Implementation of Boneh Protocol 3 in Location Based Services (LBS) to Provide Proximity Services

L. Ertaul, N. Shaikh, S. Kotipalli
Mathematics and Computer Science, California State University East Bay, Hayward, USA

Abstract – In recent years, smartphones have taken over as the pocket technology of choice. More than a half of smartphone owners use a location based information service of some kind. And a core component of Location Based Services (LBS) is proximity testing of users. These services determine if two mobile users are close to each other without requiring them to disclose their exact locations. In this paper, we present Boneh Protocol 3 which supports private proximity testing by using location tags. We study the use of "location tags" generated from the physical environment in order to strengthen the security of proximity testing in Boneh Protocol 3. In this paper, we attempt to provide a realistic assessment of proximity testing for location-based services by implementing Boneh Protocol 3. We used Android platform for an implementation of Boneh protocol 3.

Keywords- Location Based Services; Smart Phones; Proximity Testing; Location Tags; Boneh Protocol 3.

1. INTRODUCTION

Mobile phones and the Internet have revolutionized the communication and lifestyle of people. Due to the growing number of smartphone users, location-based services are growing in popularity. An increasing number of mobile phones allow people to access the Internet where ever they are and whenever they want. From the Internet they can obtain information on places (city maps, restaurants, museums, hospitals). Such kind of restaurant search with respect to position and time can be done by use of LBS [1]. Thus, one can define Location Based Services as-

"Location Based Services are information services accessible with mobile devices through the mobile network and utilizing the ability to make use of the location of the mobile device" [2]

"A Location Based Services is a wireless IP (Internet Protocol) service that uses geographic information to serve a mobile user" [3]

There exist a number of LBS providing location sharing. This includes Google Latitude, Facebook places, Foursquare, Loopt, and a large number of smartphone applications [4], [5]. In recent years there has been considerable research on privacy in LBS. The fundamental problem seem to be that few people would like even their closest friends to know their location all the time, yet will allow distant acquaintances to know their location some of the time [5], [11]. These definitions describe Location Based Services (LBS) as an intersection of three technologies (*see figure 1*), such as the mobile telecommunication system and hand held devices, from Internet and from Geographic Information Systems (GIS) with spatial databases [13].

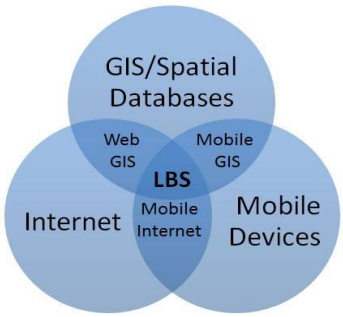

Fig.1. LBS as an intersection of technologies

The paper is organized as follows. The Security and Privacy Concerns in LBS are reviewed in Section 2 followed by Proximity Services and Private Set Intersection in Section 3 and 4. The Location Tags and Private Proximity Testing are discussed in section 5 and 6 and then protocol 1 and 2 are presented in section 7, 8. In section 9, the description of Boneh Protocol 3 is reported. To end with section 10 and 11, Implementation of Boneh Protocol 3 and its experimental results are presented respectively.

2. SECURITY AND PRIVACY ISSUES IN LBS

According to much of the research in location-based computing, privacy is an essential issue and the subject is often addressed in terms of how sensitive information is kept secured in the application [7]. A major privacy concern with the use of Location Based Services is the release to untrusted third parties of the user precise location information. This concern applies to proximity services as well [1]. One of the biggest concerns is that it can be possible to compile a very detailed picture of someone's movements if they are carrying a wireless device that communicates its location to network operators [6]. LBS providers must alleviate consumer privacy fears by implementing secure network and encryption technologies to curb illegal activity [6], [16].

2.1 Privacy Requirements

In general, privacy-preserving systems for LBS services are expected to satisfy some or all of the basic properties below [18].

- **Location Privacy:** The protocol does not reveal the (exact) user's location information to the LBS provider.
- **Identity Privacy (Untraceability):** The LBS provider is not able to find the identity of the user, based on the location information received during the user access [15], [18].
- **Tracking Protection (Unlinkability):** The LBS provider is not able to link two or more successive user positions [15], [18], [19].

2.2 Security Requirements

Access control in LBS involves satisfying some or all of the following security properties [20].

- **Mutual Authentication:** In order to protect themselves from spoofing attacks communication messages between system entities should be authenticated and integrity-protected [18], [20].

3. PROXIMITY SERVICES

Proximity based services are a special class of Location Based Services in which the service adaptation depends on the comparison between a given threshold value and the distance between two users [4]. These services inform users when they are within a certain distance of other people, businesses, or other things [10], [15]. Proximity testing is asymmetric which means one party will learn if the other party is nearby whereas the other party learns nothing [15]. In our paper we show that it is indeed possible to provide location functionality in a private manner. What this means is that a pair of friends will be automatically notified when they are nearby, but otherwise no information about their locations will be revealed to anyone.

Let us consider a application of proximity testing, keeping in mind that different applications require different proximity granularity [1].

- Alice and Bob are friends, and are serendipitously notified that they are shopping in the same mall. They meet and have a pleasant time together. Alternatively, Alice and Bob first meet online, but later decide to meet in person at a coffee shop. Alice arrives first and is notified when Bob arrives [1].
- Alice would like to get dinner with her friend Bob who travels a lot. Using privacy-preserving proximity testing, Alice can check if Bob is town before calling him. Note that for this application the proximity granularity is a wide geographic area [1].
- Bob, a student lands at his college airport and wants to check if anyone from his college is currently at the airport and can give him a ride to campus [1].
- Alice is a manager who wants to automatically record who is present at her daily meetings. However, her employees do not want their location tracked. Privacy preserving proximity testing over this well organized group allows satisfying both requirements [1].

3.1 Proximity Threshold

The distance threshold for proximity detection should not be globally fixed but instead configurable by each user. This is because a larger threshold is neither strictly worse nor strictly better than a smaller one, either from the security or the functionality perspective. With a larger threshold, the user is easier to locate but in case of a match their location is revealed less accurately [1].

4. PRIVATE SET INTERSECTION

Boneh protocol 3 is based on location tags and these are generated by 2 parties who wish to do the proximity test. Broadly speaking if these location tags have few in common, then we conclude that the parties are nearby and it there is no match, we understand that they live far away. In order to find the matching set of intersection, there are various methods proposed. In Boneh protocol, we use private set intersection proposed by Freedman, Nissim and Pinkaas [10].

5. LOCATION TAGS

A location tag is a secret associated with a point in space and time. It is a collection of location features derived from (mostly electromagnetic) signals present in the physical environment. Location tagging is a procedure to extract the tag from a point in space-time, together with a comparison or matching function [1], [17].

5.1 Properties of Location Tags

When compare the location tags, we need to compare two vectors that match approximately, fuzzy set intersection. Location tag is equal to vector and matching function i.e. space-time [15]. The two key properties are:

- ***Unpredictability***-Cannot produce matching tag unless nearby
- ***Reproducibility***-Two devices at same place & time produce matching tags (not necessarily identical) [17].

Location tags provide a different model for proximity testing. The main advantage is that since the location tags of the two parties need to match, spoofing the location is no longer possible, which stops online brute force attacks [1]. The main disadvantage is that users no longer have control over the granularity of proximity: the notion of neighborhood is now entirely dependent on the type of location tag considered [1], [17], [12].

6. PRIVATE PROXIMITY TESTING

In this section we analyse different ways to compute the proximity of Alice and Bob in terms of performance and accuracy. The obvious solution would be to calculate the distance between their positions and decide if the distance is lower than some threshold.

6. 1 Asymmetry

Proximity testing is asymmetric: one party will learn if the other party is nearby whereas the other party learns nothing [1].

The position of Alice along with a given range defines a circle, and the problem is to test if Bob is inside or outside the circle. Another solution is to approximate the area of the circle with cells of a grid. A position is then mapped to a cell, having a unique identifier, in the grid. Using this approach, proximity testing can be reduced to set inclusion as noted by others [5]. The way we detect when two friends are nearby is by dividing the plane [1], [13] into a system of 3 overlapping hexagonal grids. Cryptographic protocols for "Private Equality Testing" allow a pair of users to compare if they are within the same grid cell, but otherwise reveal nothing [1]. *See figure 2*

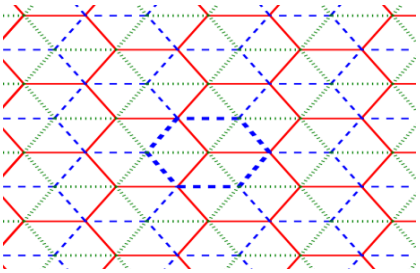

Fig.2. Three overlapping hexagonal grids. A blue grid cell is highlighted

7. PROTOCOL 1 SYNCHRONOUS PRIVATE EQUALITY TESTING

In this protocol the server is used only to forward messages between the two parties, and does not perform any computation. It is based on a mechanism of Lipmaa [27]. The protocol has the following characteristics:

- It is synchronous, i.e., both parties need to be online.
- Each party performs either 2 or 3 exponentiations.
- There are two rounds, namely Alice sends a message to Bob (through the server) and Bob responds to Alice
- Communication is about 40 bytes per edge per time interval using elliptic curves of size 160 bits (additional end-to-end encryption introduces a negligible overhead).

It is secure against arbitrary collusion assuming the hardness of the standard Decision Diffie-Hellman problem [1].

8. PROTOCOL 2 FAST ASYNCHRONOUS PRIVATE EQUALITY TEST WITH AN OBLIVIOUS SERVER

Our second private equality test is novel and requires far less communication and computation, but is only secure assuming the server does not collude with either party. The server learns nothing at the end of the protocol. The reason for the performance improvements is that this protocol uses three parties (Alice, Bob, and server) and is therefore able to rely on information theoretic methods such as secret sharing [1].

9. BONEH PROTOCOL 3

Boneh and team from Standford have proposed 2 versions of this protocol. Now let's look at the 1st version of the protocol. In this protocol, let's suppose Alice wants to know if Bob is near or not. So the protocol would work as follows :

- Alice generates a polynomial p from her set of location tags.
- Alice then sends the encrypted polynomial coefficients E(p) to Bob.
- Bob the calculates his own polynomial p(b) which his location tags and then encrypts it as E(p(b)).
- Then Bob picks random r on E(p) and computes E(r(p(b)) using polymorphic encryption.
- Then Bob sends Alice the permutation of encryptions computed in earlier step.
- Alice then decrypts it and outputs the nonzero decryptions as intersection of A and B.

This protocol has two disadvantages. First, it requires |A|.|B| modular exponentiations (E(p(b)) can be evaluated using Horner's rule [22] using O(|A|) modular exponentiations, and there are |B| such encryptions to compute). Second, it is only secure against semi-honest players. There is a version that handles malicious players, but it is significantly less efficient. More recent protocols can be more efficient [10], [22], [24], [25].

More importantly, revealing the size of the intersection can lead to security problems. For example, in our above example, Alice would come to know the intersection set (no of matching location tags) and she could resort to dictionary attack in case the threshold is very small. So to avoid the weaknesses pointed out in the version 1, the private threshold set intersection rule has been relaxed and version 2 has been developed on this basis. In version 2 protocol, neither of the parties will come to know about the intersection set. Instead one of the party that is seeking to know the proximity of its friend will come to know if set intersection has exceeded the threshold or not, but nothing other than that [5]. So this protocol will ensure the privacy between both the parties. The threshold value(t) and the number of location tags that needs to be generated (n) both are universal constants and if we could allow these values to change, there might be possibility of security issues (Brute-force attack) as mentioned in version 1 protocol. Now let's look at the protocol version 2 in detail [1].

- Alice generates its location tags using any of the generation techniques.
- Alice then uses one of the encoding techniques known, to convert her location tags into 'n' set of vertices say P {(p1, x1)(p2, x2)…… (pn, xn)}, where pi belongs F and xi belongs to F.
- Similarly Bob also generates his location tags using one of the generation techniques.
- He also encodes his location tags into a set Q{(q1, y1), (q2, y2)…. (qn, yn)}.
- Alice constructs a polynomial p of degree n-1 defined by the points P using Lagrange's interpolation technique [26].
- Alice picks a random set of points R on polynomial, p such that R ∩ P = {} and |R| = 2(n-t), where n is the number of location tags and t is the threshold.
- Alice sends R points to Bob.
- Bob then tries to find a polynomial p' such that its degree is 2n-t of points (Q U R) that Bob has.
- If bob is able to find a polynomial through LaGrange interpolation. He outputs 1, which means Alice is nearby him.
- Otherwise he outputs 0 which means Alice is far away from Bob.

This protocol version is asymmetric because here only Bob learns about the Alice's proximity while Alice remains uninformed. If Alice also wants to test Bob's proximity, the protocol needs to be run from the other end. This above protocol produces accurate results of proximity and this can

proved by the help of Berlekamp Massey algorithm as follows:

Suppose there are k pairs (xi, yi) over a field F and a degree parameter d, then if there exists a polynomial p that passes through at least (k+d)/2 of the points, BM outputs p otherwise BM outputs p. The proof of correctness now continues.

- **Case 1.** When Alice and Bob are nearby, there are at least t + 2(n - t) = 2n - t points on the polynomial. Substituting k = n + 2(n - t) = 3n - 2t and d = n, so will be able to find a polynomial
- **Case 2**. When Alice and Bob are far apart, this implies |A ∩ B| < t. This means that there are fewer than 2n - t points on the polynomial p, and by BM theorem, Bob will fail to find an appropriate polynomial.

10. IMPLEMENATION

This protocol can be better understood by looking at the following numerical example.

- *Let's assume Alice has values {91, 62,133}. She encodes them into set of points P = {(9,1), (6,2),(13,3)} where every entry is less than modulus 19.*
- *Alice then constructs a polynomial passing through the points of P by Lagrange interpolation, which is f(x) = 5*x^2+x+3 and picks 2(n-t) = 2(3-2) = 2 points and forms R.*
- *Let {4,5} be these points, then R = {(f(4),4), (f(5),5)} = {(11,4),(0, 5)}*
- *Bob gets R from Alice and let Bob's values be {62, 14, 27}, he then forms his Q using same encoding technique of Alice (lesser than modulus 19) into Q = {(6,2),(1,4),(2,7)}.*
- *Using Berlekamp-Massey algorithm [23] Bob supposed to find a 4th degree (2(3)-2). And since (Q U R) ∩ P is {(6,2)}, Bob is ouput 1 meaning Alice is nearby.*

We have implemented this protocol in Android platform as follows. We use separate emulators () for Alice and Bob and to run the application as show below. Android 2.2 (API level 8) for Alice (5556) and Bob (5554), and to run the application as show below. *See Figure 3and 4.*

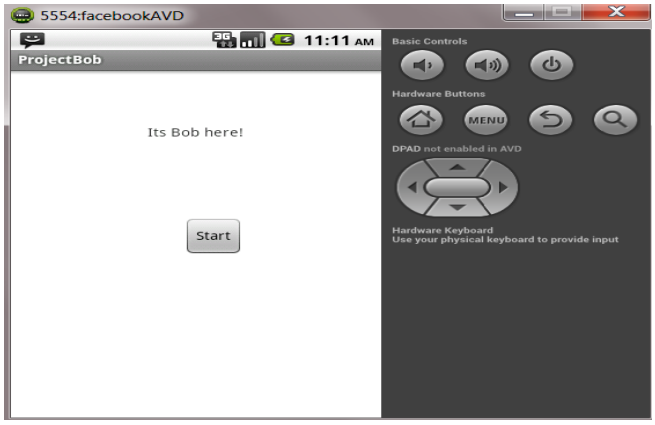

Fig.3. Start message from Bob

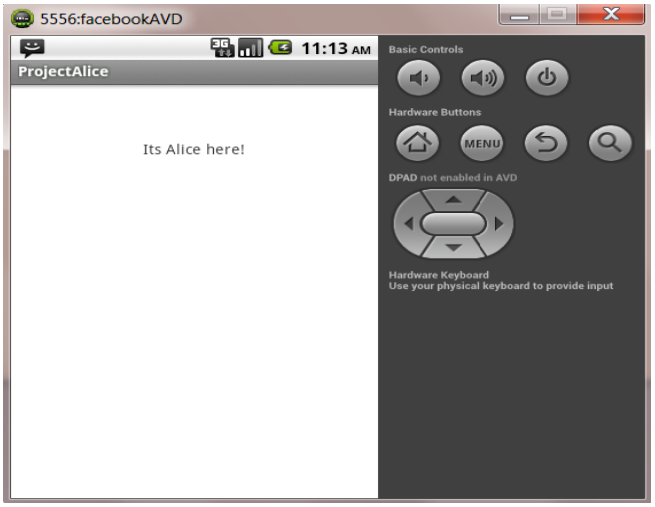

Fig.4. Message from Alice

SMSManager class of android package was used for sending and receiving messages between Alice and Bob emulators.

```
SmsManager sms = SmsManager.getDefault();
sms.sendTextMessage(phoneNumber,null, message, pi,
null);
```

- **Alice sends R to Bob**

```
SmsManager sms1 = SmsManager.getDefault()
sms1.sendTextMessage("5554", null, message, null,
null);
```

We implemented a separate class for receiving SMS from other emulators as follows.

```
Object messages[] = (Object[]) bundle.get("pdus");
      SmsMessage SMS[] = new sms
Message[messages.length];
      for (int n = 0; n < messages.length; n++) {
         SMS[n] = Sms Message .createFromPdu((byte[])
messages[n])
}
```

- **To get the last message received from the inbox**

```
SMS[0].getMessageBody()
```

Once the message is received from Bob, Alice starts calculating the location tags. Since location tags are difficult to be calculated with the present hardware available, we used Random class of java for generating random numbers as follows.

- **For creating random P (Alice)**

```
Random rand = new Random();
for (int i = 0; i < N; i++)
{
      randP[i] = rand.nextInt(29 -11) + 11;
}
```

- **Here (29-11) is the range of the random numbers that can be created**

Then these location tags are encoded into vertices as follows.

```
for(int i = 0; i < N; i++) {
data[i][0] = (int) randP[i] / 10;
   data[i][1] = (int) randP[i] % 10;
   }
```

Then using LaGrange interpolation [26] of the matrices (constant and coordinate) are computed by substituting in a (n-1) degree polynomial as follows:

```
for (int i = 0; i < N; i++) {
     for (int j = 0; j < N; j++) {
   actualdata[i][j] = 1.0;
   for (int k = 0; k < j; k++) {
 actualdata[i][j] *= data[i][0];   } }
   constdata[i][0] = data[i][1];
 }
```

Then coefficient matrix is calculated through computing matrix mathematics like determinant, transpose, inverse, multiplication of matrices. These were implemented using different java classes Matrix.java and MatrixMathematics.java respectively.

- **Finding determinant of matrix**

```
public static double determinant(Matrix matrix) throws
NoSquareException {
     if (!matrix.isSquare())   throw new
NoSquareException("matrix need to be square.");
    if (matrix.size()==2)
{  return (matrix.getValueAt(0, 0) * matrix.getValueAt(1,
1)) - ( matrix.getValueAt(0, 1) * matrix.getValueAt(1,
0));
}
double sum = 0.0;
for (int i=0; i<matrix.getNcols(); i++) {
       sum += changeSign(i) * matrix.getValueAt(0, i) *
determinant(createSubMatrix(matrix, 0, i));  }
return sum;
}
```

- **Calculating transpose of matrix**

```
public static Matrix transpose(Matrix matrix)
{
  Matrix transposedMatrix = new
Matrix(matrix.getNcols(), matrix.getNrows());
 for (int i=0;i<matrix.getNrows();i++) {
      for (int j=0;j<matrix.getNcols();j++) {
     transposedMatrix.setValueAt(j, i,
matrix.getValueAt(i, j));
  } }
  return transposedMatrix;
}
```

- **Matrix inverse**

```
public static Matrix inverse(Matrix matrix) throws
NoSquareException {
return (transpose(cofactor(matrix)).
multiplyByConstant(1.0/determinant(matrix)));
}
```

On running the application, the protocol gets triggered on Alice receiving a start message from Bob as show below. *See Figure 5.*

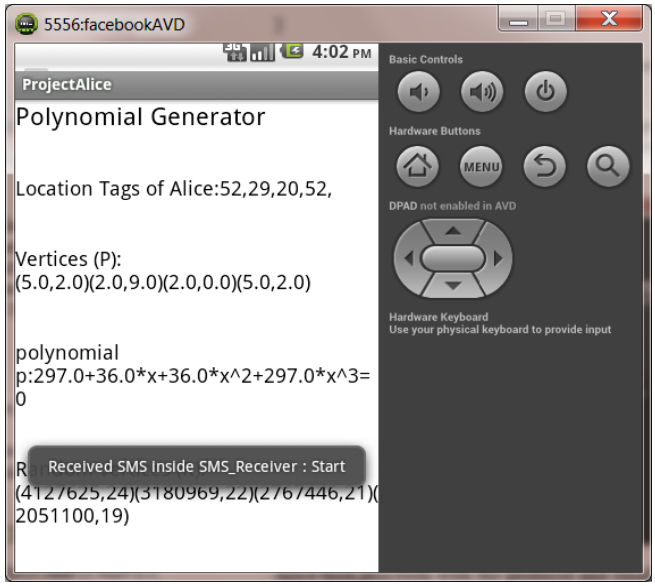

Fig. 5. Alice received a start message from Bob

Alice then proceeds with the protocol and finally sends R in a text message to Bob as follows. *See Figure 6*

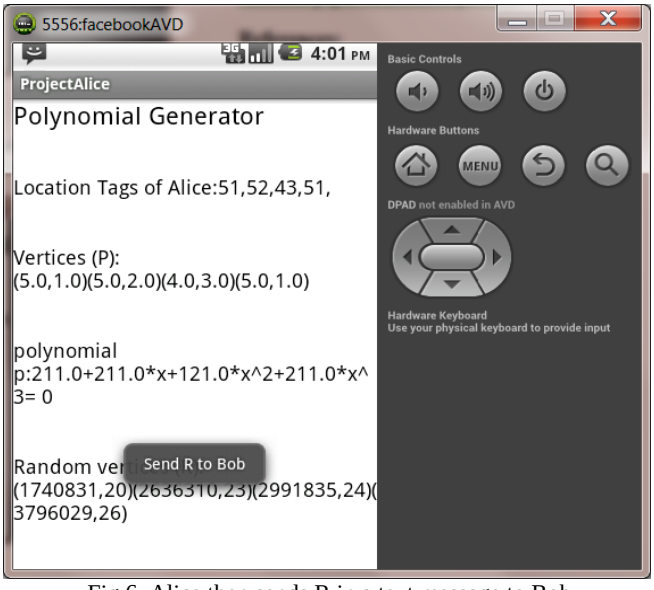

Fig.6. Alice then sends R in a text message to Bob

Bob in the meanwhile does the same processing to find Q and on receiving R from Alice will then try to find a polynomial that passes through 2n-t points. If he is successful, he outputs the following. However, if he is not

able to generate a polynomial, he outputs the following. *See Figure 7,8.*

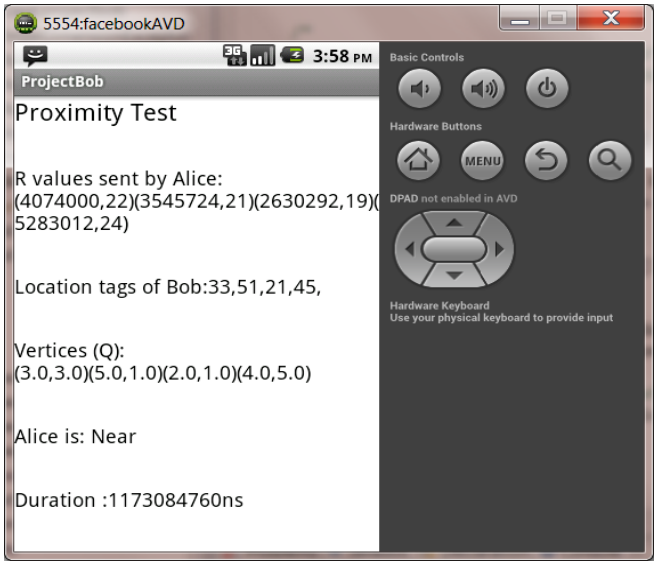

Fig.7. Bob is receiving R from Alice

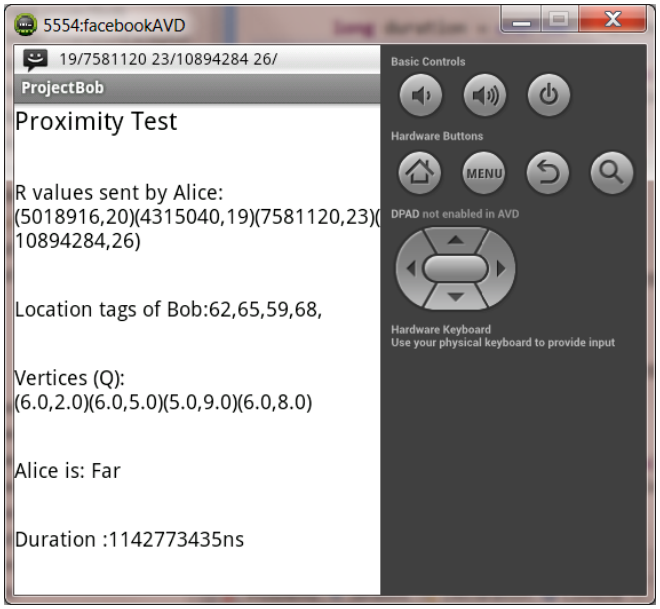

Fig.8. Proximity successful, Alice is near to Bob

11. RESULTS

When the protocol was run on Android platform with 2 emulators 1 each of Alice and Bob, it took 1.232 sec on an average for Bob to get the proximity of Alice. The performance of this protocol on android platform is good. The most time consuming parts of the protocol are the matrix operations like inverse & multiplication. This might take a longer time if the values of n & t are large. And this can be improved by implementing Strassen's algorithm for matrix multiplication which of order O(N ^ 2.8) or Coppersmith-Winograd algorithm of order O(N^2.3), when compared to the standard algorithm O(N^3). The performance of this protocol with multiple Bobs (Senders testing the proximity test of Alice simultaneously) can is represented graphically as follows.

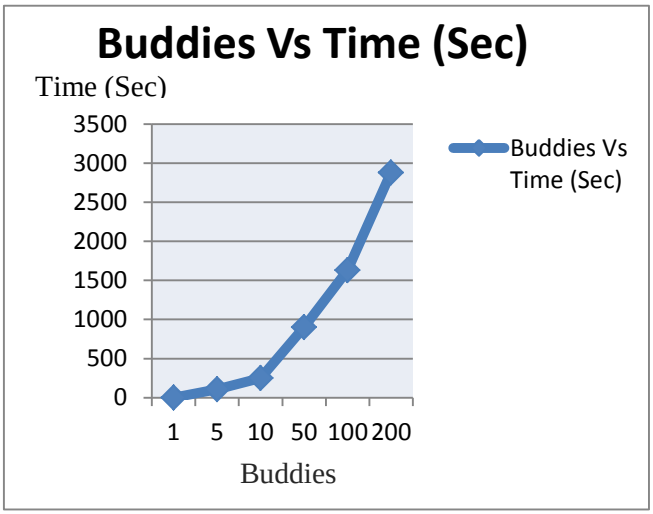

Fig.9. The performance of the protocol with multiples of Bobs

As the performance of this protocol is good in Android Platform but the main limitation is the ability of phone hardware to extract location tags. Currently the main viable method is using WiFi traffic; we showed experimentally that robust tags can be extracted within a few seconds.

On increasing the size of the location tags, the performance of this protocol is as depicted below:

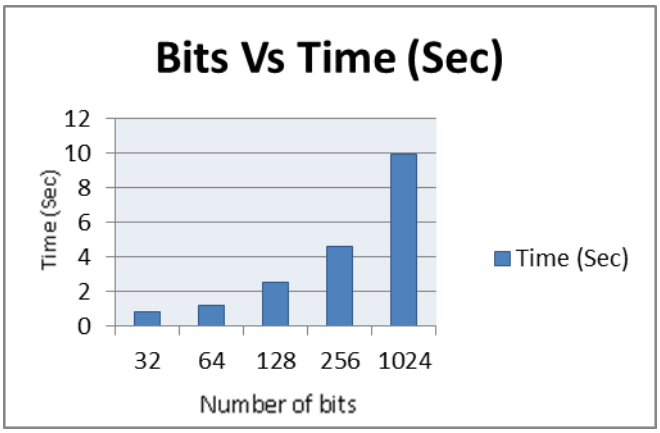

Fig.10. Performance of the protocol with increase in location tags

CONCLUSION

Location privacy is an important and growing concern in the real world today. In this paper we presented Boneh Protocol 3, a privacy preserving protocol for proximity service. We proved its correctness with respect to privacy preferences and we showed the results of an extensive experimental evaluation. Proximity is a checking for inclusion of one user's location inside another user's vicinity, offering users control over both location privacy and accuracy of proximity detection. We have implemented an actively secure protocol for proximity testing. Through the scenario that we targeted on Android Platform, from the results we have shown that it is feasible to execute the protocol on contemporary mobile devices through the android emulator. The protocol discussed in this paper doesn't use any cryptographic algorithmes. It merely uses encoding techniques to convert the location tags into

vertices. So it remains a question unless it comes in to practical use.

REFERENCES

[1] A.Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh. Location privacy via private proximity testing. Network and Distributed System Security Symposium, NDSS, 2011.

[2] Virrantaus, K., Markkula, J., Garmash, A., Terziyan, Y.V., 2001. Developing GIS-Supported Location-Based Services. In: Proc. of WGIS'2001 – First International Workshop on Web Geographical Information Systems., Kyoto, Japan. 423–432, 2001.

[3] Open Geospatial Consortium (OGC), Open LocationServices.www.nttdocomo.com/corebiz/network/index.html Internet information on mobile networks, 2005.

[4] S.Mascetti, Claudio Bettini, Dario Freni, X. Sean Wang, X. Sean Wang, Sushil Jajodia. Privacy-Aware Proximity Based Services. Tenth International Conference on Mobile Data Management: Systems, Services and Middleware, 2009.

[5] J.Dam Nielsen, Jakob Illeborg Pagter, and Michael Bladt Stausholm. Location Privacy via Actively Secure Private Proximity Testing. Fourth International Workshop on SECurity and SOCial Networking, Lugano, 19 March 2012

[6] C. Steinfield. The Development of Location Based Services in Mobile Commerce. Technology Management for Reshaping the World. Portland International Conference, 2004.

[7] L. Barkuus, and Anind Dey. Location-Based Services for Mobile Telephony: a Study of Users' Privacy Concerns. 9TH IFIP TC13 International Conference on Human-Computer Interaction, 2003.

[8] FTC report recommends privacy practices for mobile platforms, developers, advertisers. http://www.techjournal.org/2013/02/ftc-report recommends privacy-practices-for-mobile-platfor ms developers-advertisers/, Feb 4th, 2013.

[9] S. Steiniger, Moritz Neun and Alistair Edwardes. Foundations of Location Based Services, 2006.

[10] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In Proc. of Eurocrypt' 04, pages 1–19. Springer-Verlag, 2004.

[11] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D.Starin. Persona: an online social network with user defined privacy. SIGCOMM Computer. Communication. Rev.,39(4):135–146, 2009.

[12] Brimicombe, A.J. (2009) GIS, Environmental Modeling and Engineering (2nd Edition) . CRC Press, Boca Raton, FL, USA. Proceedings GIS, Bahrain. 33-45, 2009

[13] Shiode, N., Li, C., Batty, M., Longley, P., Maguire, The impact and penetration of Location Based Services. In: Karimi, H. A., Hammad, A., ed. Telegeoinformatics. CRC Press, 349-366, 2004

[14] Sabine Ehlers. Mobile Proximity Services a VAS Research Series, research report from Berg Insight, publication date, January 2008.

[15] Mike Hamburg, Joint work with Arvind Narayanan, Narendran Thiagarajan, Mugdha Lakhani, Dan Boneh Location Services with Built--In Privacy, 2011

[16] Pravin Shankar, Yun-Wu Huang, Paul Castro, Badri Nath, Liviu Iftode. Crowds replace Experts: Building Better Location-based Services using Mobile Social Network Interactions, 2012

[17] D. Qiu, D. Boneh, S. Lo, and P. Enge. Robust location tag generation from noisy location data for security applications. In The Institute of Navigation International Technical Meeting, 2009

[18] Emmanouil Magkos (Ionian University, Greece). Cryptographic Approaches for Privacy Preservation in Location-Based Services, 2011.

[19] Blog of Arvind Narayanan,http://33bits.org/2011/02/14/cryptographic-approach-location-privacy proximity -testing/

[20] Saroiu, S., & Wolman, A. (2009). Enabling new mobile applications with location proofs. In 10th Workshop on Mobile Computing Systems and Applications ACM, 2009

[21] Boneh, D., & Franklin, M. (2001). Identity-based encryption from the Weil pairing. In Advances in Cryptology - CRYPTO 2001 (pp. 213–229). Springer

[22] A. Juels and M. Sudan . A fuzzy vault scheme. Designs Codes and Crytpography, 237-257, 2006.

[23] C. Hazay and K. Nissim. Effiecient set operations in presence f malicious adversaries. In Proc. of public key crypto (PKC) , volume 6056 of LNCS, pages 312-331, 2010

[24] S. Jarecki and X. Liu. Efficient oblivious pseudorandom function with applications to adaptive to and secure computation of set intersection.

[25] C. Hazay and Y. Lindell. Effiecient protocols for set intersection and pattern matching with security against malicious and covert adversaries.

[26] Jeffrey Hightower and Gaetano Borriella. A survey and taxonomy of location systems for ubiquitous computing. IEEE Computer, 34(8):57–66, August 2001.

[27] H. Lipmaa. Verifiable homomorphic oblivious transfer and private equality test. In Proc. of Asiacrypt,, 2003.