Social Networks for Importing and Exporting Security

Bangdao Chen and A.W. Roscoe

Oxford University Computer Science Department
James Martin Institute for the Future of Computing
{Bangdao.Chen,Bill.Roscoe}@cs.ox.ac.uk

Abstract. Online social networks are rapidly changing our lives. Their growing pervasiveness and the trust that we develop in online identities provide us with a new platform for security applications. Additionally, the integration of various sensors and mobile devices on social networks has shortened the separation between one's physical and virtual (i.e. web) presences. We envisage that social networks will serve as the portal between the physical world and the digital world. However, challenges arise when using social networks in security applications; for example, how can one prove to a friend (or Friend) that your Facebook page belongs to you and not a man in the middle? Once you have proved this, how can you use it to create a secure channel between any device belonging to you and one belonging to your friend? We show how human interactive security protocols (HISPs) can greatly assist in both these areas and in general create a decentralised and user-oriented model of security. And we demonstrate that by using this security model we can quickly and efficiently bootstrap security for sharing information within a large group.

1 Introduction

Online social networks (OSNs), such as Facebook, Google+, Foursquare, Twitter, and LinkedIn, have enjoyed phenomenal growth in recent years. The authors of [13] analysed relationships and communication on Twitter, and pointed out that Twitter also plays the role of a social medium: information can spread widely and quickly. For example, in less than 12 hours after the first tweet of Osama Bin Laden being killed, there were 2.2 million tweets related to this event [3]. OSNs therefore not only help to create and maintain a large amount of relationships between humans, they also provide efficient and convenient platforms for sharing and spreading data amongst a large audience.

The future of OSNs is changing with the growing pervasiveness of device connections. For example, the CEO of Ericsson [2] has forecast that there will be 50 billion device connections by 2020, which will create a "connected society". Sensors are often used to make data about physical objects available online, for example, to display the sensory data on OSNs. An IBM researcher connected his

R. Calinescu and D. Garlan (Eds.): Monterey Workshop 2012, LNCS 7539, pp. 132–147, 2012.

house with Twitter[1]: a set of sensors are used to generate tweets about power consumption, water usage and the temperature of the house. We also notice that there are plenty of body-monitoring sensors [1] with mobile connectivity in the market today.

The integration of OSNs on mobile devices has further shortened the separation between our virtual presences on the web and our physical existence. By using a mobile device, OSNs have the opportunity to collect more private data; for example, location data or medical data from on-body medical sensors. There is already a clear need for a solid security model for social networking, and the more we use them for, the more we need them to be secured.

Given that the social network providers are increasingly making their applications available as secure web sites, there remain two primary concerns:

A How can we know that a given OSN page belongs to a given user: the *identification*, or *authentication* problem? In general such knowledge may be absolute or come with some identified confidence level.
B The provision of appropriate security models for collecting, using and sharing data from the local user and his or her devices including sensors.

In this paper we concentrate on A, and furthermore show how security developed for social networking can be used to conveniently bootstrap other secure connections.

We imagine that in general solutions to A might involve any one, or combinations of (i) pre-existing security infrastructures such as PKIs, (ii) reputational models based on trust ratings by other network users, and (iii) bootstrapping security by person-to-person contact by interaction outside the social network. In this paper we concentrate on (iii) and show how *Human-Interactive Security Protocols* (HISPs) can be used to do this efficiently when there is a means for getting a small amount of information from the owner of the page that is to be authenticated to the person who wants to authenticate it. This transmission might be via personal contact or using a second medium that is trusted as authentic.

In this paper we make the following contributions:

1. We propose a security model that exploits the trust on social networks by using HISPs. This model can be used to authenticate online identities and create secure connections between devices.
2. We demonstrate these by implementing a prototype system. It can efficiently bootstrap security for a large group. It shows the practicability of using our security model in future mobile computing.

2 Using a HISP

A typical HISP relies on the assumption that there is an empirical channel in a specific application, in which one or more humans can compare a short

[1] `http://stanford-clark.com/andy_house.html`

authentication string (SAS) received from the empirical channel. An empirical channel is a human-based, non-fakeable channel, for example, face-to-face conversations, video calls or voice calls. The best of these protocols, for example those of [14,15,16,17,18,19,20,22,23], enable assurance to these humans that there is no attack that would allow an intruder to get the system into an insecure state (where the connections established are other than what the humans believe), with probability meaningfully greater than 2^{-b}, where b is the number of bits in the check-string. In addition, to have such a chance, the attacker will have a $1 - 2^{-b}$ chance of his presence being revealed by the difference between the strings.

HISPs can be thought of as tools that enable one (perhaps informal) authentic channel to efficiently authenticate, and then secure another one. This means that they have two complementary potential uses in social networking.

1. We can use a HISP to authenticate online identities by using existing connections (typically personal or telephone conversations between the humans involved). In this case, we import security from existing social relationships to social networks.
2. We can use a HISP to create secure connections between devices, in this case, we can use authenticated social network accounts as proxies to display SASs. This can significantly improve the usability of HISPs. We therefore export security from social networks to other applications. This also provides a new channel of sharing information directly between devices, which is useful especially when the OSN providers cannot guarantee the privacy of information posted online.

In the following sections we will introduce two HISPs that we use in our implementation.

2.1 Pair-Wise HISP

Below is the pair-wise HISP we use:

1. $A \longrightarrow B : \ hash(0 : hk_A), hash(k), Info_A,$
2. $B \longrightarrow A : \ hash(1 : hk_B), pk, Info_B,$

Each party creates a *hash*, or *digest* key: we call these hk_A and hk_B. These are needed to randomise the final check-string. A creates a session key k. B either creates freshly, or re-uses, an asymmetric key pair (pk, sk). There is no need for the "public" key pk to be certified. The length of these keys will depend on the desired level of security[2], the amount of available computing power, and the crypto-system in use.

In the first pair of steps of the protocol, A and B both commit each other without knowledge to values of hk_B or hk_A. The only one of the four parameters

[2] The key certainly needs to be strong enough so that there is no realistic chance of it being broken during the life of the session being established. Further strength is required to ensure that the contents of that session remain secret after it ends.

hk_A, hk_B, pk and k communicated openly is B's public key pk. $Info_A$ and $Info_B$ are the information A and B wants to authenticate. In our example, when Alice wants to verify Bob's OSN account, $Info_B$ contains Bob's social network account profile; similarly, $Info_A$ contains Alice's social network account profile when Bob wants to verify Alice's OSN account.

The protocol now proceeds:

3. $A \longrightarrow B : \ hk_A, \{k\}_{pk}$
4. $B \longrightarrow A : \ hk_B$

The second part of Message 3 is to tell B the actual value of the session key, which is now checked against the hash. It is the transmission of the unencrypted keys hk_A and hk_B at this stage that represents the core of the protocol. Firstly, of course, the participants must check that these are the same values that were represented in Messages 1 and 2. If not, the run is abandoned. Secondly, they (and anyone else who has been listening in) can compute a value for

$$digest(hk_A \oplus hk_B, (pk, hash(k), Info_A, Info_B))$$

where $\oplus$ is bit-wise exclusive or and (X, Y) is an ordered pair. The protocol completes successfully if A (or A and B) are convinced that their two versions of the value – the check-string of this protocol – are equal: in becoming convinced they must not use a channel which can be "spoofed" by an intruder. Typically one will read their value to the other, or A will read B's value directly and compare it with her own. Whichever knows that the two values are equal can conclude that the link is authenticated. Typically this is either A or both of them. It is this comparison that makes it a HISP.

Naturally, if the protocol has proceeded uninterfered with, A's and B's values will be equal. If, however, an intruder has imposed his own values onto the receivers of Messages 1–4, A and B will not agree on all four parameters. For security, what is important is that they agree on pk and $hash(k)$, so we will concentrate on what happens if the intruder interferes with these.

The digest function [17,18] is designed so that, as hk varies, the probability that $digest(hk, X) = digest(hk, Y)$ for $X \neq Y$ is less than ϵ, where typically ϵ is very close to the theoretically optimal value of 2^{-b} for b the number of bits in the output of $digest$. It must also have the property that for any fixed value d, the chance that $digest(hk, X) = d$ as hk varies is less than ϵ also. More details of this protocol can be found in [9]. Formal verification of this protocol is presented in [21].

An important quality a HISP must have is that it protects the SAS that the users compare from combinatorial searching by potential attackers: analysis must be able to show that no matter what conceivable amount of computing an attacker uses, he has no better chance of getting lucky and persuading the users to agree on an SAS in inappropriate circumstances than if it had made a single guess. All the HISPs we see in this paper have that property.

2.2 Group HISP

The Symmetric HCBK (SHCBK) protocol [18] is used in our implementation. This, the general description, connects an arbitrary-sized group. Good examples of group authentication using HISPs are GAnGs [7] and SPATE [24].

1. $\forall A \longrightarrow_N \forall A' : A, INFO_A, hash(A, hk_A)$
2. $\forall A \longrightarrow_N \forall A' : hk_A$
3. users compare $digest(hk^*, \{INFO'_A | A \in G\})$, where hk^* is the XOR of all hk_A's for $A \in G$

SHCBK has each node "publish" its name and a collection of information that it wishes to be authentically connected with that name. It also sends a hash[3] of a randomly generated key hk_A coupled with the name. Once it has received that information from all nodes, and therefore become committed to the set of identities, $INFO$ and hashed keys it will use, it publishes its previously secret hk_A. The point is that by the time of this last publication, it was in fact *committed* to all the data used in the above protocol, even though it does not yet *know* all the hk_As. HCBK stands for Hash Commitment Before Knowledge. A careful security analysis of this protocol (see [18], for example) demonstrates that any attacker is unable to profit from combinatorial analysis aimed at getting the SASs (i.e. digests) to agree even though nodes have different views of the authenticated information. Good HISPs such as SHCBK therefore offer maximum security for a given amount of human effort.

We can reduce the number of human interactions if there is a trustworthy Initiator I, consider the rest of the group as G', then the above protocol can be modified as following: in the process of comparing digest values, I compares digest value published by $\forall A$ ($A \in G'$), $\forall A$ compares the digest value published by I; I then publishes the final result of digest comparison, $\forall A$ checks this result. We call it Semi-SHCBK protocol. Therefore the total number of messages to be exchanged via empirical channels changes from $N(N-1)/2$ to $3N-3$. If there is a trustworthy Initiator, when $N > 6$, Semi-SHCBK protocol is more efficient than SHCBK protocol.

The key generation is simple: we include a copy of an uncertified Diffie-Hellman public key in $INFO_A$, then after a successful run of SHCBK or Semi-SHCBK protocol, each user generates $N-1$ shared pair-wise secret keys sk. For example, $sk_{\alpha\beta}$ means a shared secret key between user α and user β. To generate a group key sk_G, the following group key protocol is used ($\longrightarrow_S$ means sending encrypted information using a corresponding pair-wise secret key):

1. $\forall A \longrightarrow_S \forall A' : Nonce_A$
2. $sk_G = Nonce^*$, where $Nonce^*$ is the XOR of all $Nonce_A$'s for $A \in G$

Each member also generates an anonymous ID. It can be used to publish information anonymously on OSNs. The anonymous ID is created by $hash(Nonce_A,$

[3] Hash means a standard cryptographic hash function that has two main properties: collision resistance, and inversion resistance.

A's social network ID) *mod* 10^{15}. This will generate a 15-digit[4] ID for each group member.

2.3 Improving the Usability and Security of HISPs

The practicability of using HISPs is in inverse proportion to the cost of human effort. For example, factors that determine the practicability are: the availability of empirical channels; the length of information to be compared; and the times of comparison required in one run.

In order to reduce the amount of human effort without compromising security, one solution is to allow automated comparison of SASs online. For example, when OSN pages are being used to display SASs in HISPs there is clearly also the the option for these same pages to compare the SASs provided they are connected securely to the local device that is participating in the HISP.

If all participants have this property we could use a longer SAS, but in general we assume that there is likely to be some human participant creating the link in person. The primary motivation for using HISPs is, after all, allowing this.

3 Proving Online Identities

In order to use OSNs as empirical channels we must answer the following question: "*how do I know that what I am seeing on the page comes from the person or other entity that I think it does*". To better analyse this problem, we divide it into two sub-questions: how do I know the (e.g. Facebook) page I am seeing is authentic within the OSN? and how do I know it belongs to the person I think it does? The first of these questions can be solved by conventional computer security, for example, the *https* service on OSNs. It is therefore assumed that all relevant interactions with the OSNs are via their *https* interfaces.

The second question can be converted into the following one: "is this an established Friend for which you are certain of the link between page and person?" If the answer is yes, then secure access to that page is clearly a good empirical channel. This is the most common way of authentication in our daily life. For example, one may have experiences in interacting with a social network account, one may authenticate a social network account by the number of common Friends, or one can authenticate a social network account by viewing its profile, Friends list, photos, history of participated events and other context information.

If we can not make our decision based on past experiences, we may use telephony or physical interactions to accomplish this task. A HISP can therefore be used to authenticate OSN accounts. For example, Alice wants to know that the social network account of Bob is authentic; if Alice has a phone number of Bob and she is certain of the authenticity of this phone number, she then runs a HISP with Bob to verify his account by using telephony as the empirical channel.

[4] We use the same length of digits as Facebook ID.

Note that the availability of HISPs provides us with the flexibility to bootstrap security from any existing authentic connection, whether one derived from physical proximity or other means such as telephony.

And there are other alternatives of authenticating online identities in practice, for example:

1. Centralised authentication. For example, Twitter provides authentication service. The verified account will display a special indicator (a small icon or a "badge"). However this service is limited to celebrities on Twitter. A similar situation can be found in other OSNs.
2. Introducing decentralised authorities. For example, we can publish OSN accounts of a group on a company's *https* web-page. In this case, the company acts as an authority which authenticates a group. Similarly, a trusted organisation or a trusted individual can also play the role of an authority. For example, a community leader may only keep Friends that belong to the community, therefore his or her Friend-list can be used to help authenticate the community members. This can be used to replace the human effort of authenticating group members and can greatly improve the application in authenticating a group when its size is large. In our implementation, when prompting users to verify the member-list of a group, we provide an option for users to use a trusted authority (in the form of an *https* web-page). Details of this approach are presented in Section 5.
3. Introducing trust ratings. Rating by trust is a common practice in OSN research, for example, [12] describes a semantic web-based OSN, and they developed algorithms to rate the inferred reputation of a node. Another distinct example is PGP. It exploits ratings to determine the level of authenticity of downloaded public keys. A rating scale of 1 to 4 is used: full (complete trust), marginal (partial trust), untrustworthy and don't know. The most distinct advantage of this method is that it provides pervasive automated authentication. We have implemented a demonstration rating system by using the same ratings introduced in PGP (see Section 5).
4. Blackballing. Blackballing[5] is a voting method used in many gentleman's clubs: members have a large number of white and black balls and each member casts a single ball into the ballot box to vote for a proposition, if there are one or more black balls in the ballot box, everyone will immediately know this proposition has been vetoed. In our implementation, each member checks the list objects one-by-one, if one object is "vetoed" by one member, then list L is "vetoed". This is also a form of utilising "crowd knowledge" which effectively reduces the security mistakes when members manually authenticate each other.

4 Bootstrapping a Large Group by Using OSNs

An important assumption has to be made before bootstrapping security for a group: members of a group are capable of verifying the legitimacy of each other

[5] `http://en.wikipedia.org/wiki/Blackballing`

within the group. This is supported by the methods introduced in Section 3. It allows us to start our discussion of how to bootstrap a large group by using OSNs. The insecure state we will address is where one trustworthy user believes he has an authenticated connection to another but is in fact connected to a third party (e.g. the attacker).

In some cases when bootstrapping a HISP group the identities (however defined) of those participating will be obvious. Perhaps this will be because all know each other well and have agreed to connect, or perhaps it will be because they are together is some easily identifiable context such as sitting around a table. In these cases all that is necessary for them to start the protocol is the number of them. For small groups this will be obvious; for large ones they might either organise a count themselves or build up a list to which they agree.

In other cases – for example where some members of the group do not have a direct link – it will certainly be necessary to establish the list of participants in advance. In this case the names on the list will need to be authenticated. Each intended party can check if his/her name is on the list, but it may be more difficult to establish that no undesirables are on it.

The correctness of bootstrapping a group can be defined as follows: all members acknowledge a list L, which contains details of all members; the resulting group G contains exactly the same number of members recorded in L and no one, except for the members included in L, can be allowed to join G. To fulfill this task, we need to identify and overcome the following challenges:

- Collecting group information. This is to create list L. [7] presents two solutions for collecting information from group members when they are in the same room: the first solution is to use an untrusted projector as a central node by displaying its Bluetooth address as a 2D barcode; all members connect their mobile phones to the projector by reading this barcode and send their details to this projector which then broadcasts list L to the group. The second solution is to create a tree structure of collecting member's information one-by-one by reading 2D barcodes of Bluetooth addresses. These methods are too cumbersome and inconvenient when the size of the group is large. In remote scenarios, collecting group information becomes more difficult since group information is often discrete and inconsistent.
- Counting and authentication. Counting is to check whether the size of group G matches with the size of list L. Authentication is to check whether members included in list L are legitimate. In general, there are two types of attacks: (i) man-in-the-middle (MITM) or outsider attacks; (ii) Sybil [10] or insider attacks. Counting and authentication is to detect attacks of (i) and (ii). Normally, if authentication is prudent, authentication alone can detect attacks of (i). However, an insider may be capable of providing multiple fake identities[6] to get access to more resources, therefore counting is necessary to detect attacks of (ii). Depending on physical interactions to perform counting and authentication has many limitations. For example, members of a group

[6] The fake identities can be different copies of the insider's identity or fake identities of others.

may be distributed and remote, and physical interactions may be unavailable; humans can be lazy and careless, for instance, they may not correctly count a group, or they may not correctly perform actions of authentication.

To simplify our discussion, we assume group formations are presented in the form of events; for example, the Department of Computer Science creates a list of their staff and students in order to share their project data; they arrange an event (e.g. a Facebook event) by informing all members within the department via emails or by posting a notice to the public. We generalise these events of group formation into the following two events:

A. Pre-emptive event: group members know each other and they all trust the Initiator before the event runs, therefore, the Semi-SHCBK protocol is used.
B. Non-pre-emptive event: except for the Initiator, the rest of the group does not know of the event in advance and they may not know each other. The Initiator sends out invitations to ask for participation. Those who accept it join the event. Members may not all trust the Initiator and the SHCBK protocol is used.

In our solution, all functions are achieved and performed by using a mobile application installed on users' mobile phones.

4.1 Collecting Group Information

OSNs provide two functions that make collecting group information convenient and efficient: (i) information on OSNs is rich and well formatted which is convenient for exporting information to other applications; (ii) OSN accounts are managed according to social relationships; for example, we can create and manage different groups[7], and we can create an event (e.g. a Facebook event or a Google+ page) and invite Friends to join.

In Event A, we can assume that group members are already Friends of the Initiator on OSNs, therefore the Initiator can simply create a group by selecting accounts from his/her Friend list, and then export the group information to our mobile application. In Event B, we assume that group members may not be Friends with each other. The Initiator can simply create an event and then notify all others. For example, the Initiator can introduce this event by sending emails or by publishing it on posters. Others can easily identify and join this event on OSNs. In the end we can export group information from this event.

This process can also be made via physical interactions, for example, one can display an event's OSN page address as a 2D barcode and others can read this barcode to join this event. Therefore by using OSNs, we can support group formation when group members are collocated, remote to each other, or a mixture of the former two situations.

[7] On Google+ a group is presented as a "circle".

4.2 Counting and Authenticating Members

Counting, if made by humans, has limitations. For example, one may make mistakes when the size of the group is large. [7] assumes humans can accurately count less than ten individuals via physical interactions. They randomly divide a large group into small subgroups in order to allow humans to count and verify members correctly. This action provides greater usability but leads to weaker security: there may be the chance that attackers are allocated to the same subgroup. The probability of attack detection [7] is less than the value of $1 - 2^{-b}$ assumed by the HISP (b is the bit-length of the SAS). In addition, subgrouping can be laborious and inconvenient since it has to be randomised.

Authentication normally requires more human efforts. For example, in [7] they use visual channels (created by mobile phone display screens and cameras) to check the presence of identities. Since visual channels of reading 2D barcodes on mobile phones are normally unidirectional, a symmetric authentication of a group of size N requires $N(N-1)/2$ interactions. This number increases quickly when the size of the group increases.

When using OSNs as empirical channels, we can first divide the authentication process into the following two steps:

1. Authenticate OSN accounts included in list L are legitimate. We call it the authentication of online identities.
2. Read and compare digest values displayed on members' OSN pages. We call it the authentication of connections.

Step 1 is to ensure that we can use OSN accounts as proxies of our physical presences. Step 2 is to test the presence of MITM attackers by using a HISP, which is to authenticate that the electronic connection is correctly connected to the intended device represented by the OSN account. This strategy can remove the requirement for physical interactions in Step 2.

In [7], counting is important because an insider can create multiple fake identities and then perform physical interactions of authentication multiple times. In our solution, the only chance of successful insider attacks is to add fake identities in list L and pass the authentication in Step 1. In Section 3 we have discussed various techniques of proving online identities. These allow humans to conveniently and efficiently adapt their authentication strategy according to different scenarios. In addition, we can conveniently run a program to automatically count and check whether the number of responsive[8] (or active) OSN accounts is equal to the number of accounts included in list L. We therefore conclude that counting is unnecessary in our solution and there is no need of subgrouping. This improves both security and usability.

More importantly, once we have authenticated that OSN accounts included in list L are legitimate, Step 2 can be made automatically since we use OSN accounts as proxies of our physical presences. This is a significant improvement which provides more capacity for large groups, for example, groups with size

[8] Those who display the digest value.

over 100. In addition, we can display long digest values without increasing the cost of human efforts.

It is worth noticing that on OSNs, the cost of Sybil attacks of creating multiple fake accounts are higher because OSN providers, for example, Facebook[9], Google+, require unique identifiers (email addresses or mobile phone numbers) to register, and they keep records of online interactions which can be used as indicators of authenticity. We also notice that by reducing physical interactions, we can reduce impacts from other uncontrolled factors; for example, the luminous intensity, the physical distances, the quality of mobile phone cameras or display screens, and most importantly, the human complacency.

Another significant improvement in usability may be that we can allow delayed running of HISPs. Experiments of relying on physical interactions to run HISPs have one implicit assumption that all members have to finish the process of authentication within a short period of time. And it is the reason that reducing time is critical for improving usability. However, in practice, the cost of coordination can be high and humans may not necessarily be available of carrying out the same physical action at the same time. This problem can be more significant[10] when humans are remote to each other. By using OSN accounts as proxies of our physical presences, we can divide the authentication process into two separate steps discussed earlier in this section. Because Step 2 of reading and comparing digest values can be automatically completed by using a program, and Step 1 of authenticating the legitimacy of OSN accounts in list L can be carried out asynchronously, the running of HISPs can be delayed until the last member completes the authentication in Step 1.

This allows more useful security applications. For example, a department sends out notifications of bootstrapping a secure network for internal communication to its employees. Some employees are traveling abroad and they are not responding immediately. By using our solution, the program keeps waiting until the last employee responds which triggers the authentication process. After the authentication process has been finished, the program displays results to all employees.

5 Demonstration Implementation

We have implemented a secure location sharing service to demonstrate the use of our security model. We have developed three versions of mobile applications: RIM (Blackberry), Android, and iOS (iPhone, iPad and iTouch). One server SO is used as the coordination server. All devices are connected to SO. After they have successfully bootstrapped security for the group, they start to share their locations with each other.

[9] In our investigation, we discover that Facebook normally requires at least one mobile phone number to register; and accounts registered by email addresses will later be required to be authenticated via a mobile phone number.

[10] Our experiment shows evidence of high cost of coordination.

The mobile phone application first checks the ratings; if there are accounts which fail to pass the rating check, it will prompt the user with a dialogue calling for authentication resource (from a decentralised authority), it will automatically remove the authenticated objects from the stack of the member list; the objects that are left on the stack will be verified by empirical authentication, for example, by using a HISP. Figure 1 shows the flow chart of the authentication process.

Note that while the current practices of implementing a rating system are mostly experimental, we observe that the presence of a decentralised authority is strong in scenarios with security demands. For example, in a conference scenario, the organiser can manage the "guest list" of the conference's Facebook event. He or she can either remove those illegal "guests" or set this event to be visible only to the "guests" on a given "guest list". In an online community, the community leader can manage the legitimate list of community members on his or her social web-page (for example, he or she keeps the list as a group in the Friends-list).

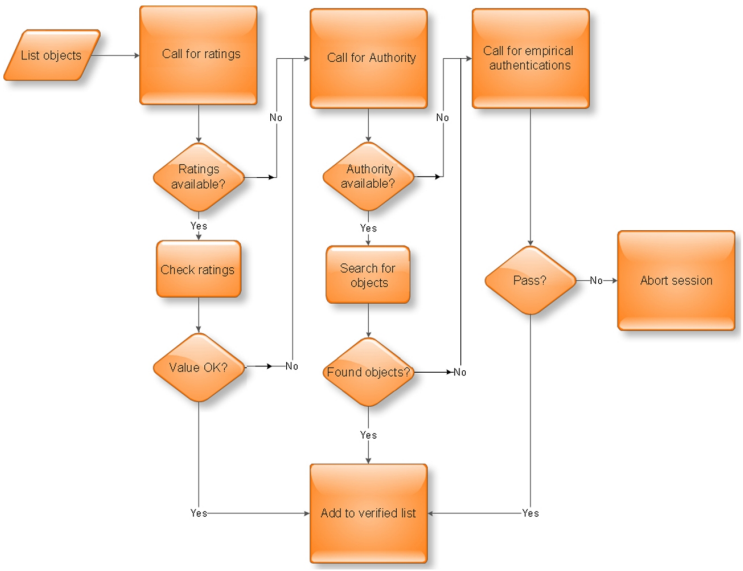

Fig. 1. The flow chart of the authentication process

If the entire member list has been verified, the protocol starts to run. The user will start to share his or her data of locations on Facebook (or directly between devices) if the protocol has been finished successfully. Figure 2 shows the screen shots of the application on Android.

We use Bouncy Castle Crypto Java API on RIM and Android; and OpenSSL C Library for iOS. We use 1024-bit Diffie-Hellman public keys to generate shared secret keys; 128-bit AES is used to encrypt data.

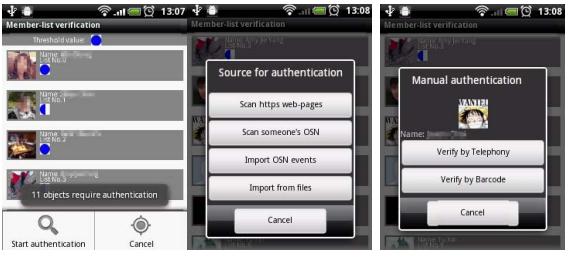

Fig. 2. Screen shots of the mobile application

6 Performance Analysis

We have tested the mobile applications on Blackberry Bold 9000 (BB9000) (4 devices), Blackberry Storm 9500 (BB9500) (1 device), HTC Wildfire (HTC) (1 device), Dell Streak (Dell) (1 device), iPhone 3 (1 device), iPad 1 (2 devices). 10 volunteers joined this test. They were located at different addresses. Coordination was made via phone calls, sending SMSs, and messaging on OSNs. Note in order to simplify our test, the member-list was imported from a Facebook event. We assumed there was a trustworthy leader. Therefore, the semi-SHCBK protocol was used. A total of 20 messages are exchanged. The size of the data sent by one device is about 18 KBytes. Compared with using traditional public key certificates, our method allows binding of contextual data (e.g. photos, voices or videos) to the uncertified public key we use in addition to names. We call these secondary security information which can be used to improve security as well as usability. Figure 3 shows the time consumption of bootstrapping a group of all the devices we have. The total time cost is around 193 seconds.

We can see the cost of coordination is high in group formation because of many uncontrolled random factors. However, the verification and comparison is efficient and only takes a small fraction of the total time.

Table 1. Facts and statistics

Device	Time	Ratio	Speed1	Speed2
BB9000	3.69s	99%	1.72kb/s	4.32kb/s
BB9500	4.49s	99%	1.35kb/s	3.75kb/s
HTC	3.74s	99%	1.56kb/s	4.80kb/s
Dell	0.85s	99%	2.42kb/s	7.15kb/s
iPhone	0.11s	99%	4.38kb/s	8.74kb/s
iPad	0.08s	99%	4.06kb/s	13.7kb/s

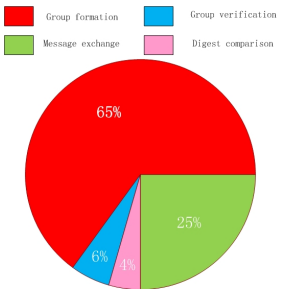

Fig. 3. Time consumption

Table 1 shows the facts and statistics of different devices. The second column is the time of computing DH secret; the third column is the ratio of the time of computing DH secret against the time of total on-device computing (excluding communication); the fourth column is the speed of connection between the device and the coordination server; the last column is the speed of the connection between the device and the Facebook server. We can see the time of on-device computation mostly originates from the DH secret computation.

According to the above analysis, we can identify two challenges for the future: (A) providing more convenient methods for large ad hoc group formation; (B) increasing the speed of mobile connections to allow including more contextual data in the protocol. Challenge A requires research on both security and usability. For example, should a group be formed using a single initiator, a tree structure, broadcasting over a fully connected graph, or some other topology? Challenge B is less significant since there are continuous developments in improving the speed of mobile connections; for example, the deployment of 4G network.

7 Related Research

WhozThat [4] is a system making use of OSN IDs among mobile phones: two users exchange their OSN IDs using Bluetooth, and it then introduces social context into the local context; for example, one may play the favourite music of the other. This is similar to our solution of binding OSN IDs with mobile devices while our intention is to facilitate identification and connection rather than interaction between humans. CenceMe [11] is a more advanced mobile OSN system which detects users' social activities by analysing sensory data on mobile phones. It demonstrates a well designed integration of OSNs on mobile phones: automated input of social information (deducted from sensory data) replaces traditional manual input. This is similar to our vision for future OSNs; for example, sensor networks like on-body sensor networks can be exploited by OSNs to automatically generate and display social patterns.

In [8] the authors presented a concrete implementation of Cloud Computing Service (for storage) on Facebook. However, there is no description as to actually utilise the Cloud after creation. Our solution gives a clear data flow between different interfaces and it can be put in use instantly.

Security is a key enabling factor for the above practices. In [5] the authors suggested OSN operators should not be trusted and data should be encrypted before posting online. They provided an example of creating a peer-to-peer system by using a pair-wise HISP to distribute public keys. A similar example was discussed in [6], which proposed a completely decentralised peer-to-peer system by storing data on user devices.

We notice that although there is much research on creating decentralised systems to improve security, practices without using a PKI or existing security infrastructures can be difficult. And such peer-to-peer systems are not efficient when the scale of sharing increases. Practices introduced in [7,24] reveal the high complexity of group HISPs when using physical interactions to collect group

information and authenticate members, therefore they are not practical when bootstrapping a large ad hoc group.

8 Conclusions

We have revealed the challenges of authenticating online identities and bootstrapping security for a large ad hoc group. The model of social networks for importing and exporting security we have presented can be used to (i) exploit existing social relationships to authenticate online identities and (ii) exploit existing online relationships to efficiently bootstrap security for a large ad hoc group. This provides a way of incorporating social context into security which can be used to deal with changing security requirements emerging from new applications. The secure location sharing service we have implemented demonstrates these features of this model.

The security of social networks remains an interesting problem on which more work is required. Its attack models based on technology are likely to be similar to those of other online services, but there is also a social/psychological dimension to investigate. We believe that in the future the growing investment in security by social network companies will make our solution more secure when exporting security to other applications, and the development of computing power on mobile devices will make it more efficient in supporting security services.

References

1. Body-monitoring sensors, `http://store.runkeeper.com/`
2. CEO to shareholders: 50 billion connections 2020, `http://www.ericsson.com/thecompany/press/releases/2010/04/1403231`
3. How Fast the News Spreads Through Social Media, `http://blog.sysomos.com/2011/05/02/how-fast-the-news-spreads-through-social-media/`
4. Beach, A., et al.: Whozthat? evolving an ecosystem for context-aware mobile social networks. IEEE Network 22(4), 50–55 (2008)
5. Anderson, J., Diaz, C., Bonneau, J., Stajano, F.: Privacy-enabling social networking over untrusted networks. In: Proc. WOSN 2009 (2009)
6. Buchegger, S., Datta, A.: A Case for P2P Infrastructure for Social Networks - Opportunities & Challenges. In: Proc. WONS 2009 (2009)
7. Chen, C.-H.O., et al.: GAnGS: gather, authenticate 'n group securely. In: The 14th ACM International Conference on Mobile Computing and Networking (2008)
8. Chard, K., Caton, S., Rana, O., Bubendorfer, K.: Social cloud: Cloud computing in social networks. In: Proc. IEEE CLOUD 2010 (2010)
9. Chen, B., Nguyen, L., Roscoe, A.W.: Reverse authentication in financial transactions and identity management. To appear in Wireless Networks, Mobile Networks and Applications (2012)
10. Douceur, J.: The Sybil Attack. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 251–260. Springer, Heidelberg (2002)
11. Miluzzo, E., et al.: Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In: Proc. ACM SenSys 2008 (2008)

12. Golbeck, J., Hendler, J.: Accuracy of metrics for inferring trust and reputation. In: 14th Int'l Conf. on Knowledge Engineering and Knowledge Management (2004)
13. Kwak, H., Lee, C., Park, H., Moon, S.: What is Twitter, a social network or a news media? In: Proc. the 19th Int'l Conf. on World Wide Web (2010)
14. Laur, S., Nyberg, K.: Efficient Mutual Data Authentication Using Manually Authenticated Strings. In: Pointcheval, D., Mu, Y., Chen, K. (eds.) CANS 2006. LNCS, vol. 4301, pp. 90–107. Springer, Heidelberg (2006)
15. Lindell, A.: Comparison-Based Key Exchange and the Security of the Numeric Comparison Mode in Bluetooth v2.1. In: RSA Conference (2009)
16. Nguyen, L. (ed.): Part 6: Mechanisms using manual data transfer
17. Nguyen, L., Roscoe, A.: Efficient group authentication protocol based on human interaction. In: Proc. FCS-ARSPA 2006, pp. 9–31 (2006)
18. Nguyen, L., Roscoe, A.: Authenticating ad hoc networks by comparison of short digests. Information and Computation 206, 250–271 (2008)
19. Nguyen, L., Roscoe, A.: Separating two roles of hashing in one-way message authentication. In: FCS-ARSPA-WITS (2008)
20. Nguyen, L., Roscoe, A.: Authentication protocols based on low-bandwidth unspoofable channels: a comparative survey. Computer Security 19(1), 139–201 (2011)
21. Roscoe, A., Smyth, T., Nguyen, L.: Model checking cryptographic protocols subject to combinatorial attack, `http://www.cs.ox.ac.uk/files/4157/guess.pdf`
22. Roscoe, A.W.: Human-centred computer security (2006) (unpublished draft)
23. Vaudenay, S.: Secure Communications over Insecure Channels Based on Short Authenticated Strings. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 309–326. Springer, Heidelberg (2005)
24. Lin, Y.-H., et al.: SPATE: Small-Group PKI-Less Authenticated Trust Establishment. IEEE Transactions on Mobile Computing 9(12), 1666–1681 (2010)