chosen algorithms will then be evaluated in terms of efficiency and applicability. The aim of the study is to determine whether privacy preserving algorithms at their current state are sufficient enough for use in real mobile applications on the market.

When choosing a privacy preserving methodology, we have a number of options. Zhu et al [1] proposed a k-anonymity protocol which bypasses Trusted Third Parties (TTP) with the help of homomorphic encryption. The resulting protocol kept user data private from dishonest third parties and location privacy attackers. On the other hand, a solution by Wu et al [2] utilizing the k-anonymity model made it more difficult to pinpoint the location of the real user for possible malicious 3rd parties. However, if the amount of querying users is too small, the conditions may be not met for the algorithm to function properly. This is a notable disadvantage of the algorithm, as like with many things, upon first implementation the user pool is very limited.

As we can see, while various solutions can protect user privacy in theory, they may have flaws that prevent them from functioning properly when implemented. Our first goal therefore is to evaluate the efficiency of the latest privacy preserving algorithms as part of a mobile application in order to determine if the state of current technology is efficient enough for use in real market applications. Our second goal is to determine if the flexibility of the algorithms in question is wide enough that they could be adapted to other mobile applications without limiting their functionality. This leads us to the following research questions:

1. How efficient are the implementations of state-of-the-art privacy-preserving location-proximity algorithms when integrated into a mobile application?

1.1 How do they compare to other Location Based Service mobile applications in regards to efficiency?

2. To what extent can such algorithms be applied to mobile applications without limiting their functionality?

To answer these research questions we chose the constructive research methodology. This method requires us to develop an artifact and measure its performance. In our case, the artifact is a mobile application that incorporates a state-of-the-art algorithm. Data collection will consist of both quantitative and qualitative methods. Quantitative method will be used to measure the efficiency of the algorithm and it will be performed by measuring the response time of our application in seconds. Qualitative method will determine the applicability of the algorithm by examining real mobile applications and determining to what extent could they incorporate the algorithm while maintaining the same function. Data analysis will be of qualitative nature.

II. Related Work

A. LOCATION PRIVACY

Location-privacy is a type of privacy that, as the name suggests, is concerned in keeping the location of the user confidential. Research shows mixed results on whether location-privacy is important to users of LBS or not. Barkhuus and Dey [3] paper compared two scenarios: location-tracking services vs. location-aware services. They have conducted an experimental case study on 16 participants where location based services were hypothetical. The researchers found that the participants had more privacy concerns regarding location tracking services compared to location-aware services, but in general were not overly concerned about mobile applications of either type obtaining their location data. Nevertheless, Barkhuus and Dey recommended focusing on developing services around location-aware concept. In case of location-tracking services, the researchers believe such services can still be acceptable as long as users have the option to turn-off the tracking capability at any time.

Xu and Gupta [4] developed a model to examine the impact of privacy concerns on intention to use Location Based Services (LBS). The model incorporated elements from the Unified Theory of Acceptance and Use of Technology (UTAUT). The model studies four elements which are: Privacy Concerns, Effort Expectancy, Performance Expectancy (accuracy of LBS) and Personal Innovativeness and their impact on intention to use LBS. The researchers found that performance expectancy had a positive impact on participants' intention to use LBS and effort expectancy had a positive impact only for inexperienced users, but privacy concerns had no direct effect. However, interestingly privacy concerns negatively impact performance and effort expectancy of the participants, thus indirectly affecting user decision to use LBS mobile services. This implies that privacy concerns are relevant to at least a limited extent to user of LBS applications.

Zickuhr [5] conducted a survey to examine the level of mobile LBS use by Americans. The findings show that the use of such applications is rapidly growing, from 55% of smartphone owners using LBS application in 2011 to 74% of smartphone owners using LBS in 2012. Furthermore, taking into account that smartphone ownership itself quickly grew from 35% of adults in 2011 to 46% in 2012, it is safe to assume that the importance of LBS privacy concerns, although somewhat irrelevant now, will grow in the coming years.

B. METHODS FOR PRESERVING PRIVACY

When choosing a method for preserving location-privacy, we are presented with a wide array of choices. One popular strategy is a k-anonymity model, which masks the user sending his data (in this case, location) together with similar real data of other users to the server. This makes the original user indistinguishable from the rest of the population and thus anonymous. Wu et al [2] created a solution based on k-anonimity model which implements "split cloaking to generate a set of distributed cloaking regions." These regions provide a higher volume of querying users to the client to distract the client, thus making it harder to pinpoint the location of the real user. However, if the amount of querying users is too small, the conditions may be not met for the algorithm to function properly. This is a notable disadvantage of the algorithm, as like with many things, upon first implementation the user pool is very limited. Another k-anonymity solution was proposed by Gedik and Liu [6] with the aim to hinder location and time sampling of users by LBS providers. The solution requires a trusted server which would act as an intermediary between users and LBS providers. The server would get data from users, remove any identifying information (such as an IP address) and perturb location and time data before sending it out to LBS providers. What is unique about Gedik and Liu's proposal is that unlike previous k-anonimity models, the solution allows users to personalize their level of anonymity and the extent of location and time data perturbation. This is advantageous as anonimyzing and perturbing data is costly, but by allowing users to reduce the extent of it to the level that they need allows the process to be faster. However, as researchers noted, it is possible the levels of anonymization and data perturbation set by users may cause conflict with the quality of service offered by the application, which may lead to failed high rate of failed anonimyzation, which is a considerable issue for a proposal attempting to protect the privacy of users.

Another method of approach is in utilizing homomorphic encryption. Originally proposed by Rivest et al [7], homomorphic encryption makes it possible to perform mathematical operation on data possible in its encrypted form. Different types of homomorphic encryption exist, which allow to perform different operations. For example, additive homomorphism allows performing additions. Improvements by Gentry [8] increased homomorphic encryption ability to modify encrypted data, by creating a single scheme which was capable of performing both addition and multiplication as well as improving its efficiency. The ability to perform such operations on encrypted data allow it to remain confidential while being handled by 3rd parties which makes a Trusted Third Party (TTP) no longer necessary for preserving privacy. In recent years the popularity of homomorphic encryption increased with several researchers utilizing it in their privacy-preservation protocols. Zhu et al [1] combines a k-anonymity protocol and homomorphic encryption in their solution. The resulting protocol avoids bottlenecks that are possible when using traditional k-anonymity model with TTP, while at the same time keeping user data private from dishonest third parties and location privacy attackers. Solution by Hallgren et al [9], called InnerCircle, also circumvents the need for TTP by sending homomorphically encrypted coordinates of their users. The encrypted coordinates of the two users are then compared by using additive homomorphic encryption and a result is returned to the requesting users, informing them if the target user is within a certain range or not. As encryption and decryption of data are costly in regards to processing power (especially on a mobile device), using homomorphic encryption to preserve privacy of mobile LBS raises concerns. However, the use of paralellizability and other optimization techniques should increase the efficiency of InnerCircle when encrypting or decrypting the results.

Generating dummy data for LBS providers is also an option when preserving location-privacy. One such solution was made by Zhou et al. [10], who's proposal concerns privacy in a more general sense, but can also be applied to location-privacy. The researchers claim that current Android OS permission system is inflexible as it does not allow users to accept some but not all permission requirements when installing apps, and that it also does not allow to change these settings after the installation. To remedy the situation Zhou et al. propose a system called TISSA. TISSA allows users to choose what data an application may get access to at any time after installation. In case an application demands access to data that the user is unwilling to provide, the system send dummy data as substitute, keeping the real data private. The system was tested in Android OS and successfully prevented leakage of information to restricted applications and caused no significant slow down to performance of the phone. However, sending only dummy data may be problematic as the application may require real data to function properly. In such cases where genuine data is essential for the proper function of a mobile application but the user wants to keep their data private, solutions that send real data and mask it with dummy locations may be preferable. Such solution has been created by Kido et al. [11], who propose a system which simultaneously sends mobile LBS providers real user data as well as dummy data. As the LBS providers cannot distinguish which data is real and which is fake, the anonymity of the user is preserved. The solution by Kido et al. also implements an algorithm for the generation of dummies in order to make them more realistic. This makes it harder for LBS providers to pinpoint the genuine user through the simple method of discarding obviously fake data. Pinpointing becomes further complicated with multiple different users sending data to the LBS providers. However, the drawback of the solution is that it causes communication delays with large number of users using the service simultaneously. As the end goal of any solution is to be widely adapted and highly used, this causes concern.

Puttaswamy et al [12] presented their own new technique to secure location privacy which they called LocX. This technique provides the client server with an encrypted data of the user's location which can only be decrypted using secret keys. These secret keys are restricted to the social circle of the user only and no 3rd party can decrypt the location details. A prototype has been developed and it showed that it can be used in commercial applications with minimum overhead. However, unlike other algorithms mentioned in this section, the user's exact location is revealed to the person with the secret key, which forces the users to limit their social circle to people they trust with their location or risk losing confidentiality of their location to untrustworthy people. Another concern is keeping the secret keys secret: similar to regular passwords, the keys can be leaked outside of trusted parties, in which in case it can cause damage to the privacy of the users.

III. System Overview

A. *METHOD OF PRIVCY PRESERVATION*

The mobile application will use the InnerCircle algorithm [9]. This algorithm was chosen because of its use of homomorphic encryption, which is one of the safest methods to preserve data confidentiality. Homomorphic encryption avoids the need for TTP that regular k-anonymity models require, and due to LBS providers only having access to encrypted data, triangulation of user's location is not possible as it is with methods utilizing dummy data approach. Decryption of coordinates is relatively power demanding which raises concerns in regards to the efficiency of the InnerCircle protocol. However, optimization techniques such as multithreading should reduce the negative effects on execution time of the artifact, albeit at the cost of higher battery consumption. In recent years homomorphic encryption became a popular choice in privacy preservation protocols and as such, it is a good representation of the state-of-the-art technology in location-privacy preservation.

In InnerCircle algorithm, several cryptosystems where used but in this research we have chosen to implement ElGamal's [14] encryption system using 1024 bit keys since it had a notably fast performance in the original implementation. When Alice sends a location-proximity request to Bob , Alice constructs a location request. The location request contains Alice's public key, 3 ciphertexts which will be used by Bob and the radius in which Bob's proximity is to be checked. When Bob receives the request, he will use the 3 ciphertexts and his own coordinates to calculate the encrypted distance and then compute the proximity result using the method lessThan(). Bob then sends the proximity results which is a shuffled list encrypted under Alice's public key. The list shuffling is needed so Alice does not know at which range Bob was found. Last, Alice receives the result and decrypts each ciphertext in the list using the method inProx() to check if it contains a zero which means Bob is in range.

B. *SYSTEM ARCHITECTURE*

The mobile application that will serve as the artifact for this constructive research study will be done for an Android OS. Android OS was chosen because it has the most established development environment for mobile application developers when compared to other smartphones. The abundance of resources for programming Android applications means the researchers have the highest chances of successfully developing a mobile application for this OS rather than other smartphones.

The architecture of the developed mobile system is of Client-Server type. The system involves communication between two clients: client Alice, which is the user checking the proximity, and client Bob, which is the user who's proximity is being checked. The system also employs two servers: Application Server (deployed on Amazon AWS) and Google Cloud Messaging Server (GCM). GCM Server allows applications to push messages to Android devices without any input from the receiver. This is convenient as it allows us to avoid sending undefined amount of requests from mobile devices to Application Server to check if any messages are present. However, currently GCM Servers limit message size to 4 kilobytes, which is not enough to transfer an encrypted response from Bob. Therefore, Application Server is responsible for sending messages between the clients while the GCM server is used only for notifying user mobile devices that a message is ready to be retrieved from Application Server. This allows us to transfer messages larger than 4 kb without indefinitely checking if a message is available on the server.

A single communication request can be seen in Figure 1. It starts when Alice chooses to check Bob's proximity. Alice's phone generates a location request which is sent from mobile phone to Application Server (1), then from Application Server to GCM Server (2), and from GCM Server to Bob's mobile device (3). Bob then creates a proximity result using the method lessThan() to return a list (4) which is sent directly to the Server (5). The server notifies GCM server (6), which in turn notifies Alice that the answer is ready to be retrieved (7). Alice's mobile device then retrieves the answer by sending an answer request to the Application Server (8). When Alice retrieves the answer, it is decrypted using the method inProx() which will return true if Bob is in range (9), completing the communication request.

As InnerCircle assumes Euclidean plane, the longitude and latitude of both users was converted to ”world” coordinates using Google Maps API implementation. The researchers had to implement a java version of the conversion method since the original version was done in JavaScript.

IV. METHODOLOGY

A. RESEARCH SETTING

Our chosen research method is a Constructive Research study. For this study we will develop a mobile application that utilizes a state-of-the-art location-proximity algorithm intended for preserving the privacy of the users. The performance of the artifact will be measured and the results will be analyzed.

The research process will start with the development of the artifact. When the initial application is complete, collection of data will begin. Optimization of the application will also be undertaken simultaneously to data collection, as information regarding the efficiency of the algorithm may reveal unsatisfactory performance which can possibly be remedied. Once the data collection process is completed, the researchers will discuss and analyze the results, before finally concluding their findings.

B. DATA COLLECTION AND ANALYSIS

When determining the efficiency of the algorithm in the application, we will measure the Total Time of response of the artifact and CPU usage in milliseconds 10 times for ranges of 25, 50, 75 and 100 meters. The tests will be performed on two different smartphone devices of different advancement level, which should allow the researchers to make educated assumptions on whether it is reasonable to expect the InnerCircle protocol to be more efficient on future devices as well. Each device will act as Alice and Bob to measure how the phones handle both answer generation and decryption of the messages. The devices used in testing the artifact are:

- Samsung Galaxy S4 Mini: this smartphone model was first released in July 2013 and utilizes 1.7 GHz dual-core Krait 300 CPU processor.
- Nexus 5X: first released in October 2015, this device uses a 1.8 GHz hexa core 64-bit ARMv8-A CPU processor.

When measuring the efficiency, Total Time represents the amount of time it took in milliseconds for the application to display the results to the user. It starts when the user opts to locate a target by pressing the locate button, and ends when the answer is available on the user's phone. It is displayed in the user's screen together with the target's in range result. In case of multiple targets, the total time is calculated for each target separately, starting at the same time for all targets and ending for each target individually when their result is available on the user's phone.

The artifact's CPU time is measured in milliseconds by using method profiling feature of Android Device Monitor tool, which is part of the Android Studio IDE. Monitoring of smartphone devices began before a request was sent and ended once the results were available to the requesting user. Monitoring was restarted after each trial to reset the recorded CPU time by the monitoring tool. The methods of which CPU time is measured are LessThan() and inProx(). The reason these methods are the ones measured is because they are responsible for implementing the InnerCircle protocol. Encryption of Alice's coordinates is not taken into account as it only requires several milliseconds and is irrelevant to the efficiency of the artifact as a whole.

The efficiency of real applications will be measured by using a stopwatch and recording the time from the moment user opts to check location proximity to the moment the application displays its results to the user. The data will be compared to the artifact's total time and CPU time usage, which will allow us to determine if the efficiency of the algorithm is sufficient enough to be successful in the market of mobile applications.

To determine the applicability of the application, we will examine other LBS mobile applications and determine qualitatively if their functions can be supported by the algorithm. To describe the applicability of the algorithm for each application we will use the following three categories:

- Not Applicable - this is the case where the algorithm is too limited to support the features of the mobile application while keeping its functionality in tact.
- Limitedly Applicable - in this case, to incorporate the algorithm the mobile application would require to make minor changes to its features but would still be able to maintain its purpose for the most part.
- Fully Applicable - the algorithm can be incorporated into the mobile application without any changes in the

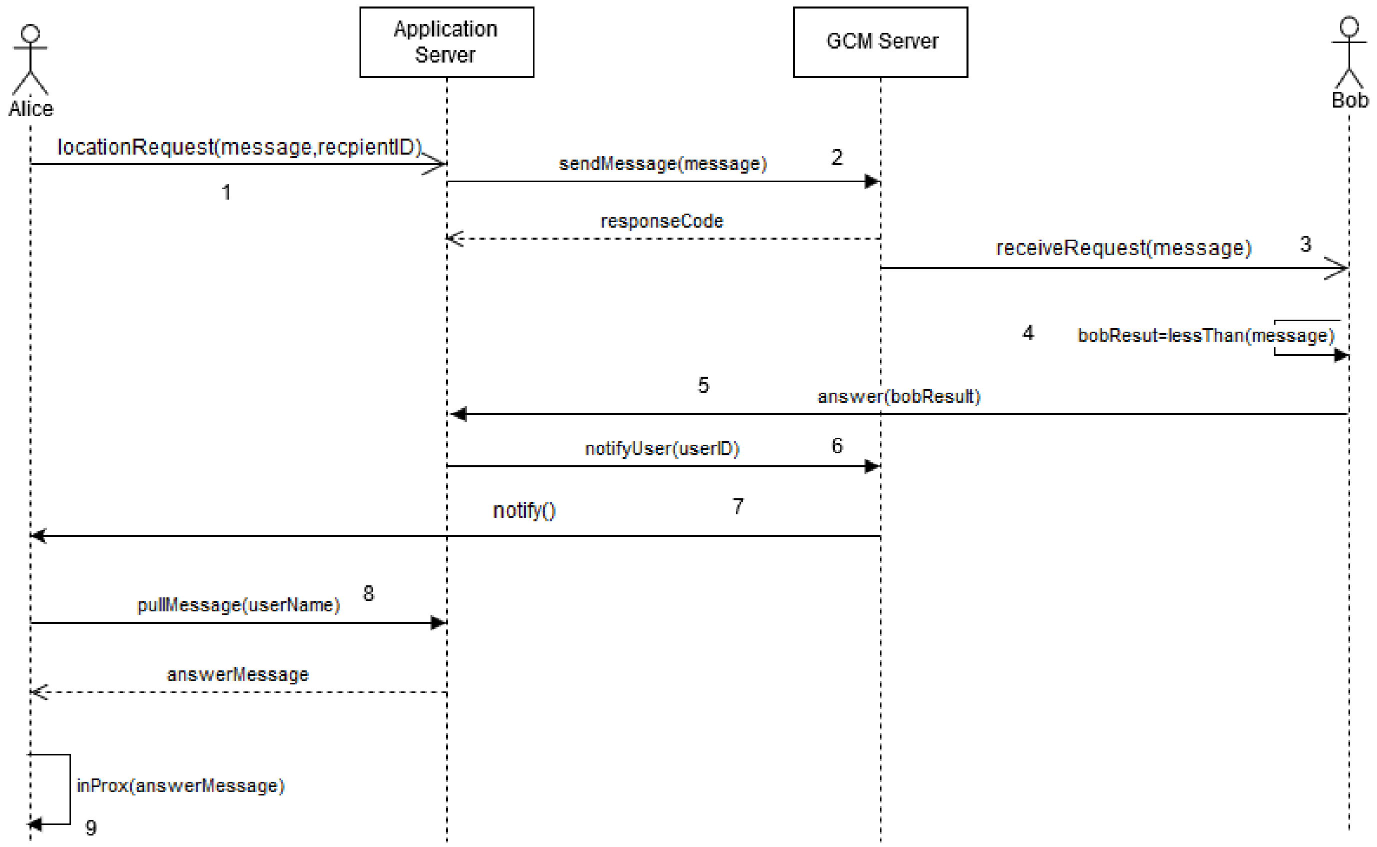


Fig. 1. Sequence Diagram of a Communication Request.

functionality of the mobile application. This is the perfect case.

The analysis of collected data will be of qualitative nature. The researchers will determine if the algorithm is sufficiently efficient by comparing its response time with response time of other applications. The level of applicability of the algorithm will be determined by examining the number of applications that can or cannot incorporate the algorithm without significant change in functionality.

C. VALIDITY THREATS

Runeson and Höst [13] propose 4 types of validities that are relevant to studies in the field of Software Engineering. The four categories are Construct, Internal, External validities and Reliability and they will be used to categorize the validity threats to our study.

Construct validity revolves around how well the data the researchers obtain corresponds to answering the goals of the study. In our study we measure the efficiency and applicability of location privacy-preserving protocols. Efficiency is defined as the amount of time it took to carry out the protocol and is measured by the time it took for the artifact to display the results as well as the protocol time used by the CPU of the smartphone device. These measurements allow us to measure the efficiency of the protocol itself as well as when it is used in a system containing GUI and server communication elements. We believe these results correspond well with our goals. On the other hand, applicability is determined by rudimentary observation of real applications and determining if their features could be replicated by the protocol. This does not take into account much of the software elements of the applications and their limitations, which could possibly be incompatible with the protocols. Mitigation of this threat would require us to analyze the coding and system architecture of such applications, but as we do not have access to such information, the threat will have to remain. Regardless, taking into account both applicability and efficiency results we believe our construct validity is sufficient.

Internal validity is determined based on whether the investigated factor can be affected by a third factor. In our case, our results were obtained from private persons mobile devices which had installed a number of different applications on their phones. These applications could use CPU of the phone at the same time as the artifact of the research study is being executed, thus causing the artifact to run slower. This makes it difficult to tell the real efficiency of the protocol in real conditions, thus negatively affecting our internal validity. Moreover, since the phones contained a different amount of applications prior to testing, the slow down effect could have been different for each device. This makes it difficult to tell if the difference/similarity of the results recorded on different devices was caused by different CPU processors or different applications on the devices. However, the purpose of the study is to test implementations of the state of the art privacy preserving algorithms in real conditions. In reality, it is safe to say that smartphone users will have many different applications in their devices and as such, the protocol must be efficient even with other applications running in the background to be useful in real market mobile applications.

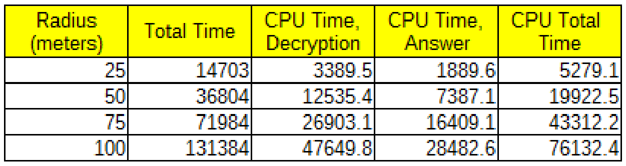
External Validity is concerned about the generalization of the results and their relevance to other researchers. Our artifact was developed in Android OS, therefore it is difficult to judge if the application can be easily adapted to other smartphone devices. In particular, GCM push notifications are only supported by Android devices, meaning that the implementation of the artifact would most certainly have to change if developed on other OS. However, as Android is one of the most popular OS currently on the market our results regarding protocol efficiency and applicability would be relevant to a large amount of application developers and users worldwide. Furthermore, iOS system offers similar push notification functionality to GCM, which we can assume would allow the artifact to preserve similar architecture when ported to iOS devices. Therefore, our external validity is high.

Reliability of a study was defined as the dependency of the study results on researchers. A reliable study could be replicated with the same results by other researchers. One of the main threats to the reliability of our study is the artifact itself: it is likely that if other researchers would implement their own application, they would choose a different approach which could affect the efficiency of the artifact. However, as the code to the artifact of this study is publicly available on GitHub (see Appendix A), researchers trying to replicate study could use that repository to try and replicate results as close as possible.

V. RESULTS

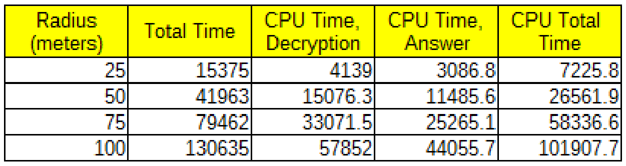
A. EFFICIENCY

The protocol was executed 10 times on each device in ranges of 25, 50, 75 and 100 meters. Tests were started by sending location-proximity requests from Samsung Galaxy S4 Mini device to Nexus 5X, for which the average of results can be seen in Table I. Afterwards the same tests were done for Nexus 5X to Samsung Galaxy S4 Mini proximity checks, for which results are displayed in Table II. The outcome was measured in total time and CPU time in milliseconds. Total time represents the real time the protocol took to display the results to the user, from the moment user opted to locate someone until the result was available in their device. This time includes encryption, decryption, answer generation and communication between the devices and the server. CPU time displays the amount of time answer generation and decryption took in Bob and Alice's smartphone devices respectively. CPU Total Time displays the sum of time that encryption and decryption took. The raw data for the tests can be seen in Appendixes B and C.



Radius (meters)	Total Time	CPU Time, Decryption	CPU Time, Answer	CPU Total Time
25	14703	3389.5	1889.6	5279.1
50	36804	12535.4	7387.1	19922.5
75	71984	26903.1	16409.1	43312.2
100	131384	47649.8	28482.6	76132.4

TABLE I
PROTOCOL TESTING RESULT AVERAGES IN MILLISECONDS FOR SAMSUNG GALAXY TO NEXUS PROXIMITY CHECKS.



Radius (meters)	Total Time	CPU Time, Decryption	CPU Time, Answer	CPU Total Time
25	15375	4139	3086.8	7225.8
50	41963	15076.3	11485.6	26561.9
75	79462	33071.5	25265.1	58336.6
100	130635	57852	44055.7	101907.7

TABLE II
PROTOCOL TESTING RESULT AVERAGES IN MILLISECONDS FOR NEXUS TO SAMSUNG GALAXY PROXIMITY CHECKS.

As the results show, the time for protocol executions increases drastically with increase in radius. Mapping the average speeds for Nexus-to-Samsung and Samsung-to-Nexus into a line chart in Figure 2 suggests that the increase in time is exponential, although more data points are required to say for certain.

It is important to note that the parallelization of the InnerCircle protocol was not implemented in our mobile application. This has a notable negative effect on the efficiency of our application, making it impractical to check location-proximity within distances of 100 meters or higher. However, we believe that parallelization can be substituted by methodically lowering the precision of the coordinates used as input in the protocol. This will be discussed in more detail in section VI.

The efficiency of real Android applications was also tested 10 times for each application. The execution time of the application was measured with a stopwatch, with the countdown starting when the researchers opted to locate a target and ending when the application provided the output. The results for applications output and their averages can be seen in Table III. It is worth to note that application Meet Me and Singles Around Me used GPS to find the user's location, with the latter application never turning off GPS tracking.

B. APPLICABILITY

The applicability of location-proximity protocol to the applications is divided into 3 categories: Not Applicable, Limitedly Applicable and Fully Applicable. The applications were selected by searching for "location social networking" query on Google Play Store. This section will describe the selected applications and justify why we believe a certain results had to be given to the applications. The results are presented in Table IV.

Meet-up: this application allows users to meet and chat with people who have the same interests and get updates on whats happening in their vicinity. This application was considered fully applicable because it could use InnerCirle to find users nearby instead of the current mechanism.

	Meet Me	SKOUT	Badoo	LINK	Tagged	Singles Around Me	Mico	MeetUp	Foursquare
1	2.18	2.3	1.75	1.7	3.1	6.13	3.19	1.86	1.54
2	2.04	2.96	2.27	1.71	2.75	9.21	3.09	2.18	1.82
3	2.13	2.5	1.76	1.74	4.27	9.83	3.05	1.93	1.57
4	2.12	4.48	2.08	1.7	3.21	6.66	3.11	1.67	1.56
5	2.15	3.83	2.23	1.59	3.6	8.24	3.27	1.66	1.58
6	1.83	3.9	2.39	1.6	2.71	3.29	3.04	1.76	1.79
7	2.07	3.31	2.13	2.75	2.83	9.93	2.9	1.73	1.58
8	2.28	1.23	1.85	1.65	2.73	3.43	2.86	1.43	1.85
9	2.78	2.9	1.84	1.6	2.94	3.19	3.11	1.66	1.44
10	2.29	1.59	3.1	1.63	3.53	13.49	3.4	1.38	1.81
Average	2.19	2.9	2.14	1.77	3.17	7.34	3.1	1.73	1.65

TABLE III
EFFICIENCY RESULTS FOR REAL APPLICATIONS IN SECONDS.

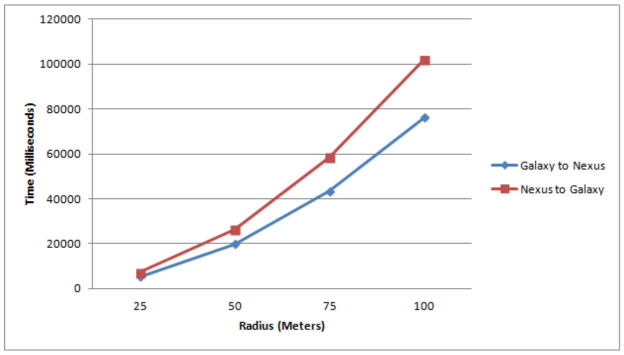


Fig. 2. Averages of Execution Times for Protocol in Different Ranges.

SKOUT: allows users to find people based on search criteria, like age or gender. The application displays the results along with found people's distance from the user. As the application shows precise distance which InnerCircle cannot do, the application would need to change its functionality somewhat to reduce the accuracy of its functions. Therefore InnerCircle protocol is deemed as Limitedly Applicable.

Badoo: an application that allows users to find people nearby their location and even notifies users when other users are extremely close, such as crossing the same street at the same time. This case was deemed to be Limitdly applicable because Badoo displays the location of other users on a map: this requires knowing precise coordinates, which InnerCircle does not provide.

MeetMe: allows users to find and chat with people nearby who have the same interests. The protocol for this application was deemed to be fully applicable since MeetMe just finds people nearby within a certain radius, just like InnerCircle does.

LINK: another meet up application that connects users with people nearby them. The application also allows user to create groups based on location or interests. The protocol is fully applicable for this application because it shows people nearby without disclosing their location and this can be done with InnerCircle.

Tagged: with this application the user can find people based on age or location and chat with people nearby. The protocol is fully applicable since Tagged shows people nearby without showing their exact location, which can be done with InnerCircle.

Singles Around Me: a dating application that connects user with singles nearby based on the specified location range. The user can also chat with and "like" other users. This application is limitidly applicable since it shows the exact location of the users nearby and this is not supported by InnerCircle

Mico: allows user to discover people nearby and share moments with other users. This application is limitedly applicable since it shows the exact location of its users. However, the feature of finding people nearby can be replicated by InnerCircle.

Find My Friend: allows the users to locate their friends and share their own location. The protocol is not applicable for this mobile application as the application requires the exact location of the user.

Family Locator: similarly to the previous application, Family Locator allows users to locate their family members and friends as well as share their own location. An option for hiding/showing their location is also given for the users. As this application provides precise location of its users, InnerCircle protocol is not applicable.

VI. DISCUSSION

A. *INFLUENCE OF SMARTPHONE DEVICES*

Our results indicate that Nexus 5X generates answers faster than Samsung Galaxy S4 Mini, but Samsung Galaxy S4 decrypts the messages faster than Nexus 5X. This contradicts our assumption that Nexus X5, the newer device, would carry out both processes faster. It is our belief that the two devices handle encryption differently, with Samsung Galaxy device generating longer messages which the Nexus device then decrypts, explaining why the newer device decrypts messages slower than the older one. We can also see that

	Application	Not Applicable	Limitedly Applicable	Fully Applicable
1	Meet-me			X
2	SKOUT		X	
3	Badoo		X	
4	LINK			X
5	Tagged			X
6	Singles AroundMe		X	
7	Mico		X	
8	MeetUp			X
9	FourSquare		X	
10	Find My Friends	X		
11	Family Locator	X		

TABLE IV
THE APPLICABILITY OF INNERCIRCLE ALGORITHM TO SELECTED MOBILE APPLICATIONS.

proximity checks from Nexus to Samsung Galaxy were in general slower than Samsung Galaxy to Nexus checks, likely for the previously mentioned reason that Samsung Galaxy device creates longer encrypted messages.

Although not implemented, the parallelization component of InnerCircle could further affect the difference between android devices. As there was only one thread running in the execution, the protocol did not make use out of Samsung's dualcore and Nexus's hexacore CPU processors. In case of multithreaded optimization implementation, we believe hexacore processor of Nexus could afford more threads running in parallel and allow the artifact to run faster than dualcore processor of Samsung. This is important to keep in mind, as our results indicate that for the protocol to be efficient enough for use in real world applications, optimization is necessary.

Overall, judging by the difference of answer generation on our tested devices it seems that it can be expected that the location privacy-preservation protocols will become more efficient for use in real world applications as smartphone devices become more advanced. This means that even though some protocols may be too slow to be implemented in real applications on the market right now, that may change in 2 or 3 years without the need to modify the protocols themselves. However, it is important to keep in mind that a considerable amount of people hold on to their old devices for years instead of getting state-of-the-art phones upon their release. This is particularly significant to protocols such as InnerCircle, which involve considerable amount of computation on both Alice's and Bob's devices.

B. EFFICIENCY

Based on the results for Samsung Galaxy-Nexus proximity checks in radius of 100 meters we see it takes the algorithm 76 seconds to return the answer. Adding times required for devices to communicate with servers and display GUIs, the time required to receive output from the application further increases the time. For comparison, a study by Narayanan et al [15] implementing their own protocols in Android applications found execution times to be 46 seconds for Protocol 1, which is comparable to the InnerCircle time at 50 meter radius values. Such time frames are unacceptable to most users as they are not willing to wait so long for an application to provide a service. Furthermore, it is reasonable to assume that Bob will not always be stationary in his position while the answer is generated on his phone. Given that 74 seconds are enough time to cover a distance of 100 meters by foot (more so by transportation vehicle), it is plausible that Bob's position can change to in/out of range before Alice receives an answer, thus making it no longer valid.

Using InnerCircle algorithm with smaller radius values is more feasible: it took the protocol 5 and 7 seconds to calculate proximity for Galaxy-Nexus and Nexus-Galaxy proximity requests respectively at 25 meter radius. These time frames are much closer to the results of real applications, which provide output to the users in range of 1 to 4 seconds on average, with one application taking 7 seconds. However, real application response time is not affected by the radius specified by the users, so an user could locate people in other cities or countries. This is not likely to happen in InnerCircle even after implementation of all optimization techniques. As we can see in Figure 2, the time taken to execute the protocol increases drastically with small increases in radius values. Protocol 2 from the previously mentioned Narayanan et al [15] study takes 3 seconds to carry out which is sufficient time for implementation in real applications. However, this protocol does not use homomorphic encryption as Narayanan's et al Protocol 1 and Per Halgren's et al [9] InnerCircle protocols, and is less secure as Bob's location can be easily revealed in case Alice collaborates with the server.

Considering the results we obtained it seems that the answer to our research question 1 is mixed: the efficiency of state-of-the-art location-proximity algorithms is sufficient enough in short distances, but is far too slow to be implemented as part of real mobile applications using longer distances. In regards to research question 1.1 we see similar results: the protocols compare well to real LBS mobile applications in small ranges, but quickly fall behind in longer radius values. As the majority of mobile applications focus on distances much longer than 100 meters, our research study has not found enough evidence to suggest that the state-of-the-art location-proximity algorithms are efficient enough for applications in most mobile applications.

However, as mentioned previously, the multithreading component of the protocol was not implemented, which has a severe negative effect on the efficiency of the algorithm. If implemented, multithreading component could reduce the answer generation and message decryption on Bob's and Alice's devices multiple times, depending on the amount of CPU processors present on the devices. In theory, answer generation and decryption should decrease twice for Samsung Galaxy S4 Mini device as it has a dualcore CPU processor, and six times for Nexus 5X as it has a hexacore CPU processor. In this case, proximity checks from Samsung Galaxy to Nexus would take 2, 7.4, 16.1 and 28.5 seconds for ranges of 25, 50, 75 and 100 meters. Proximity checks from Nexus to Samsung

Galaxy would take 2.2, 8.2, 18.1 and 31.6 seconds for the same radius values. While these performance times are much faster than the ones without parallelization component, they match the performance of real market applications only at distances of 25 and 50 meters, with performance for 50 meters radius being equal to the slowest real application, Singles Around Me. Therefore, it appears the protocol would still be inefficient enough for use until more advanced smartphone devices are available on the market and the answers to research questions 1 and 1.1 remain unchanged.

We believe there is an additional option for improving the efficiency of the protocol that would allow to check proximity in large radius values. This option involves manipulating the precision of the coordinates used as input for the InnerCircle algorithm. The precision of the coordinates depends on the number of digits included after the decimal point. When dealing with longitude and latitude degrees, the 8th decimal digit denotes precision of 1.1 millimeters at the equator. Reducing the values by each digit decreases the precision of the coordinates by a factor of 10 (i.e. 7th digit denotes precision of 11.1 millimeters and 6th digit denotes precision of 111.1 millimeters). With this approach application developers can allow their users to check proximity in large radius values through a combination of small radius values with imprecise coordinates. For example, to allow Alice to check proximity to Bob in more significant radius like real mobile applications, such as 3000 meters, the mobile program could really check the proximity in 27 meter range with precision of the world coordinates equivalent to the 3rd decimal digit of longitude and latitude degrees (approximately 111.1 meters). As this approach would only affect world coordinates, but not the radius values used in lessThan() method, the efficiency of the algorithm should not be affected and the radius of 3000 meters should be checked with the same performance it would take to check proximity in the real 27 meter radius. That is, little more than 5 to 7 seconds that it takes to check 25 meters on average, depending on the Android device being used. This would require configuration on the application developer part to determine which radius and precision combinations would provide the desired effect. As InnerCircle algorithm uses world coordinates as input and not latitude and longitude degrees, conversion would need to be determined for the specific world coordinate system used by the application. One negative side effect of this approach is that it increases the range at the edge of the circle in which Bob's coordinates may be displayed incorrectly to Alice. However, this error range would be equal to the precision value of the coordinates. Using previous example, Alice would be able to check the proximity of Bob in the radius of 3000 meters, with the error range being 111.1 meters. Although real applications most likely check the proximity with higher accuracy at the same range, we believe the significance of the error range remains small in comparison the the radius values in which proximity is checked. Moreover, the application incorporating InnerCircle protocol has the added bonus of guaranteeing that the location privacy of its users stays intact, while the same cannot be said about mobile applications without the protocol.

C. APPLICABILITY

Applicability of the protocols to real applications is something that is often not reviewed in depth by research papers. Šikšnys et al [16] briefly mention 2 friend location services but does not explain them in detail or how the location privacy-preserving algorithm could affect the applications if it was implemented. Narayanan et al [15] provides use cases where LBS mobile applications could be used and relate these use cases to the protocols proposed in their research papers. However, it is debatable whether the use cases themselves are sufficient proof that the algorithm could be applicable enough to be used in general applications. One obstacle in implementing location privacy-preserving protocols in mobile applications is the possible limitation of the application's functionality. In case the protocol can only apply to rarely used or unknown applications, the protocol usefulness is questionable regardless of its efficiency. For example, dummy location and k-anonymity based protocols are suitable for situations where a user would want to hide his location from service providers. In mobile applications where location is shared between different users instead such algorithms would not be applicable.

InnerCircle algorithm is created for cases where users are allowed to see the proximity of other users, but not their precise location. Applications that could adapt this protocol can be venue adviser applications, which recommend their users venues nearby their location, and augmented reality mobile games which allow users to interact with other users based on their location proximity (such as playing hide-and-seek or tag). However augmented reality games do not appear to be popular based on our application search on Google Play store and venue advisers may require modification of the InnerCircle protocol as the venues would be treated as Bob and have their locations hidden from the user. This is clearly not a desired effect for the venues like restaurants and bars, which want the users to know their location.

It appears that the algorithm is most useful for applications that facilitate interaction with strangers before a possible meeting in reality. In such cases proximity between users is important as users want to know in advance if the potential meeting is possible, but at the same time want to keep their location private as they have not built enough trust yet to reveal it. As search on Google Play Store shows, such cases are relevant to meet up and/or dating applications. Applications with similar purpose but which focus on interaction between people who already know each other (Find My Friends and Family Locator, which focus on friends and family respectively) seem to place less importance on location privacy and allow their users to see the precise location of different users on a map. For such applications InnerCircle is less applicable.

Overall, in regards to research question 2 the answer varies greatly based on the purpose of the application. It appears that state-of-the-art privacy-preserving location-proximity algorithms can be applied relatively freely without limiting the functionality to applications with the purpose of meeting strangers through the internet. Importance of location privacy in such applications is crucial, and as meet up and dating applications are relatively popular in the mobile market, good location privacy-preserving algorithm applicability means such algorithms can become quite popular in mobile application market.

D. MODIFICATION OF THE PROTOCOL

During the course of the study we have noticed that certain modifications had to be made to the protocol to adapt its architecture to an Android application. First, the original algorithm was capable of performing all its tasks in a single round trip. However, when implementing the algorithm we noticed that the messages between Alice and Bob were much too big to be sent via GCM service, which limits the size of the messages to 4 kb. Having a server socket opened to receive server streamed messages on users' devices permanently was deemed as an inadequate alternative as such solution would cause issues like wasted computing power and possible program crashes, among others. Instead, a combination of server sockets and GCM service was used in the implementation of the artifact, where GCM services were used to notify when a message is available for the users to retrieve via regular socket servers. As a result the implementation of InnerCircle protocol no longer uses a single round trip and instead has none. This means the communication times of our algorithm are much slower than the original algorithm would have. Furthermore, the use of 2 servers by our artifact means that the protocol is no longer decentralized as intended. However, TTPs are still not necessary as, thanks to homomorphic encryption, the servers used by the artifact are unable to retrieve location data about either Bob or Alice.

It is also worth to note that the original InnerCircle protocol generated sum of squares for all values until the maximum radius value allowed to be chosen by the users. This optimization technique, while increasing the efficiency of answer generation, consumed a considerable amount of time. The researchers behind InnerCircle proposed creating sum of squares since the distance in the protocol is always a sum of two squares. The sum of squares was computed only for the radius values used in the application, as this decreases the time to generating the sum of squares. Furthermore, another optimization technique related to the sum of squares is to initialize the sum of squares and store the negated values for each value of sum of square for the available radius values encrypted under their public key in the mobile application itself. This would reduce computation time by considerable amount since lessThan() would retrieve the negated value from the negated values list resulting in less computation.

VII. CONCLUSION

Our research study set out to determine whether privacy preserving algorithms at their current state are sufficient enough for use in real mobile applications on the market. For this study we chose an algorithm that we believe represents state of the art technology best and implemented it as part of our artifact: an Android application that checks the location-proximity between two users. We have found that due to various limitations the protocol required several modifications when implementing it in Android application. Applicability wise the protocol could be applied to a number of popular applications, in particular for the ones that facilitate meetings in real life between strangers, such as meet up and dating apps. However, the performance was sub-par, taking over a minute to check proximity between two users at a radius of 100 meters - a time that users would not be willing to wait when using a mobile application. While implementing full optimization techniques would increase the efficiency of the protocol, we hypothesize that the efficiency of the artifact would match real world applications at radius values 25 and 50 meters, while values at 75 meters and above would still be falling short to be acceptable for potential users. One possible option for overcoming this obstacle is manipulation of the coordinate precision used as input for the privacy preserving protocol. With less precise coordinates the algorithm can effectively check the radius of 3000 meters with the time required to check 27 meter radius - little more than 5 to 7 seconds - which would allow the protocol to be efficient enough for implementation in real applications. Calibration on the software developer part would be required to make sure the values are accurate, but this solution should make the algorithm efficient enough for use in large distance radius values.

Further research on this topic could review the efficiency of location-proximity privacy-preserving protocols after implementing coordinate precision manipulation as a optimization technique. Research examining in depth the effects on the accuracy of proximity checking when using various precision for coordinates could also be beneficial. Finally, recreation of the study with different proximity checking algorithms could determine if other state of the art methods for preserving location privacy are more efficient.

ACKNOWLEDGMENT

We would like to thank Per Hallgren, one of the researchers behind the InnerCircle algorithm, for his role in supervising our progress in this research study.

REFERENCES

[1] Zhu, X., Lu, Y., Zhu, X., & Qiu, S. (2013, December). A Location Privacy-Preserving Protocol Based on Homomorphic Encryption and Key Agreement. In Information Science and Cloud Computing Companion (ISCC-C), 2013 International Conference on (pp. 54-59). IEEE.

[2] Chien-Ping Wu, Chen-Che Huang, Jiun-Long Huang & Chih-Lin Hu, ”On preserving location privacy in mobile environments,” Pervasive Computing and Communications Workshops

[3] Barkhuus, L., & Dey, A. K. (2003, July). Location-Based Services for Mobile Telephony: a Study of Users' Privacy Concerns. In INTERACT (Vol. 3, pp. 702-712).

[4] Xu, H., & Gupta, S. (2009). The effects of privacy concerns and personal innovativeness on potential and experienced customers' adoption of location-based services. Electronic Markets, 19(2-3), 137-149.

[5] Zickuhr, K. (2012). Three-quarters of smartphone owners use location-based services. Pew Internet & American Life Project.

[6] Gedik, B., & Liu, L. (2008). Protecting location privacy with personalized k-anonymity: Architecture and algorithms. Mobile Computing, IEEE Transactions on, 7(1), 1-18.

[7] Rivest, R. L., Adleman, L., & Dertouzos, M. L. (1978). On data banks and privacy homomorphisms. Foundations of secure computation, 4(11), 169-180.

[8] Gentry, C. (2009, May). Fully homomorphic encryption using ideal lattices. In STOC (Vol. 9, pp. 169-178).

[9] P. Hallgren, M. Ochoa and A. Sabelfeld, ”InnerCircle: A parallelizable decentralized privacy-preserving location proximity protocol,” Privacy, Security and Trust (PST), 2015 13th Annual Conference on, Izmir, 2015, pp. 1-6.

[10] Zhou, Y., Zhang, X., Jiang, X., & Freeh, V. W. (2011). Taming information-stealing smartphone applications (on android). In Trust and Trustworthy Computing (pp. 93-107). Springer Berlin Heidelberg.

[11] Kido, H., Yanagisawa, Y., & Satoh, T. (2005, July). An anonymous communication technique using dummies for location-based services. In Pervasive Services, 2005. ICPS'05. Proceedings. International Conference on (pp. 88-97). IEEE.

[12] Puttaswamy, K. P., & Zhao, B. Y. (2010, February). Preserving privacy in location-based mobile social applications. In Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications (pp. 1-6). ACM.

[13] Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering, 14(2), 131-164.

[14] T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory, 31(4):469–472, 1985.

[15] Narayanan, A., Thiagarajan, N., Lakhani, M., Hamburg, M., & Boneh, D. (2011, February). Location Privacy via Private Proximity Testing. In NDSS.

[16] Šikšnys, L., Thomsen, J. R., Šaltenis, S., Yiu, M. L., & Andersen, O. (2009). A location privacy aware friend locator. In Advances in Spatial and Temporal Databases (pp. 405-410). Springer Berlin Heidelberg.

APPENDIX A

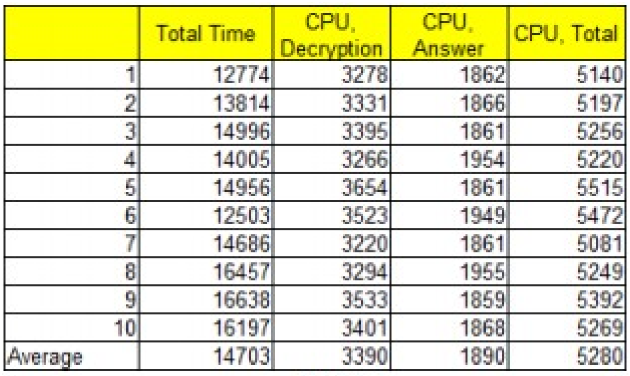
The artifact developed in this research study is available at the following location:

https://github.com/SimonasStirbys/InnerCircle

The latest commit is on May 30, 2016.

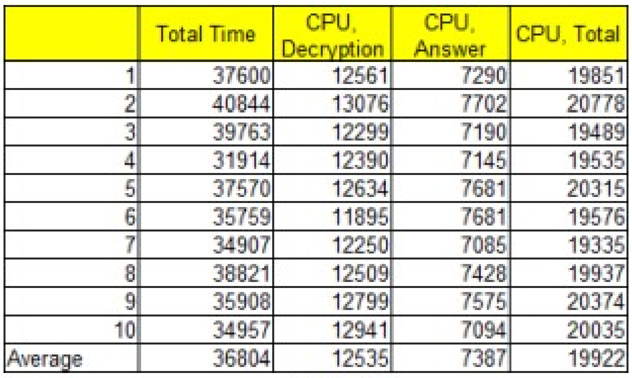
APPENDIX B

Tables for testing results in milliseconds for Nexus to Samsung Galaxy proximity checks at distances of 25, 50, 75 and 100 meters.



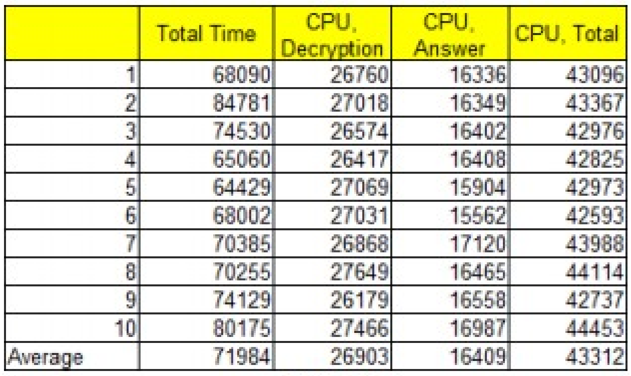
	Total Time	CPU, Decryption	CPU, Answer	CPU, Total
1	12774	3278	1862	5140
2	13814	3331	1866	5197
3	14996	3395	1861	5256
4	14005	3266	1954	5220
5	14956	3654	1861	5515
6	12503	3523	1949	5472
7	14686	3220	1861	5081
8	16457	3294	1955	5249
9	16638	3533	1859	5392
10	16197	3401	1868	5269
Average	14703	3390	1890	5280

TABLE V
TESTING RESULTS FOR SAMSUNG GALAXY TO NEXUS PROXIMITY CHECKS AT 25 METERS IN MILLISECONDS



	Total Time	CPU, Decryption	CPU, Answer	CPU, Total
1	37600	12561	7290	19851
2	40844	13076	7702	20778
3	39763	12299	7190	19489
4	31914	12390	7145	19535
5	37570	12634	7681	20315
6	35759	11895	7681	19576
7	34907	12250	7085	19335
8	38821	12509	7428	19937
9	35908	12799	7575	20374
10	34957	12941	7094	20035
Average	36804	12535	7387	19922

TABLE VI
TESTING RESULTS FOR SAMSUNG GALAXY TO NEXUS PROXIMITY CHECKS AT 50 METERS IN MILLISECONDS.



	Total Time	CPU, Decryption	CPU, Answer	CPU, Total
1	68090	26760	16336	43096
2	84781	27018	16349	43367
3	74530	26574	16402	42976
4	65060	26417	16408	42825
5	64429	27069	15904	42973
6	68002	27031	15562	42593
7	70385	26868	17120	43988
8	70255	27649	16465	44114
9	74129	26179	16558	42737
10	80175	27466	16987	44453
Average	71984	26903	16409	43312

TABLE VII
TESTING RESULTS FOR SAMSUNG GALAXY TO NEXUS PROXIMITY CHECKS AT 75 METERS IN MILLISECONDS.

	Total Time	CPU, Decryption	CPU, Answer	CPU, Total
1	109087	47217	27727	74944
2	118666	47407	29152	76559
3	129919	47196	28765	75961
4	107755	47644	28268	75912
5	125113	48920	28630	77550
6	153234	47847	29245	77092
7	116975	47878	28221	76099
8	119147	47284	27925	75209
9	191815	48164	28711	76875
10	142132	46941	28182	75123
Average	131384	47650	28483	76133

TABLE VIII
TESTING RESULTS FOR SAMSUNG GALAXY TO NEXUS PROXIMITY CHECKS AT 100 METERS IN MILLISECONDS.

APPENDIX C

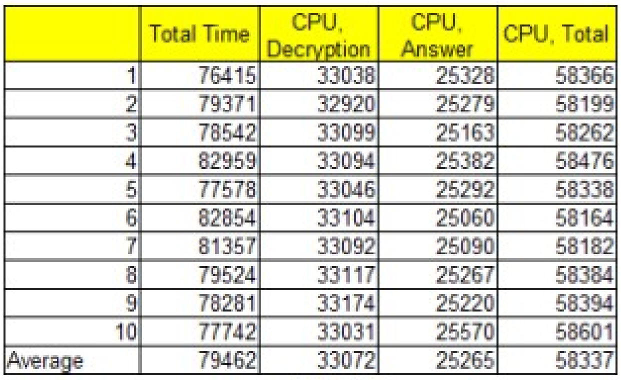
Tables for testing results in milliseconds for Samsung Galaxy to Nexus proximity checks at distances of 25, 50, 75 and 100 meters.

	Total Time	CPU, Decryption	CPU, Answer	CPU, Total
1	13514	4162	2997	7159
2	14982	4098	2931	7029
3	16590	4133	3051	7184
4	16368	4105	2979	7084
5	15187	4180	2927	7107
6	15896	4117	2934	7051
7	15258	4202	3037	7239
8	16688	4062	3713	7775
9	13719	4132	3202	7334
10	15546	4199	3097	7296
Average	15375	4139	3087	7226

TABLE IX
TESTING RESULTS FOR NEXUS TO SAMSUNG GALAXY PROXIMITY CHECKS AT 25 METERS IN MILLISECONDS

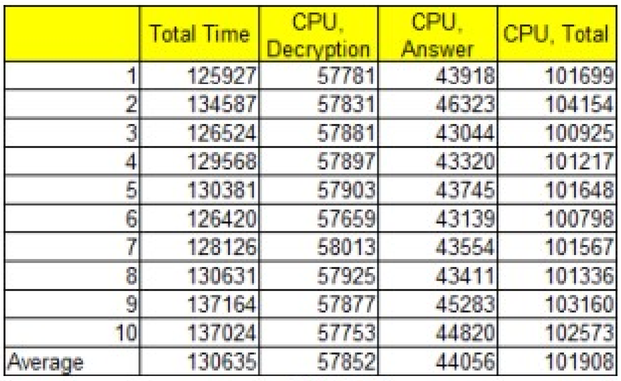
	Total Time	CPU, Decryption	CPU, Answer	CPU, Total
1	39924	15131	11278	26409
2	46712	15145	11088	26233
3	38783	15081	11879	26960
4	42749	15055	11634	26689
5	43848	15084	11674	26758
6	43296	15092	11624	26716
7	41757	14848	11613	26461
8	41995	15139	11415	26554
9	40883	15118	11299	26417
10	39685	15070	11352	26422
Average	41963	15076	11486	26562

TABLE X
TESTING RESULTS FOR NEXUS TO SAMSUNG GALAXY PROXIMITY CHECKS AT 50 METERS IN MILLISECONDS.



	Total Time	CPU, Decryption	CPU, Answer	CPU, Total
1	76415	33038	25328	58366
2	79371	32920	25279	58199
3	78542	33099	25163	58262
4	82959	33094	25382	58476
5	77578	33046	25292	58338
6	82854	33104	25060	58164
7	81357	33092	25090	58182
8	79524	33117	25267	58384
9	78281	33174	25220	58394
10	77742	33031	25570	58601
Average	79462	33072	25265	58337

TABLE XI
Testing Results for Nexus to Samsung Galaxy Proximity Checks at 75 Meters in Milliseconds.



	Total Time	CPU, Decryption	CPU, Answer	CPU, Total
1	125927	57781	43918	101699
2	134587	57831	46323	104154
3	126524	57881	43044	100925
4	129568	57897	43320	101217
5	130381	57903	43745	101648
6	126420	57659	43139	100798
7	128126	58013	43554	101567
8	130631	57925	43411	101336
9	137164	57877	45283	103160
10	137024	57753	44820	102573
Average	130635	57852	44056	101908

TABLE XII
Testing Results for Nexus to Samsung Galaxy Proximity Checks at 100 Meters in Milliseconds.