DiscoverFriends: Secure Communication with OSN Friends in Mobile Ad Hoc Networks

Eric Chung
Department of Computer Science
University of California at Los Angeles
echung@cs.ucla.edu

***Abstract*—This paper presents a secure communication application called DiscoverFriends. Its purpose is to communicate to a group of online friends while bypassing their respective social networking servers under a mobile ad hoc network environment. DiscoverFriends leverages Bloom filters and a hybrid encryption technique with a self-organized public-key management scheme to securely identify friends and provide authentication. Firstly, Bloom filters provide a space-efficient means of security for friend discovery. Secondly, a combination of asymmetric and symmetric encryptions algorithms utilizes the benefits of both to provide increased security at lower computational cost. Thirdly, a self-organized public-key management scheme helps authenticate users using a trust graph in an infrastructureless setting. With the use of Wi-Fi Direct technology, an initiator is able to establish an ad hoc network where friends can connect to within the application. DiscoverFriends was analyzed under two threat models: replay attacks and eavesdropping by a common friend. Finally, the paper evaluates the application based on storage usage and processing.**

***Index Terms*—Mobile ad hoc networks (MANET), bloom filter, key management**

I. Introduction

Mobile devices, such as smartphones and tablets, have become a rising platform for Online Social Networks (OSNs). Facebook, for example, reported over 1 billion mobile users [1] in the first quarter of 2014. Why are so many users moving to the mobile platform? One possible reason is that users can have instantaneous access to OSN with mobile devices at any time and place. But more importantly, users are attracted by the various interesting features that are only available on mobile platforms. Location-based services (LBS) [2] is one of the top features exploited by mobile users.

With GPS integrated into smartphones, mobile OSN users can easily localize themselves, which allows them to share their own location and related experiences with friends (e.g., the "Check-in" feature of the Facebook mobile app). Since it does so, users are more able to find nearby friends based on their locations. Therefore, the LBS functionality provides great convenience for users' social activities.

However, the application of LBS in OSNs also incurs some concerns surrounding the issue of privacy. When a user activates the LBS function in an OSN application, it exposes the user's location to the OSN service provider. In other words, Facebook can track the user's activity with this location information. One question is: Can the OSN users still utilize partial LBS without reporting their location to OSN providers? Typically, can we find nearby friends without requesting them from the Facebook servers? This project provides a solution to this problem.

In this work, a Wi-Fi-based solution called DiscoverFriends helps an OSN user find nearby friends. The core idea is to leverage the local Wi-Fi communication to directly discover an OSN friend while bypassing OSN servers. The design faces two main challenges. The first problem is how to authenticate an OSN user without going to the OSN server. Second, due to the broadcast nature of wireless communication, the messages can be overheard by other users. The potential eavesdropping can be a new threat to the privacy. To this end, DiscoverFriends addresses these issues using a Bloom filter-based approach with hybrid encryption to provide a higher level of security.

The solution provided by DiscoverFriends is not only limited to discovering nearby friends. It can further be employed to setup communication between friends in the same local Wi-Fi. In other words, OSN users can exchange text messages and other data without going through the OSN server, which further helps to prevent users from being tracked by those OSN providers.

This paper is organized as follows. Section 2 begins by elaborating on the application design followed by implementation details presented in Section 3. Section 4 describes the risk analysis under two threat models. Performance evaluation in Section 5 is done against alternative schemes described in the related works in Section 6. Lastly, Section 7 summarizes the features of the application and concludes the paper.

II. Design

In DiscoverFriends, any OSN user can serve as an $Initiator$ and leverage Wi-Fi broadcast to send request messages in order to find a specified friend, hereby referred to as $Target$. If the target receives the request, the target will then send back an acknowledgement to the initiator. The request message is constructed using Bloom filters and a certificate to achieve confidentiality and authentication, respectively.

The solution proposed is built based on an important assumption: each OSN user has a confidential ID, which is only accessible to the user's friends. This ID is not necessarily the string that OSNs use to identify each user, but it can be an extra confidential string. For instance, Facebook provides a user with a public username and private ID. This assumption

is reasonable since many OSNs have access control mechanisms, in which users can specify the accessibility of private information (e.g., whether it is public to all users or friends only).

Furthermore, DiscoverFriends is not limited to supporting only one OSN as it can utilize IDs from multiple OSNs to improve security. More specifically, users can XOR their IDs in Facebook and Google+ to generate a new ID. This new ID cannot be recognized by any single OSN provider even if the provider somehow captures the message. One drawback of this technique is that the intended friend has to be the initiator's friend on multiple OSNs, which may limit the usage of the application.

In this section, the application of Bloom filters is introduced and then explained how they are applied in DiscoverFriends. In addition to that, two encryption mechanisms are analyzed and a combined solution is selected to be used in the application. Moreover, an overview of key management techniques is provided, where one scheme is selected to be used in the application. Finally, the communication protocol is examined between the initiator and the target.

A. Bloom Filter

Bloom filter is a space-efficient probabilistic data structure, whose purpose is to test whether a specific element is in a given set or not. It may produce false positive results but will definitely not return false negative results. Bloom filter is widely used in caching mechanisms as well as security solutions.

Bloom filter is simply an m-bit array. When an element is added into a Bloom filter, k hash function is applied on the element to get k hash values in the range of [0, m). Then, the corresponding bits in the array are set to 1. When testing for an element, the same set of hash functions are applied before checking to see whether the corresponding bits in the Bloom filter are set to 1. If all of them are 1, then it means that the given element is included in the set; otherwise, it is excluded in the set. This explanation can be shown in Figure 1.

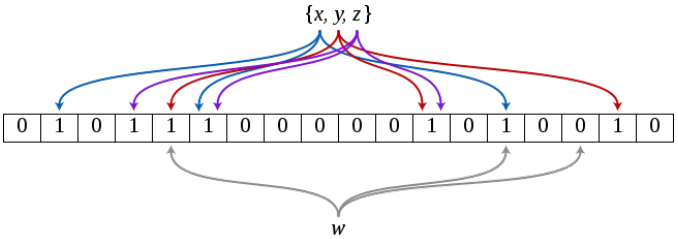


Figure 1. Bloom filter representation

When designing the Bloom filter, one focus is to optimize the number of hash functions k and the length of the Bloom filter m, which can be calculated as a function of n, the number of inserted items, and p, the desired false positive probability as shown in Equation 1. In the case of DiscoverFriends, n is the number of targeted friends the initiator wants to communicate to. Also, the choice of a small p value is necessary as it corresponds to the probability that an attacker may be able to ascertain the hash of the initiator's ID. Subsequently, the appropriate k value can be computed using m and n using Equation 2.

$$m = -\frac{n \ln p}{(\ln 2)^2} \tag{1}$$

$$k = \frac{m}{n} \ln 2 \tag{2}$$

In DiscoverFriends, Bloom filters serve as the container of the IDs for friends. More specifically, when the initiator, who will be called Alice, sends the initial request, she adds the target's ID into a Bloom filter, which will be sent during the network initialization phase for friend discovery. Thus, when the target, say Bob, gets the request, he can determine whether or not the request is directed to him. In addition, Alice will also include her own ID in another Bloom filter carried in the request, so the target can also figure out who the initiator is.

By using this technique, OSN users are not required to put their confidential ID into the broadcast message, which can be listened to by other people. Therefore, even if an adversary, Eve, eavesdrops the request message, it is extremely difficult for her to infer the initiator's ID if she is not the initiator's friend because the adversary only knows the hash values and cannot acquire the accurate corresponding ID with the hash values, let alone understand the user corresponding to that ID. However, it is important to note that although the use of Bloom filters can speed up initiator ID discovery and reduce latency, it makes it easier for non-friends to discover the ID. Thus, security is traded off at the expense of lower discovery latency.

B. Encryption Mechanisms

When designing DiscoverFriends, the idea was to optimize over computational cost, storage cost, and security. Here, computational cost is an important issue under a mobile setting due to energy consumption from calls to encrypt and decrypt data. Asymmetric encryption algorithms such as RSA, PGP, and GPG grant higher security at the cost of processing power. On the other hand, symmetric encryption algorithms such as AES, Blowfish, DES, and 3DES provide lower security at a lower computational cost and storage cost.

In order to achieve the best of both worlds, DiscoverFriends uses a combination of asymmetric and symmetric algorithms. The general idea is to use a symmetric key to encrypt data and then encrypt that key asymmetrically. This message is then transmitted to a destination node along with the encrypted key. Because the message is encrypted using a symmetric algorithm, the computational cost to decrypt the message is less while at the same time, the encrypted key produced using an asymmetric algorithm provides higher security. For the symmetric algorithm, AES is chosen as it outperforms the other algorithms in terms of throughput in MANETs as shown in Table I [3].

Table I
THROUGHPUT IN MB/SEC

THROUGH-PUT	TEXT	IMAGE	AUDIO	VIDEO
DES	10.616	9.326	10.01	11.16
3DES (112 bit key)	3.875	3.635	3.883	3.909
3DES (168 bit key)	3.885	3.802	3.872	3.953
AES	23.503	20.504	22.099	27.447
BLOWFISH	17.64	15.328	17.094	19.602

C. Digital Signature and Key Management

To ensure the security of the application, the idea was to use certificates for authentication. In order to generate a public key certificate, two schemes exist: public key infrastructure (PKI) and web of trust. In PKI systems, a certificate authority (CA) issues certificates that binds public keys to identities, and this information is kept in a central repository. On the other hand, a web of trust entails identity-based cryptography through self-signed certificates, where a key generation center (KGC) generates users' private keys. Evidently, this scheme is highly susceptible to man-in-the-middle attacks.

However, things are not that simple in an infrastructure-less model. Because of the ad hoc nature of DiscoverFriends, conventional public key infrastructure and web of trust schemes are less suitable for the application due to complicated certificate management [4]. Also, certificate authorization becomes impractical. The problem becomes clear when nodes cannot guarantee their connectivity and cannot rely on infrastructure to detect compromised mobile nodes [5]. In addition, certificate revocation in MANETs [6] becomes an issue because of the lack of centralized repositories and trusted authorities.

There are two main types of public-key cryptography: certificate-based cryptography (CBC) and ID-based cryptography (IBC). Previous works [7] have attempted to use CBC in MANETs. A naive approach for key distribution occurs in the network setup phase, where every node is preloaded with all other's public key certificates. However, as mentioned above, key management becomes an issue as key updates need to be handled in a cost-effective manner [8]. As a result, an improvement [9] using self-organized public-key management was built on the idea of a certificate chain. In a different angle, IBC eliminates the need for public-key distribution and certificates altogether by using a public key based off a public string identifying an individual. The underlying idea of IBC-based certificateless public-key management schemes is to have a set of network nodes share a master-key generated by threshold cryptography and collaboratively issue ID-based private keys. Built on top of the underlying concept, variations on the designing of certificateless public keys have come about in recent literature [10], [11], [12]. The downside of IBC is if the threshold number of nodes gets compromised, the network is breached.

DiscoverFriends uses a self-organized public-key management approach. Further details can be read up on in [13]. Every node in the network contains the following:

- **Key repository (KR)**: Stores public keys sent by neighbor node.
- **Shared key repository (SKR)**: Stores public keys of all nodes from the KR.
- **Certificate repository (CR)**: Stores valid self-signed certificates.

In order to assess the security in the formation of trust, the network initialization phase plays an important role. The main objective of this phase is to distribute all the public keys to every node in the network. Instead of a certificate graph, a trust graph (TG) is generated by the shared key repository per node. At the end of the initialization phase, this trust graph is stored as a master graph (MG), which facilitates frequent public key updates.

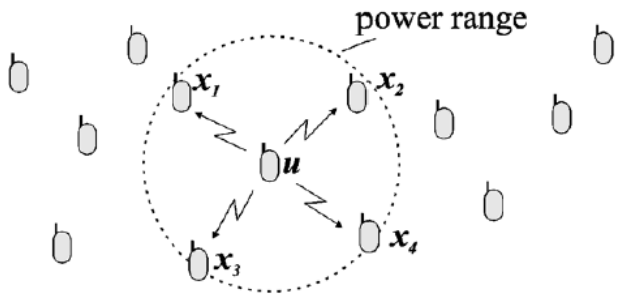


Figure 2. Architecture

In its reduced form, DiscoverFriends' model can be simplified as target nodes do not serve as intermediary nodes for other target nodes as shown in Figure 2. However, during prolonged application usage, it may be energy-efficient [14], [15] to pass group ownership to a target node (i.e. x1) once the secure environment has been set up. This sets the need for an efficient key management scheme specifically designed for MANETs. In this case, the target node must know the certificate and public key associated with all nodes within its group. Therefore, the TG provides a path to the end node, which has the corresponding private key to decrypt the sender's message. Within the data structures of each node, valid certificates and public keys of all other nodes in the group are contained. The transmission of the information is done during the network initialization phase, which in the case of DiscoverFriends, happens between friend discovery and communicating to the group of nodes.

D. Communication Protocol

The communication protocol consists of two parties: $Initiator$ and $Target$. The first two stages represent the network initialization phase. In the first stage, the initiator, who wants to discover a certain friend (target), sends a Wi-Fi broadcast request consisting of three parts:

- BF_c: The Bloom filter containing the $Target$'s ID.
- BF_c+: $BF_c \oplus hash\,of\,Initiator's\,ID$.
- CF: $Initiator$'s certificate encrypted with AES; the encryption key is the hash of the $Initiator$'s ID.

Here, BF_c is used to identify whether the target is a person of interest in the invitation. BF_c+ builds on top of this knowledge and helps the target identify the initiator. Lastly, CF is used for authentication without going through the OSN server. In detail, the certificate contained inside is self-signed by the initiator.

In the second stage of the protocol, the target receives the request message and proceeds in the following steps:

1) Test whether the $Target$'s own ID is in BF_c . If not, terminate the procedure. Otherwise, proceed to next step.
2) $BF_c \oplus BF_c+ \rightarrow hash\,of\,Initiator's\,ID$.
3) Traverse the $Target$'s own friend list and apply has functions to each friend's ID to determine if a match with the $Initiator$'s ID exists.
4) Once a match has been found, take the hash of $Initiator$'s ID to decrypt CF.
5) Check the validity of the certificate in CF and finish authentication.
6) If the $Target$ accepts the invitation from the $Initiator$, the $Target$ replies with own certificate encrypted with AES; the encryption key is the hash of the $Initiator$'s ID.

In stage three, once all peers are connected, the $Initiator$ sends the set of certificates to the connected peers. Step 6 and stage three are necessary for the $Initiator$ to update the SKR and CR of other targets that are connected to the $Initiator$ such that after the network initialization phase, proper key management is ensured and eavesdropping attempts are averted. Assuming all targets who want to connect to the $Initiator$ are connected, the network initialization phase ends. This leads into stage four and stage five, which represent subsequent communication. Optionally, step 6 is repeated as many times as necessary during network connection in order to push certificate updates to the group as certificates may expire during the communication session. These two stages may be repeated multiple times until the connection is broken. In stage four, the $Initiator$ communicates to the $Target$ as follows:

1) Generates a random symmetric encryption key.
2) Uses symmetric key to encrypt a message.
3) Encrypts symmetric key using own public key.
4) Broadcasts the encrypted message and encrypted key.

In response, the $Target$ replies in the following manner:

1) Uses own private key to decrypt the symmetric key.
2) Uses symmetric key to decrypt the message.
3) Sends back an AES-encrypted message to $Initiator$.

Now, the initiator and target are able to communicate directly, effectively bypassing OSN servers. The packet data for the communication protocol can be shown in Figure 3.

III. Implementation

In this section, details of the implementation are provided, including the Bloom filter, Wi-Fi communication, and OSN

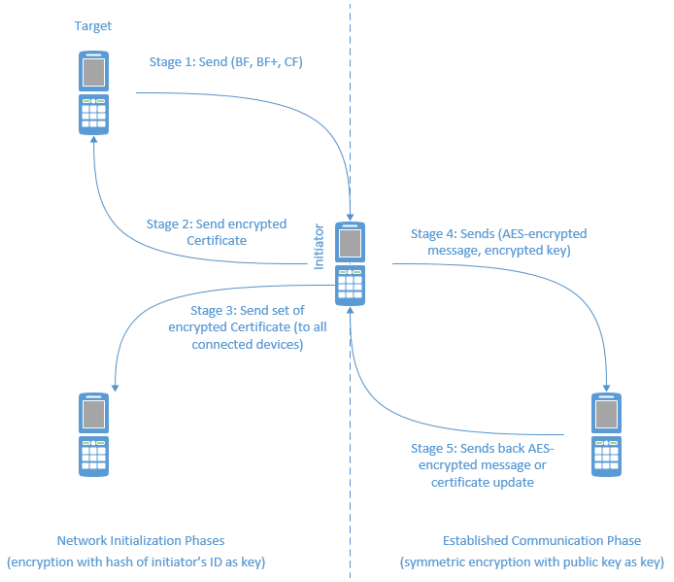


Figure 3. Communication Protocol

adaptors. The simple interface for the DiscoverFriends is shown in Figure 4a, 4b, and 4c. The application has features for turning on Wi-Fi Direct, discovering nearby peers, creating a group, connecting to OSNs, and querying the list of friends from OSN servers.

A. Bloom Filter

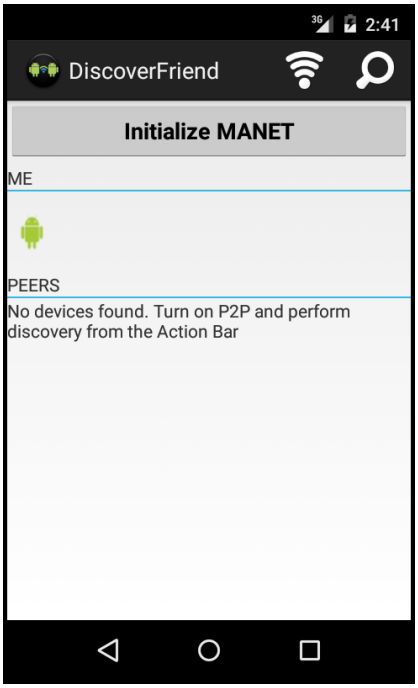
One important issue of implementing Bloom filters is the usage of the hash function. There are several candidates, including murmur, fvn series of hashes, Jenkins Hashes, etc. Not only should the ideal hash function provide independent and uniformly distributed results, it should also be as fast as possible to account for the mobile platforms' limited computation capabilities. Therefore, the final choice should preferably avoid widely used hashing algorithms, which generally run slow. After considering different factors, murmur32 was selected as the prime hash function for DiscoverFriends and the parameter to reduce the false positive probability was tuned to 2%.

B. Encryption

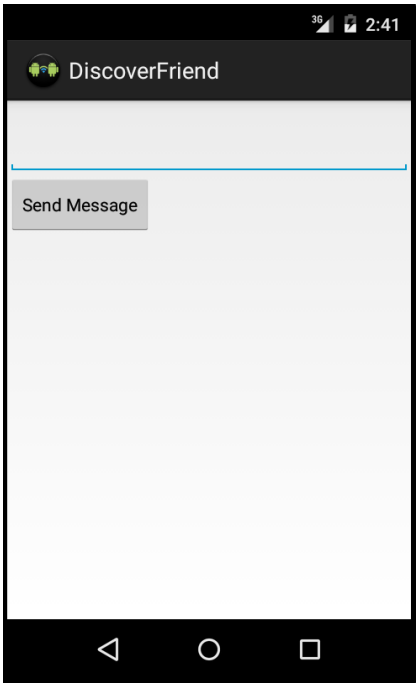
For symmetric key encryption, DiscoverFriends uses AES encryption. In the network initialization phase, the secret key is the SHA-1 hash of the initiator's ID, trimmed to use only the first 128 bits. During subsequent message exchanges, the secret key is randomly generated using SHA1PRNG. Also, for this scheme, the cipher uses a PKCS5Padding transformation, which supports AES 128-bit keys.

As for asymmetric cryptography, a public/private key pair is generated using a RSA algorithm, in which key sizes are 1024 bits. With this key pair, DiscoverFriends generates self-signed X.509 certificates using the sun.security.x509 package. An alternative library was to ues Bouncy Castle X509V*CertificateGenerator classes. The algorithm used to sign the certificate is MD5WithRSA.

(a) MainActivity



(b) InitiatorActivity



(c) TargetActivity

C. Wi-Fi Broadcast

To implement Wi-Fi broadcast on Android, the IP address of the Wi-Fi interface is firstly retrieved. Then, the corresponding Wi-Fi broadcast address is derived. Afterwards, a UDP socket is established to send broadcast messages.

Despite the straightforward implementation, trouble arose when testing the Wi-Fi broadcast. Although successful observation of broadcast packets on the receiver with *tcpdump* traces, the UDP socket on the receiver cannot receive the packet correctly. It is found out later that it was indeed a device-dependent issue. Some Android HTC devices do not handle the broadcast messages properly due to the configuration of the device.

D. Wi-Fi Direct

First, a brief introduction of Wi-Fi Direct is necessary, followed by a discussion on the choice and effectiveness of this technology. The main purpose of Wi-Fi Direct is to enable peer-to-peer connections without requiring a wireless access point (AP), allowing both one-to-one and one-to-many connections among different devices through Wi-Fi speeds. To enable compatibility for older devices using this newer technology, only one device needs to be certified to the protocol to allow for communication. Android's Wi-Fi Direct group supports legacy devices by setting up a wireless access point for these devices to connect to. However, this approach is not suitable under the DiscoverFriends application as described below. In addition, this technology enables connection establishment with any device discovered within a 200 meter radius, thereby leveraging the concept of locality. As this technology is relatively new, only devices with Android 4.0 (API level 14 or later) have this feature.

To introduce the reasoning behind choosing to utilize Wi-Fi Direct technology over other methods, two general scenarios where DiscoverFriends may be used are examined: a user is connected to a local area network and a user is not connected to one. Lastly, this section ends with an analysis of the applicability of the technology.

1) Connected to LAN: For the first scenario, the user is connected to a LAN, so we utilize Android's Network Service Discovery to detect other devices that can service DiscoverFriends. After detecting which devices are currently running the application, a UDP broadcast is sent to these devices. However, a few devices such as some HTC devices block these broadcasts, so a workaround for these exception cases is needed. Here, Wi-Fi Direct is not necessary because of the presence of a wireless network backbone.

2) Not Connected to LAN: In a latter more probable scenario, the user enters a location where there is no network infrastructure available. For example, the user, Alice, goes to a park and wants to be able to communicate to all her friends there. Here, she can use the Wi-Fi Direct technology in DiscoverFriends to accomplish this goal through a two-step process.

1) The initiator discovers nearby devices and sets up the device, making the user the group owner.
2) Attempts to establish a one-to-many connection with all the discovered devices.

For the purpose of this application, the group owner is set to be the server and the connected devices to be the clients. Once connected, the initiator is able to communicate to all its connected clients using the channel managed by the WifiP2pManager. Wi-Fi Direct is suitable for Discover-

Friends' assumptions because it leverages the locality of this technology. However, all this assumes that the initiator, as well as the friends, have the Wi-Fi Direct functionality. If the friends do not support Wi-Fi Direct, then this technology is not suitable for the DiscoverFriends environment. Although the initiator can set up his phone as a wireless access point, the network name and passphrase to connect to the device, which is generated by Android's WifiP2pManager, are randomly generated. Also, note that how DiscoverFriends is designed, discovery of new friends is not permitted unless the network is reinitialized. This is not because of Wi-Fi Direct limitations but because of preventing replay attacks. Therefore, it is not possible for the initiator to provide his friends with the access information beforehand, thwarting Wi-Fi Direct's usefulness in the application.

3) Applicability: Finally, the applicability and general drawbacks of Wi-Fi Direct is examined. Aside from compatibility with legacy devices, Wi-Fi Direct draws significant power from the Android device, so it is not feasible to keep Wi-Fi Direct on for an extended time [16], which even the system warns the user when tries to enable the feature. However, it is usable for DiscoverFriends because this application only runs for a short period of time to exchange short messages. In addition, the connection range is limited to 200 meters, but this is a suitable distance for the application.

E. OSN Adaptors

There were many OSN adaptors to choose from, where the main ones were Facebook, Google+, and Twitter. Here, DiscoverFriends implements the first two.

1) Facebook: Facebook was integrated in DiscoverFriends in order to obtain a user's Facebook friends using Graph API v2.0. Unlike the username, which is found in the URL section of a user's Facebook page, the ID is hidden away from the public is only known by Facebook friends and the Facebook servers. Therefore, by running a Graph API search query on the user's friends, the corresponding GraphObject can be obtained, which can then be used in the extraction of the user's friends' IDs. These IDs will then be placed into the initiator's request Bloom filter. Another Bloom filter is constructed by copying the previously constructed Bloom filter, which contains the hashed friends' IDs, and appending onto it, the user's own ID. Note that the length of the friend list in the worst case for the expected insertions. With this length and a 2% false positive probability, the optimal size of the Bloom filter is decided using 2. Lastly, due to a recent upgrade to Graph API v2.0 on April 2014, Facebook adds more security such that only the list of friends that use this application is returned rather than the complete list.

2) Google+: The solution is extended to support Google+. Similar to Facebook, a user's ID is extrapolated using the Google+ API. When using both OSNs, the two IDs representing a user is XORed to generate a new ID. This new ID can only be identified by the initiator and the target who is the initiator's friend on both OSNs. One issue is how to determine which friends meet this requirement. Currently, the solution is to hard code some common friends in the system. Later, it can be replaced by a smarter mechanism such as comparing the name and email to infer the potential common friend.

IV. Risk Analysis

Risk analysis is done on two potential forms of network attack: replay attack and eavesdropping by a common friend. In the following attacks, let us consider the simplified scenario where Alice is the initiator who connects to Bob.

A. Replay Attack

In the case of a replay attack, we assume an adversary is able to intercept and retransmit our application data. One purpose of this attack may be to fraudulently establish a subsequent connection between the adversary and the initiator's targeted group of friends for a malicious rendezvous. Consider now, an adversary, Eve, that is eavesdropping and wants to use that knowledge of the connection to communicate to Bob by posing as Alice. In order for Eve to execute a successful replay attack, she must firstly be authenticated by Bob.

For authentication, DiscoverFriends uses certificates, which have a limited validity period, so if Eve tries to replay data that contains an expired certificate, Bob will know. Procedure for certificate revocation for the self-organized public-key management system is described in [13]. As a result, any attempts at replay attacks are thwarted.

Although man-in-the-middle attacks are prominent in self-signed certificates as a user does not have the certificate in advance for validation, these malicious attempts are handled by proper detection of a compromised node using the trust graph in the chosen key management scheme. In other words, building a trust graph during the initial insecure network initialization phase helps enable users to determine valid certificates provided by the nodes contributing to the master graph. This scheme's strong chain of trust also protects against Sybil attacks, where an attacker can take on multiple identities in attempt to subvert the reputation system of a peer-to-peer network. Therefore, self-signed certificates by Eve will not be present in each node's CR, successfully preventing man-in-the-middle attacks.

B. Eavesdropping by a Common Friend

When a common friend, Eve, eavesdrops, she is able to identify the initiator because the hashed ID of one of his friends will come out as a positive match. Without any additional security measurements, Eve is able to decrypt subsequent messages sent from Alice using AES with the matching ID from her friend list as the key. One such approach is that instead of using the ID of the initiator as the key for AES, the application uses the public key obtained from the certificate. In the case of DiscoverFriends, an additional security measurement is present as the AES key is encrypted using public-key cryptography. As a result, the communication between initiator and target will be private, preventing future eavesdropping.

It is also important to note that aside from common friends, bystanders may be able to guess who the initiator is. This leads to guessing the respective ID as it may be mnemonic such as a variation of the initiator's name. Similar to the case of the common friend, the extra security measurement of using the public key is necessary to hide the conversation.

V. Performance Evaluation

A. Experiment Setup

To evaluate the performance of DiscoverFriends, two Android 4.0+ devices were used. As mentioned earlier, although the application supports legacy devices, using any versions older than 4.0 will undermine the application's purpose as an Android randomly generated passphrase is needed to be known to Wi-Fi Direct unsupported phones.

Here, the computational and storage costs are compared for three systems: DiscoverFriends' using hybrid encryption, DiscoverFriends using only AES with Bloom filters, and a system using prearranged ABE policy trees. In the following two sections, it is shown that using Bloom filters and a hybrid encryption technique performs the best over the latter two systems.

B. Computational Cost

Because encryption and decryption are computationally costly especially in a MANET environment, energy-efficient approaches in applications are necessary. The system using ABE keys has high computational cost due to its large keys versus DiscoverFriends using only AES with Bloom filters approach, which cuts down the cost to a fraction. However, this approach trades off security for efficiency. Here, security suffers due to by the small chance of false positives using Bloom filters, whereas the advantage of using ABE is its stronger protection scheme. The cost of encryption and key generation for ABE scales with the number of attributes and the complexity of the access policy. Note that performing offline key generation can speed up initial computational cost while encryption must be done online. Therefore, a common approach of using hybrid cryptosystems such as that in the final form of DiscoverFriends lowers computational cost by only using the slow asymmetric algorithm on keys rather than on messages while maintaining the necessary security features.

C. Storage Cost

The amount of storage used in DiscoverFriends is proportional to the subset of successfully connected friends. It is shown that DiscoverFriends' storage cost is significantly less than the other. In an experimentation, two initiators who have 100 and 1,000 friends, respectively only want to communicate to a subset of 10 friends. Table II depicts this experiment, and it is shown that DiscoverFriends' approach requires less storage space compared to the other approach. The reasoning behind this is because the Bloom filter prunes the initiator's full friend list to a targeted group of friends. However, not everyone in the list may be present or choose to respond to the initiator. As a result, only the keys corresponding to the connected friends are stored versus the entire set of ABE keys the other model has to store. Because ABE generates an ASK for each user, the number of keys corresponding to each user in an OSN may amount in the thousands, thus it is not practical. Furthermore, under this approach, the social network servers would know the keys and can defy attempts of covered rendezvous. From this, the effectiveness of using Bloom filters come to light as they only manage which users can decrypt the message using a bit array structure while cutting OSN servers out of the picture.

Table II
KeyStore size comparison in KB

	100 friends	1000 friends
Use of Bloom Filters	4.52 KB	4.52 KB
No Use of Bloom Filters (e.g. ABE)	44.9 KB	449 KB

Another measurement was done on the sizes of DiscoverFriends' different types of network packets: setup, certificate update, and normal. In detail, the setup packet consists of the user's certificate and two Bloom filters, where the parameters for the number of expected insertions n is 1000 and the false positive probability p is 0.02. The normal message has a maximum limit of 160 characters, which gets encrypted using AES-128. These measurements are shown in Table III.

Table III
Total packet sizes in bytes

	Total packet size
Setup	2,516.59 bytes
Certificate Update	481 bytes
Normal	176 bytes

VI. Related Work

A. Encryption Mechanisms

Attribute-based encryption (ABE) is an alternative method of discovering friends in place of Bloom filters. Users of ABE create three different types of keys: ABE public key (APK), ABE master secret key (AMSK), and ABE secret key (ASK). The first two are generated on a user bases while the third is created per friend. The uses of these keys are summarized as follows: APK is for encryption, AMSK is for secret keys generation, ASK is for associating the users set of attributes. There are two types of ABE schemes: Ciphertext-Policy and Key-Policy. The Ciphertext Policy Attribute-Based Encryption (CP-ABE) consists of four steps: Setup, Encrypt, KeyGen, and Decrypt. Similar to Bloom filters, ABE targets a specific group that can decode the encrypted message. Through this scheme, only a subset of all users with attributes that match the access policy can decrypt the messages. Synchronization is enabled using a key chain mechanism. The second scheme, Key-Policy Attribute-Based Encryption (KP-ABE) is similar to the first scheme, consisting of the same four algorithms but differs on attribute association. Rather than associating the attributes with the user, KP-ABE instead associates with the plaintext message. In other words, the decryption policy

enables only those users who match the ciphertext attributes, which are associated with a plaintext message. In addition, there are different flavors of ABE systems such as token-based ABE (tk-ABE) [17], which protects against key cloning.

B. Applications

Unlike LBS [18], DiscoverFriends does not need to know the user's location with GPS. The application exploits the broadcast nature of wireless communication to directly find nearby friends without the location information. Also, the user does not expose location information to any server in DiscoverFriends, which completely avoids the potential privacy leak.

In year 2009, a social network application called Safebook [19] was implemented. It adopted a decentralized architecture and capitalized on the trust relationships that existed outside of social networks. Like DiscoverFriends, it addresses the concern where an omniscient service provider such as Facebook or LinkedIn can intercept and potentially monitor interactions between OSN users, essentially nullifying certain privacy policies. Unlike DiscoverFriends however, this application is not based off a MANET environment.

In year 2012, an algorithm for social networking on OLSR MANET utilizing Delayed Tolerant Network (DTN) [20] was implemented. The core idea was to discover friends based on similar interest within the user's neighborhood. The solution realized that the use of Cosine Similarity as the similarity metric yielded the highest number of similar interest matching. Unlike this approach, DiscoverFriends' algorithm has the list of friends already and instead, focuses on how to communicate to them securely assuming they are nearby.

Recently in 2014, an idea to build symmetric private information retrieval (PIR) systems using encrypted Bloom filters was conceived [21]. Rather than putting the user's ID in the Bloom filter as in DiscoverFriends, a RSA signature of each user is placed instead. A client wanting to query a local Bloom filter constructs a blinded query using David Chaum's blinded signature scheme, which then gets signed by the server and passed back. Receiving this, the client proceeds to unblind the query to reveal the server's RSA signature for the targeted client and checks if it exists within its local Bloom filter. However, this approach like DiscoverFriends will fail with a reasonably large user base as the Bloom filter sizes will be too large and inefficient to transmit.

VII. Conclusions

In this project, a Wi-Fi-based solution, DiscoverFriends, is designed to find nearby OSN friends without disclosing one's location information to OSN servers, which eliminates the potential privacy issue in using the LBS of OSNs. The use of Bloom filters-based message exchange and a hybrid encryption system with a self-organized public-key management enables confidentiality and authentication while providing higher security by bypassing OSN servers. DiscoverFriends further obscures user ID information by combining multiple OSNs, effectively protecting messages from being identified by any single OSN. Under this model, it successfully prevents the effectiveness of eavesdropping in wireless communications as described by the risk analysis under two security threat models.

References

[1] (2014) Number of mobile monthly active facebook users worldwide from 1st quarter 2009 to 3rd quarter 2014 (in millions). [Online]. Available: http://www.statista.com/statistics/277958/number-of-mobile-active-facebook-users-worldwide/

[2] I. A. Junglas and R. T. Watson, "Location-based services," *Communications of the ACM 51, vol. 3, pp. 65-69*, 2008.

[3] M. Umaparvathi and D. K. Varughese, "Evaluation of symmetric encryption algorithms for manets," in *IEEE International Conference on In Computational Intelligence and Computing Research (ICCIC), pp. 1-3*, 2010.

[4] K. Sadasivam and T. A. Yang, "Evaluation of certificate-based authentication in mobile ad hoc networks," in *From Proceeding (464) Networks and Communication Systems*, 2005.

[5] J. Macker and M. Corson, "Mobile ad hoc networking and the ietf," *Mobile Computing and Communications Review*, 1998.

[6] P. Rathi and P. Mahalle, "Certificate revocation in mobile ad hoc networks," *International Journal of Application or Innovation in Engineering & Management (IJAIEM), vol. 2, no. 1, pp. 356-365*, Jan. 2013.

[7] S. Yi and R. Kravets, "Moca: Mobile certificate authority for wireless ad hoc networks," *in 2nd Annual PKI Research Workshop (PK103), pp. 65-79*, Apr. 2003.

[8] J.-J. Haas, Y.-C. Hu, and K. P. Laberteaux, "Efficient certificate revocation list organization and distribution," *IEEE Journal on Selected Areas in Communications, vol. 29, no. 3, pp. 595-604*, 2011.

[9] S. Capkun, L. Buttyán, and J. P. Hubaux, "Self-organized public-key management for mobile ad hoc networks," *IEEE Transactions on Mobile Computing, pp. 52-64*, 2003.

[10] Y. Zhang, W. Liu, W. Lou, Y. Fang, and Y. Kwon, "Ac-pki: Anonymous and certificateless public-key infrastructure for mobile ad hoc networks," in *IEEE International Conference on Communications, vol. 5*, 2005.

[11] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Securing mobile ad hoc networks with certificateless public keys," *IEEE Transactions on Dependable and Secure Computing, vol. 3, no. 4*, Oct. 2006.

[12] Z. Zhang, W. Susilo, and R. Raad, "Mobile ad-hoc network key management with certificateless cryptography," in *2nd International Conference on Signal Processing and Communication Systems*, 2008.

[13] K. Sahadevaiah and O. B. V. Ramanaiah, "Self-organized public key cryptography in mobile ad hoc networks," *Journal of Ubiquitous Computing and Communication*.

[14] D. Camps-Mur, X. Pérez-Costa, and S. Sallent-Ribes, "Designing energy efficient access points with wi-fi direct," *Computer Networks 55, no. 13, pp. 2838-2855*, 2011.

[15] S. Trifunovic, A. Picu, T. Hossmann, and K. A. Hummel, "Slicing the battery pie fair and efficient energy usage in device-to-device communication via role switching," in *Proceedings of the 8th ACM MobiCom workshop on Challenged networks, ACM*, 2013.

[16] K.-W. Lim, W.-S. Jung, H. Kim, J. Han, and Y.-B. Ko, "Enhanced power management for wi-fi direct," in *In Wireless Communications and Networking Conference (WCNC), IEEE, pp. 123-128*, 2013.

[17] M. J. Hinek, S. Jiang, R. Safavi-Naini, and S. F. Shahandashti, "Attribute-based encryption with key cloning protection," *International Journal of Applied Cryptography, vol. 2, no. 3, pp. 250-270*, 2012.

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

[19] L. A. Cutillo, R. Molva, and T. Strufe, "Safebook: A privacy-preserving online social network leveraging on real-life trust," *IEEE Communications Magazine vol. 47, no. 12, pp. 94-101*, 2009.

[20] T. Sanguankotchakorn, S. Shrestha, and N. Sugino, "Effective ad hoc social networking on olsr manet using similarity of interest approach," *In Internet and Distributed Computing Systems, pp. 15-28*, 2012.

[21] M. Marlinspike. (2014, Jan.) The difficulty of private contact discovery. [Online]. Available: https://whispersystems.org/blog/contact-discovery/