SIP: Secure Information Provider Framework for Mobile Phone Users

Pallavi Arora and Huy Nguyen
Department of Computer Science
University of Houston, Houston, TX 77204
E-mail: *palpal@cs.uh.edu* and *nahuy@cs.uh.edu*

***Abstract*—The increasing power of the smart phone devices have posed many challenges to the information privacy of the device owner. Privacy has become of paramount importance with the explosion of increasingly popular (but malicious/curious) mobile phone apps. In this paper we provide a framework for preventing malicious services from accessing users data on his mobile phone without his permission. We show the proposed framework in action with the use of a music recommendation service. We evaluate the usability of the service in presence of our Anonymizer framework.**

I. Introduction

According to a study in August 2010, 55.7 million users owned a smart phone in USA and the rate of people buying a smart phone has gone upto 22% in 2011. The sales of Andriod phones have increased 900% from 2008-present. Similar trends have been observed for others. It can be accredited to the tremendous increase in the processing power of the smart phones, for example Samsung Galaxy can process 90 million triangles per second. Many factors have played a role in the increasing popularity of smart phones like large memory, bigger and better screens and open operating systems etc. Like every other powerful commodity, smartphones have had their fair share of controversy. With the recent fiasco of "mobile phones manufacturers tracking their consumers", mobile phone users have become concerened about their privacy and that of their data. Paul Wilson of Dallas said: "No way will a game have access to my contact list or call log. Next they'll want me to send them a key to my house so they can go through my bank and tax statements." The statement aptly describes the concerns of many of us. First we need to understand what harm can be caused by installing any malicious third party applications on our phone. Third party applications can access features like contacts list, location history, times of past meetings and future appointments, photographs and videos, access to camera (in some cases), details of who the user contacted and when, whether it was via voice, e-mail, SMS, IM, or social networking, often including a verbatim transcript of the message. Access to user's contact list can lead to lost friendships, missed business opportunities, even a ruined marriage. Access to user's appointment calendar could inadvertently disclose his medical condition. Similarly access to user's location data could let burglars know when he is away from home, could also reveal to pedophiles what route his children walk to school. Also can the data be used to determine user's behaviour and target repetitive advertisement to the user.

We propose a system in this paper that prevents the user from the malicious third party services without compromising the usability of the service. We propose a central trusted authority that acts as an anonymizer for the user and gives user full control of his data. We call this trusted server Secure Information Provider (SIP). We show how SIP server could function in practice. As we can observe that there exists a fine balance between the accuracy of the service and the privacy for the user. We used perturbation based cloaking mechanism to ensure user's privacy. Privacy is achieved by adding random noise to user's data thereby preventing the third party services to identify a particular user. We show the functioning of our system using a music recommendation service as a third party service. The novelty of the system lies in the fact that we remove the noise from the result obtained from third party service before handing back the results to the user. The paper is organised as follows, In section II we provide an overview of the systems where similar idea of anonymizer is used to prevent user privacy. In section III, we describe the scope of model. In section IV, we describe the attacker model. In section V, we describe the components of our system. In depth implementation details of the model are provided in the section VI. We give the evaluations in section VII, followed by the conclusion in VIII.

II. Related work

The anonymizing framework provided in this paper has been well studied in the field of privacy preserving location based services (LBS), like location-aware emergency response, location-based advertisement, and location-based entertainment or to infer private life of an individual [1], [2], [3], [4]. The goal of most of these papers is to protect the association between users identity, query sources (like k-nearest neighbors), servers and databases thus preventing the attacker from mapping users of LBS to certain locations. Anonymizing is done using a cloaking mechanism. Various cloaking mechanisms have been proposed in the literature. Cloaking mechanisms are based on the spatial/temporal cloaking [5] or locality-preserving Hilbert Curve [6]. Cloaking meachnasims are developed to provide k-anonymity, l-diveristy and s-diversity. k-anonymity cloaked spatial area thus makes it harder for the attacker to identify the user by making k-

users share the same spatial boundary. l-diversity, [2], was devised to overcome the drawback of k-anonymity by adding another dimension to privacy by enforcing l-different locations in a region. In [4], s-diversity was introduced to provide privacy by imposing a constraint that there are at least $s(> 1)$ different road segments associated to the location upon release. The challenges in this field lie in the detection of various location privacy vulnerabilities and development of privacy threat models then develop the corresponding defense methods to counter the deficit regardless of population density.

III. Applicability

We show the functioning of our system on a music recommendation service but similar concept is readily applicable to services like reddit (Social news website) and Netflix movie recommendation etc. We can call these services Noise tolerant as the user is indifferent to low noise levels in the results of these services. The anonymizer could also be used for homomorphic encryption of the data.

IV. Attacker model

We assume the third party services are *Honest but curious*, also we assume that they do not collude. An *Honest but curious* adversary model assumes that no information is gained by the attacker apart from what he can learn from the results of the protocol i.e all parties are curious, in that they try to find out as much as possible about other inputs despite following the protocol. As is clear that if we use the perturbation based model the third party services can not learn the identity of the user.

V. System architecture

The proposed system architecture consists of three main components, namely: the mobile client, the SIP gateway, and the third party services. In this section, we will gradually present each one of them with details and functionalities. An overview of the entire system is illustrated in Figure 1.

A. Mobile client

Mobile clients are devices that are operated by end-users, whose identity needs to be protected. In this paper the term *mobile client* and *end-user* will be used interchangeably. Consider a system where end-users are not suppose to trust any third party service since their identity can be inferred from the exposed information if we consider the attacker model mentioned in the previous section. End-users now are not willing to give out their information directly to the services anymore. Instead, they will choose to "subscribe" securely to their desired service through a middle man whom they trust.

B. SIP gateway

SIP gateway will be the aforementioned middle man who handles private user information, help them connect to the outer world (i.e. third party services) in a safe manner so that no malicious attacker could infer users' identity. Users will have to register themselves to the gateway and provide the gateway with necessary information. All user information will then be stored in a relational database which helps faster data query processing. When sending user information to the third party services, SIP gateway will perturb the original information with a portion of noise data, and then send that perturbed information to the services. Beside providing users the cloaking ability, SIP gateway can also detect malicious services based on the way they access user information. All transmitted data will be logged and anomaly detection algorithms could be applied on the logs to detect any kind of malicious activities. The SIP gateway itself contains two components with different function: the SIP server and the SIP web portal.

1) SIP server: SIP server is the main component of SIP gateway which handles all connections between end-users and third party services, and provide users the cloaking ability. In this work, we make the following assumptions: 1) the SIP server is secure and can't be hacked by attackers, 2) SIP server is constantly available so that the service won't be disrupted, and 3) the connection between clients and server in which clients send their private information is not compromised or eavesdropped. There are many ways by which we can ensure the assumptions hold. To protect the server against attackers, a firewall is a common solution that will ensure the server is secured and available. Intrusion and anomaly detection algorithms are to be deployed on the firewall to detect any efforts to compromise the server. When clients send out their data to the server, the data can either be encrypted or sent using a secure protocol (e.g. using https protocol instead of normal http) to ensure the information is not leaked.

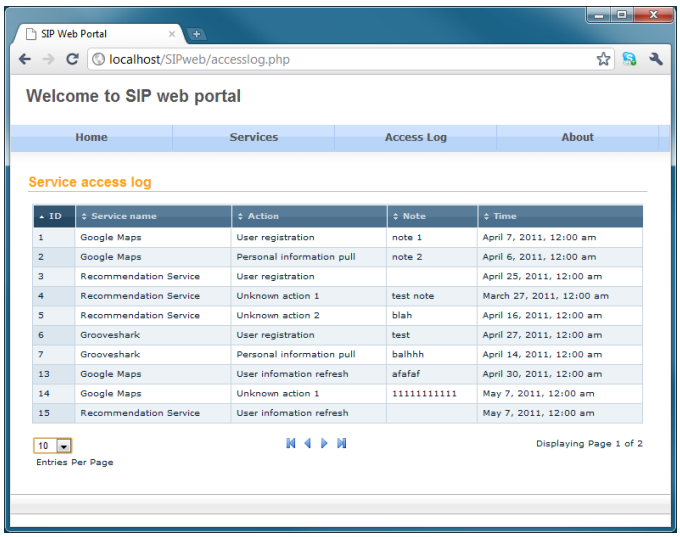

Fig. 2: Access log screenshot

2) SIP web portal: SIP web portal provides users another way to manage all their service subscriptions and personal information rather than having to use their phone. Users can access the web portal from anywhere just with a web browser. Besides, web portal also provides users access to the information access log (Figure 2) which records in detail

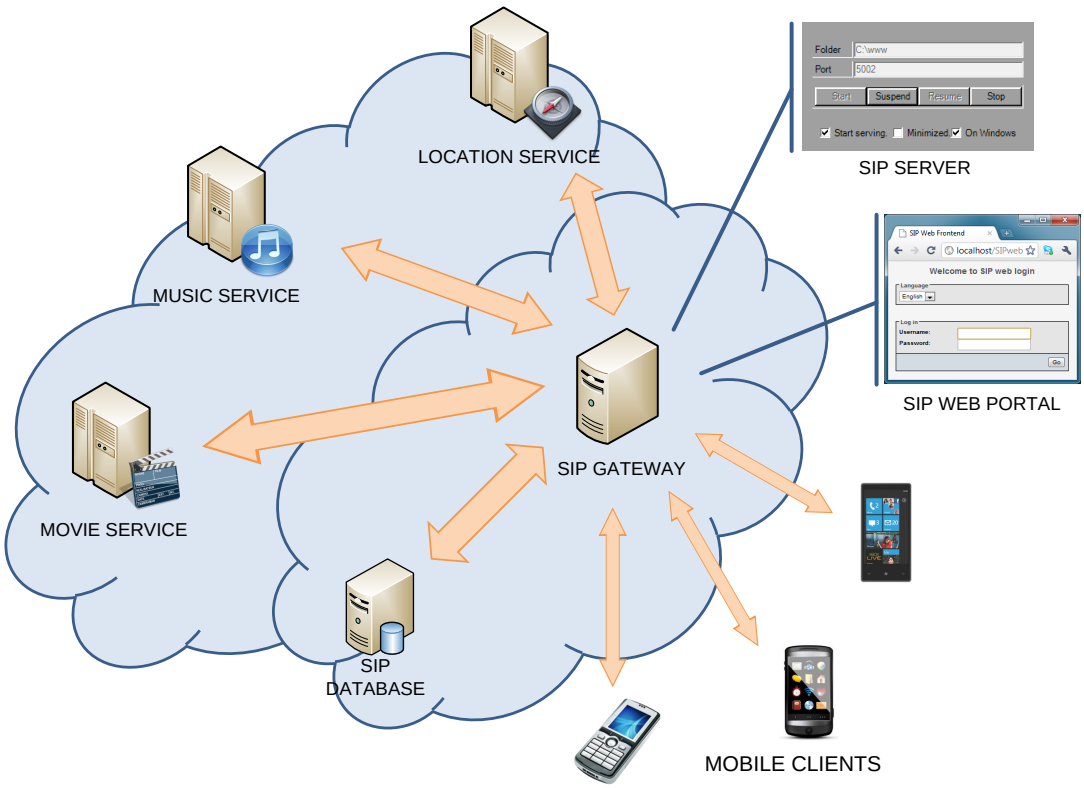

Fig. 1: SIP system overview. The system is divided into three layers: mobile clients, SIP gateway and third party services.

which service accesses the user information and why they did so. The access list will be significantly helpful for users to keep track of recent activities relating to their privacy. It is also feasible for the SIP server to apply data mining techniques on the access log to help users automatically detect suspicious patterns of activities.

C. Third party services

Third party services are (mostly) information service providers from whom the end-users consume the data. Common types of information services are location-based services (Google Maps, Groupoun), movie services (Netflix, youtube), music services (last.fm, grooveshark, Pandora), social networks (Facebook, myspace, Twitter)... Currently, users who want to use one of the aforementioned services have to subscribe to each of them and provide them with their personal information. This exposes a security risk if one of the services is compromised and become malicious, users will face the threat of having their privacy leaked. In our model, third party services can only provide their service to "anonymous" user requests sent from SIP gateway. This way, they can provide user their services without the ability to compromise user privacy.

There is, however, a tradeoff between user anonymity and the quality of service. In other words, let consider the location-based service of [7] where real user's location are hidden by just exposing an approximate area where the user can be located rather than the user's exact location. This approach hurts the quality of the location recommendation service since the service can not perform well without the accurate user location. Challenging that tradeoff is a non-trivial problem and is our motivation in this research.

VI. Implementation

Many different implementations of the SIP model are possible. The right choice depends on the environment and what kind of services the users are being protected from. In this section, we describe our SIP implementation targeted to cloak the end-users from music recommendation services. Note that even though we are implementing a music service, the proposed model can be applied to other kinds of services as well.

In the first part of this section, we will first introduce our cloaking mechanism that is implemented on SIP server to protect user identity. The second part of the section will present details on how we implement the system.

A. Cloaking mechanism

As previously mentioned, directly exposing user song list to the service could entails user privacy violation. Therefore we need SIP server in the middle to perturb user song list in such a way that the user can still use the recommendation service. The whole cloaking process is illustrated in Figure 3.

In our scenario, each user has his own song list L and would like to use the music recommendation to get a list of similar songs that he might be interested in. SIP server perform the

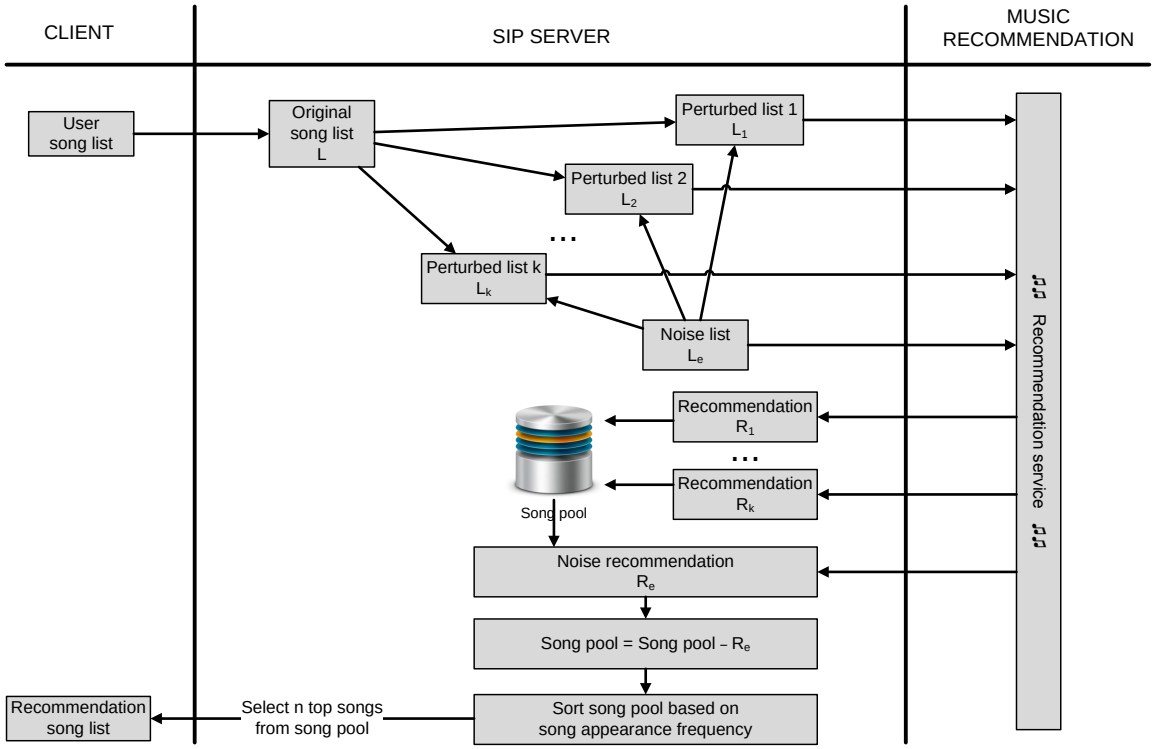

Fig. 3: SIP cloaking mechanism.

cloaking service on the user identity by first generate a list of irrelevant songs L_e. Then from the user song list L and the noise list L_e, k perturbed lists are generated $L_1, L_2, \ldots, L_k$ are generated by picking n_e percent of random songs from L_e and $(1 - n_e)$ percent of random songs from L. n_e can be thought of as a "noise-level" by which we generate the perturbed lists. Intuitively, increasing n_e or reducing k, has the benefit of less user information exposed, but will affect the quality of the recommendation service. We will perform an intensive study in section VII to verify whether this conception holds in practice.

Now that the server sent $k + 1$ requests to the recommendation service, k lists of songs $R_1, R_2, \ldots, R_k$ are received according to each perturbed list and one corresponds to the noise list, we call it the "noise recommendation" list R_e. SIP server will then build the song pool of recommended songs from $R_1, R_2, \ldots, R_k$. The noise recommendation list R_e indicates irrelevant recommended songs and will therefore be removed from the song pool. This step ensures the song pool only contains songs that are suitable to be included in the final recommendation list that is sent back to the user.

The song pool, in another hand, has many overlapped songs since similar recommendations can be suggested for similar songs in k perturbed lists. SIP server sorts the song pool based on their frequency of appearance, give more weight to songs that appears many times since they have more chance to be songs on which the user is interested. n top songs are then selected to be included in the final recommendation list and sent to the user.

B. System implementation

1) Mobile client: For the illustration purpose, we implemented an application on Windows Phone 7 that has the following functionalities: user registration, third party services selection, subscription and unsubscription, get the recommended song list, processing the list, save songs from the recommendation list on the phone's storage based on user's interest. We use asynchronous http calls to send and receive XML messages (Figure 4) between the phone and SIP server.

From the recommendation list, user can choose to save their favorite songs on the phone's local storage to playback later. The next time users use the music recommendation service, these songs will also be sent along with other songs that are currently in user's song list.

2) SIP gateway: SIP server is a simple HTTP server that constantly listen to connections from clients and third party services on port 5002. We implemented SIP server using C#. SIP web portal is implemented with PHP 5.2.14 and hosted by the server running IIS 7.0. We also use AJAX technique to code the access log, which improves user interaction experience with the system. Also, AJAX helps speed up the system process since the server doesn't have to load more data than asked by the user. Both SIP server and SIP web portal share the same relational database. In our implementation, we use SQL Azure cloud service from Microsoft to store data for both the server and the web portal. In order to ensure that SIP server can handle many user connections, we program the server to

```
<sipmessage>
   <type>usergetrecommendedsongs</type>
   <size>3</size>
   <song>
      <songID>1</songID>
      <songName>Larger than Life</songName>
      <artistName>Backstreet Boys</artistName>
      <album>Millennium</album>
      <genre>pop</genre>
      <status>1</status>
      <ownerID>1</ownerID>
   </song>
   <song>
      // more songs ...
   </song>
</sipmessage>
```

Fig. 4: XML message from client to get recommended song list. Details of each song on the client will be encapsulated in a *song* tag

generate a system thread to handle each user request. The thread will be terminated to free up system resource as soon as user request is complete.

3) Music recommendation service: Similar to SIP server, we build a music recommendation service that connects to SIP server through the port 5003. To make the recommendation list more realistic, and to also allow us to evaluate the system later on, we use last.fm APIs [8] to pull the song list, artist info and recommendation songs. This will benefits us in two ways: 1) allow us to have complete control over the third party service, and 2) the service operates just like a real one. The only drawback is the connection quality to last.fm server is sometimes slow and unreliable.

VII. Evaluation

In this section, we first present our metric to evaluate the satisfaction degree of user with the recommended song list. Then we conduct three sets of experiments to evaluate performance of the proposed system under different settings. The song database that is used in our experiment sets are from last.fm.

A. Song similarity index

In order to evaluate how good are the recommended result, we propose a new metric for estimating the degree of similarity between two songs. If we have two songs s_1 and s_2, artist of s_1 and s_2 are a_1 and a_2, respectively, then song similarity index is defined as:

$$SI(s_1, s_2) = (a + b)w + \frac{\gamma}{\alpha + \beta + \gamma}, \quad (1)$$

where $a = 1$ if s_1 and s_2 belong to a same album, and $b = 1$ if s_1 and s_2 have a same genre; a and b are equal to 0 otherwise. $(a + b)$ thus can be though of as the similarity between two songs themselves. w is a weight factor such that $0 \leq w \leq 1$. The second part of the equation calculates the similarity between two artists with α is the number of genres of a_1 which a_2 does not have, β is the number of genres of a_2 which a_1 does not have, and γ is the number of common genres of both a_1 and a_2.

Essentially, if two songs have a high similarity index, one is most likely to be recommended from the other. In other words, if $SI(s_1, s_2)$ is high and $s_1 \neq s_2$, then if the user send s_1 to the recommendation service, the probability that recommended result contains s_2 is high.

From (1), we obtain the normalized similarity index between two song lists by computing the mean SI between each song in two lists. This metric can be justified as the satisfaction degree of user on the recommended song list.

B. Simulations

1) Weight factor: In the first set of experiment, we vary the weight factor w from 0 to 0.5 in the equation. $w = 0$ corresponds to the case when we only consider the similarity between two songs based on their artists. We run the experiment 10 times, and in each time, a random set of songs are selected to be the user song list. This song list will be sent to SIP server to perform the cloaking process before arriving at the recommendation service. We measure the satisfaction degree of user on the final recommendation result in two cases, with and without SIP server. Note that user privacy is at risk in the case there is no SIP server. As depicted in Figure 5, changing w does not affect performance of the proposed method. Satisfaction degree of user on the recommendation result in two cases are similar, proving that SIP *does not* affect the quality of service.

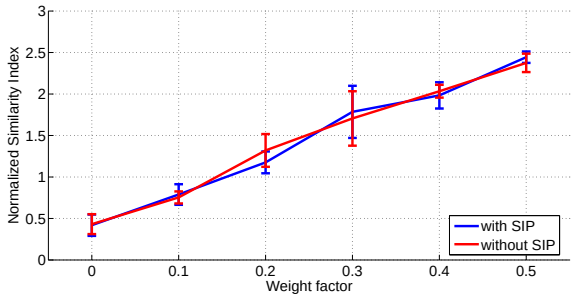

Fig. 5: Normalized similarity index with varying weight factor. Error bars are symmetric and indicates standard deviation of 10 runs with different seeds.

2) Perturbed list quantity: The second set of experiment varies the number of perturbed list from 5 to 10. w is fixed at 0.2 and noise level is fixed at 20%. Generating more perturbed lists increase the user happiness, however, it entails the fact that more information on user are exposed to third party services. The perturbed list therefore should not be too small, not too large. In future implementation, we would like to consider the number of perturbed list as a function which is proportional to the system noise level.

3) Noise level: The final experiment set studies the effect of noise on the system performance. Number of perturbed lists

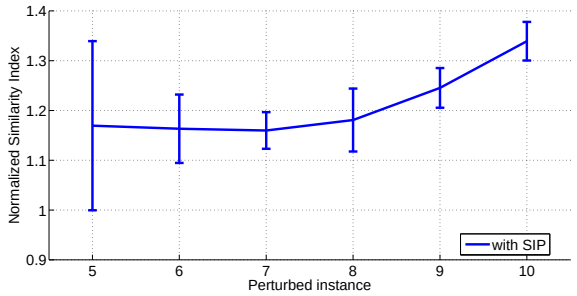

Fig. 6: Normalized similarity index with varying quantity of perturbed lists.

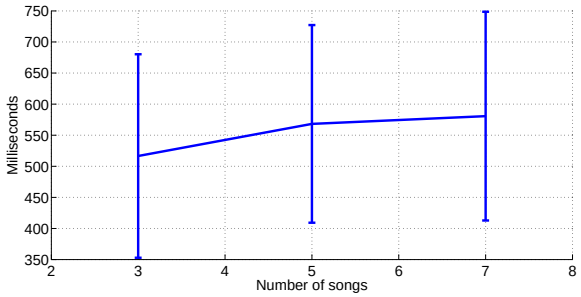

Fig. 8: Time taken to get the recommendation service list for the emulator.

is fixed at 5 and w is fixed at 0.2. It is clear that more noise decrease the system performance. However, it is noteworthy to see that as noise increases from 0 to 80%, user happiness only decrease by 10%. This proves our approach to be high noise tolerant.

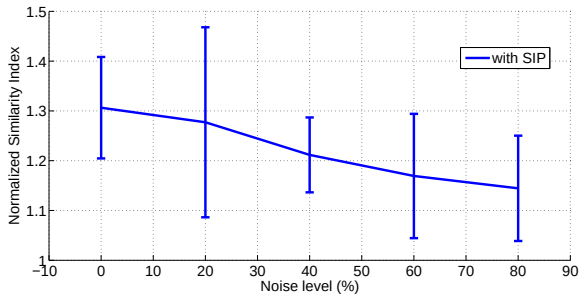

Fig. 7: Normalized similarity index with varying level of noise.

C. Device Experiments

We carried out the experiments to test the time taken for obtaining the recommendation service list from our server from the Windows phone 7 (WP7) emulator and the device. We used speedtest.net to obtain the bandwidth of the UH wireless for the computer and used an application called bandwidth on WP7 to obtian the bandwidth for the device. The resuls are as follows

	Download	Upload
Emulator	8.46Mb/s	2.80Mb/s
Device	10.40Mb/s	9.20Mb/s

1) Emulator: We tested the emulator speed for getting the songs with 3, 5 and 7 songs. We took the average of 15 runs. The result is shown in Figure 8.

2) Device: We tested the response time for the server on the device for 15 runs. Test was conducted with 4 songs on the device. The mean time was 652.57ms and the standard deviation was 326.23ms. Higher standard deviation could be the result of network fluctuation. Demo video of the system working on the real device can be found at http://www2.cs.uh.edu/ nahuy/data/SIPmovie.wmv.

VIII. Conclusion

In this research, we proposed SIP, a secure framework that can protect mobile phone users' identity without having to scarify quality of the third party services. The methodology is simple and can be applied on many kinds of services. Extensive simulation results showed not only that the system can maintain high user satisfaction degree, it is also resilient to noise. For future direction, we would like to expand the application pool by experiencing more types of services. It is also interesting to derive rigorous mathematical performance bounds of the proposed system.

References

[1] L. Sweene, "k-anonymity: A model for protecting privacy." *International Journal of Uncertainty,Fuzziness and Knowledge-Based Systems*, vol. 10(5), no. 557570, 2002.

[2] L. Liu, "From data privacy to location privacy: Models and algorithms," *VLDB*, 2007.

[3] L. Ling, "Privacy and location anonymization in location-based services," *SIGSPATIAL Special*, vol. 1, pp. 15–22, July 2009.

[4] T. Wang and L. Liu, "Privacy-aware mobile services over road networks," *VLDB*, 2009.

[5] B. Gedik and L. Liu, "Location privacy in mobile systems: A personalized anonymization model," in *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, ser. ICDCS '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 620–629. [Online]. Available: http://dx.doi.org/10.1109/ICDCS.2005.48

[6] H. Samet, *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[7] M. Gruteser, D. Grunwalddepartment, and C. Science, "Anonymous usage of location-based services through spatial and temporal cloaking," 2003, pp. 31–42.

[8] "Last.fm web services." [Online]. Available: http://www.last.fm/api