CrowdShare: Secure Mobile Resource Sharing

N. Asokan[1], Alexandra Dmitrienko[2], Marcin Nagy[5], Elena Reshetova[4], Ahmad-Reza Sadeghi[2,3], Thomas Schneider[3], and Stanislaus Stelle[3]

[1] University of Helsinki, Finland
`asokan@acm.org`
[2] Fraunhofer-Institut SIT Darmstadt, Germany
`{alexandra.dmitrienko,ahmad-reza.sadeghi}@sit.fraunhofer.de`
[3] Technische Universität Darmstadt, Germany
`{thomas.schneider,stanislaus.stelle}@cased.de`
[4] Intel Open Labs, Finland
`elena.reshetova@gmail.com`
[5] Aalto University, Finland
`marcin.nagy@gmail.com`

Abstract. Mobile smart devices and services have become an integral part of our daily life. In this context there are many compelling scenarios for mobile device users to share resources. A popular example is tethering. However, sharing resources also raises privacy and security issues.

In this paper, we present `CrowdShare`, a complete framework and its (Android) implementation for secure and private resource sharing among nearby devices. `CrowdShare` provides pseudonymity for users, accountability of resource usage, and the possibility of specifying access control in terms of social network relationships. Further, `CrowdShare` preserves secure connectivity between nearby devices even in the absence of the mobile infrastructure. We have implemented `CrowdShare` on Android devices and report good performance results.

1 Introduction

The popularity of inexpensive communication services like Skype, Gtalk, and WhatsApp is increasing rapidly. They allow people to communicate with almost the same ease as with phone calls and Short Message Service (SMS) messages, but at a significantly lower cost to the users. However, the pre-requisite to all such services is Internet access, which can be quite difficult to obtain in certain situations. First, Internet access can be expensive while traveling abroad. As a result, *tethering*, the process of sharing Internet connectivity from one device by turning it into a wireless access point that other devices can connect to, has gained popularity. Some devices provide tethering as part of their base functionality, while other third party applications like JoikuSpot [9] and OpenGarden [1] can enable tethering. Second, in some situations Internet connectivity may be impossible like in the aftermath of a disaster or while visiting rural areas with little network coverage or when organizing demonstrations against totalitarian regimes. In such situations, ad-hoc mesh networks among mobile devices can provide similar communication or data exchange services. For example, the Serval [2] project aims to preserve connectivity between mobile devices by providing MeshSMS and Call services even in the absence of the mobile support infrastructure; Nokia Instant Community [11] allows mobile devices to form an ad-hoc network to exchange messages or share content.

Naturally any such service that allows the resources of some users (providers) to be used by other users (consumers) has to identify potential security and privacy threats and provide solutions to address them. In particular, providers need to have convenient means to specify suitable *access control*. Access control may be specified in terms of membership in a service (as is done by the community-based WiFi sharing service Fon). Another natural basis to specify access control is to share internet connectivity to "friends of friends", e.g., for visitors of an organization or guests at a party. Consumers need some level of privacy which has to be balanced against the providers' need for accountability so that providers would have evidence of resource usage by consumers.

Our Goal and Contribution. In this paper we present `CrowdShare`: a service design and its (Android) implementation that allows users to *share connectivity*. `CrowdShare` distinguishes itself from other tethering and mesh networking applications through incorporating a security architecture with privacy-preserving access control based on social relationships, pseudonymity for users, and accountability of usage. Although `CrowdShare` focuses on connectivity sharing, the architecture is generic and can be applied to resource sharing in general.

In summary our contribution is *design and integrated implementation* of a complete generic framework for secure resource sharing among nearby devices by incorporating a *security architecture* into existing technologies for mesh networking, tethering, and social network interfaces.

2 System Model and Requirement Analysis

System Model. The system model of the `CrowdShare` system is depicted in Fig. 1. It consists of a trusted `CrowdShare` server S, a social network server N, and a set of users $\mathcal{U}$. S admits the users to join the `CrowdShare` service, while N provides information about friend relationships among users. Each user $\mathsf{U}_i \in \mathcal{U}$ possesses a mobile platform which runs the `CrowdShare` application and enables communication of different users via the mesh network. A user U_i can play one of the following roles: (i) resource provider P, (ii) resource consumer C, or (iii) forwarding node F. P has access to (a set of) resources $\mathcal{R}$ and shares access to them with other users (e.g., Internet bandwidth, media files, or location information). P can restrict the access to his resources either to any U_i, or to a subset of users $\mathcal{F} \subset \mathcal{U}$, who are in a social relation with P in the social network (e.g., friends or friends of friends). C does not have direct access to $\mathcal{R}$ or $\mathcal{R}$ might be available but expensive, hence it consumes resources provided by P via the mesh network. Forwarding nodes F forward messages in the mesh network such that P and C can be connected over multiple hops.

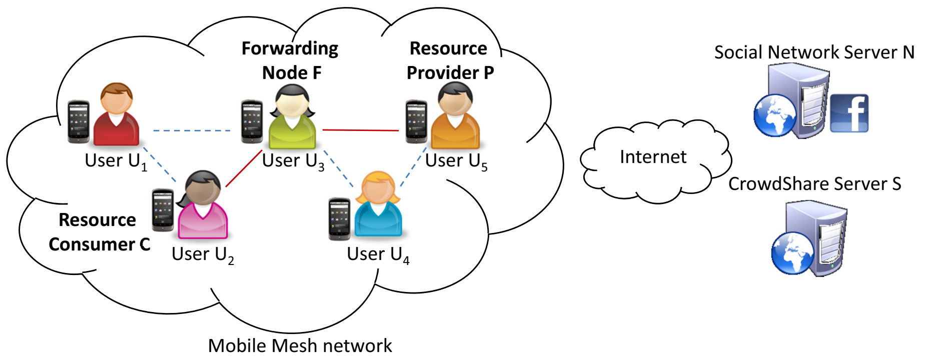

Fig. 1. `CrowdShare` system model

Threat Model and Security Requirements. `CrowdShare` and its infrastructure could be subject to several attacks. Our threat model does not cover any attacks against the operating system of the mobile device or any outside component, e.g., a remote server. Instead we concentrate on the attacks that users perform against the service itself. We focus on protecting against semi-honest adversaries that modify the `CrowdShare` service in order to learn sensitive information or get unauthorized access to services. We identify the following threats for `CrowdShare` which motivate the need for the respective security requirements.

1. **Man-in-the-middle Attacks ⇒ Channel Protection.** Devices in the ad-hoc mesh network should not be able to act as man-in-the middle that eavesdrops on or modifies messages that are routed through them. This motivates channel protection.
2. **Framing Attacks ⇒ Accountability.** C could use P's resources for illegal purposes. For instance, in the case of Internet sharing C could download a pirated song, leading the copyright owner of the song to accuse P of unauthorized use. In case of such violations, P needs the ability to give evidence that the resource was requested by a particular C. This motivates accountability.
3. **User Identification ⇒ Pseudonymity.** It should not be possible for a user to learn personally identifiable information such as the phone number or the email address of another user. This motivates pseudonymity.
4. **Unauthorized Usage ⇒ Access Control.** C should not be able to use P's resources without its consent. This motivates access control, i.e., P can attach a policy to the shared resource that needs to be fulfilled by C.

3 `CrowdShare` Protocols and Services

3.1 `CrowdShare` Protocols

Registration. The purpose of registration is to figure out a real user identity and to issue pseudonymous certificates which will be used by the `CrowdShare` community members for subsequent communication.

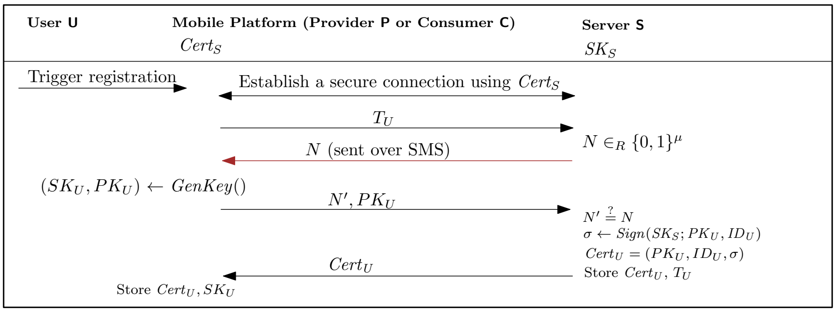
User U
Mobile Platform (Provider P or Consumer C)
$Cert_S$
Server S
SK_S
Trigger registration
Establish a secure connection using $Cert_S$
T_U
N (sent over SMS)
$N \in_R \{0,1\}^\mu$
$(SK_U, PK_U) \leftarrow GenKey()$
N', PK_U
$N' \overset{?}{=} N$
$\sigma \leftarrow Sign(SK_S; PK_U, ID_U)$
$Cert_U = (PK_U, ID_U, \sigma)$
Store $Cert_U$, T_U
$Cert_U$
Store $Cert_U, SK_U$

Fig. 2. Registration protocol

The registration protocol is depicted in Fig. 2. Fist, the user $\mathsf{U} \in \mathcal{U}$ establishes a secure channel to the server S using the certificate $Cert_S$ of S that is provided together with the `CrowdShare` application. Next, U sends the user's phone number T_U to S, who generates a one time password (OTP) N and sends it over the short message service (SMS) back to U. In turn, U generates an asymmetric key pair (SK_U,PK_U), and sends PK_U to S together with $N' = N$. Next, S verifies if the received N' matches N sent over SMS, generates a user certificate $Cert_U$ for this user, stores $Cert_U$ together with T_U and returns $Cert_U$ to U. The received $Cert_U$ is stored together with SK_U for future use.

The user's identity is verified, because S has the assurance that the submitted phone number belongs to the user, as he was able to receive the OTP N. S keeps the mapping between certificates and phone numbers secret and reveals it only to authorized entities, e.g., in case of a subpoena.

Provider Discovery. The goal of the provider discovery protocol is to discover a resource provider P which can share its resources with resource consumer C. The corresponding protocol is shown in Fig. 3. It is initiated when the resource request cannot be served locally (e.g., the request of the web-browser for the network connectivity cannot be served due to unavailable network connection).

First, $\mathsf{C} \in \mathcal{U}$ connects to a potential resource provider $\mathsf{P} \in \mathcal{U}$ (using mesh networking services) and establishes a secure (i.e., authentic and confidential) channel to it based on $Cert_C$ and $Cert_P$ (i.e., certificates obtained during registration). Next, C sends the resource request over the established channel along with the description of the resource R. If R is available, P responds with *policy* which specifies conditions for resource sharing. Particularly, *policy* may allow resource sharing with friends (or friends of friends) only, or require execution of the accountability protocol in order to protect P from framing attacks. If required by *policy*, P and C additionally use *Friend-of-Friend Finder* service (cf. §3.2) to identify friend relationships and execute the *accountability protocol.* If all conditions are met, P is added to a set of suitable provider candidates $\mathcal{P}$.

The protocol repeats n times to populate $\mathcal{P}$ with n candidates, where n is a configurable system parameter. The best candidate $\mathsf{P}^* \in \mathcal{P}$ is selected for

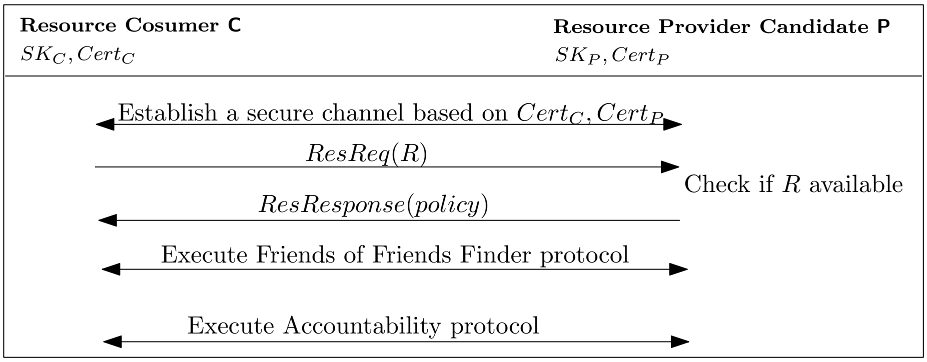
Resource Cosumer C
$SK_C, Cert_C$
Resource Provider Candidate P
$SK_P, Cert_P$
Establish a secure channel based on $Cert_C, Cert_P$
$ResReq(R)$
Check if R available
$ResResponse(policy)$
Execute Friends of Friends Finder protocol
Execute Accountability protocol

Fig. 3. Provider discovery protocol

resource sharing, while others are kept as back-ups. The availability of every $\mathsf{P} \in \mathcal{P}$ is monitored through listening to heart beat messages transmitted on a regular base. If any of them disappear, a new round of the provider discovery protocol is triggered to find a new candidate.

Accountability. Accountability is achieved by having C sign a resource quota request RQR that contains PK_C, the type of the resource R, and the resource leasing time τ. The resulting signature σ_{RQR} is sent to P^*, verified, and stored as an evidence.

Data Channel. A data channel is used for the delivery of the resource R from P^* to C. To provide confidentiality and authenticity to the data channel, we use standard techniques for setting up virtual private network (VPN) connections. Depending on the type of the shared resource, the VPN connection is either between C and P^* or S (e.g., between C and S for Internet connectivity sharing).

3.2 Friends-of-Friends Finder (FoF Finder) Service

The following server-aided approach allows to determine if two users P and C, are mutual friends or friends of friends in an existing social network.

During registration, each user authorizes S to access his friend list from the social network server N and to map the social network identifiers of the user's friends (and friends of friends) to their CrowdShare membership certificates. This mapping is sent to the registering device. During provider discovery, P checks if the certificate of C belongs to one of his friends or friends of friends by comparing the certificate identifier of C with identifiers in the friends database.

The server-aided solution requires each user to learn about entities in his social graph at hop lengths > 1, e.g., the number of friends of friends he has or their certificates which serve as pseudonyms. Depending on how users have set the visibility of their friend relations in the social network, this may be information that was otherwise not available to users.

As an alternative, we also allow P and C to determine common friends by running a private set intersection (PSI) protocol directly between them. The input to PSI is a set of "capabilities" that serve as proof of the friend relationship in the social network. We use a social network application as a generic secret distribution channel to exchange capabilities among friends. Due to space limitations we do not describe this approach in detail. The interested reader is referred to our technical report [4].

4 Security Considerations

In the following we provide an informal security analysis that demonstrates that the security requirements of §2 are fulfilled.

Channel Protection. For channel protection, all protocols are executed over a secure (i.e., confidential and mutually authenticated) channel. Particularly, the registration protocol runs over a channel where the server S is authenticated based on the server certificate $Cert_S$, while the user is authenticated by verifying user's phone number. The provider discovery and accountability protocols run over the secure channel established based on mutually exchanged certificates between P and C. The resource delivery is protected by a data channel established between C and P or C and S. The former is used in use cases which are not sensitive to eavesdropping by P, e.g., in case of file sharing (a file originating from P is already known to P). The latter is applied in case if P is a subject for confidentiality requirement, e.g., when sharing Internet connectivity (to ensure P cannot eavesdrop or manipulate traffic downloaded by C).

Pseudonymity. Pseudonymity is fulfilled by deploying pseudonymous user certificates which do not include any user specific information. The only entity which can map certificates to user identities is the server S, which is trusted to keep this information confidential.

Accountability. Accountability is satisfied by deploying an accountability service which protects P from framing attacks. The signed resource quota request submitted during the accountability protocol can be used by P as evidence toward possible misuse by C. Further, the signature can be mapped to a user identity with the help of the server S, which keeps the mapping between pseudonymous user certificates and user phone numbers. Hence, the real user identity can be traced back in case of illegal usage (e.g., accessing illegal content).

Access Control. We use two access control mechanisms: (i) membership-based access control which allows users to deny non-members access to resources of members, and (ii) social-relationship-based access control which allows users to grant access to friends or friends of friends.

5 Implementation

In this section we describe the implementation of the trusted server S and the mobile device which integrates functionality of the resource consumer C, resource provider P, and a relay device F in one.

Server. The server S provides the following main functionalities: (i) registration of new `CrowdShare` community members and (ii) a database that includes persistent information from other services (e.g., mapping from user identities to certificates). The functionality of S is implemented in Java 1.5 and stores objects in MySQL using the Hibernate framework. The implementation has 5 188 lines of Java code (LoC).

Mobile Platform. Our implementation targets Android-based devices. We used Google's Nexus One and the HTC Desire smartphones with Cyanogenmod 7.0 images for our development. The code is written in Java (for the API level 10) and makes use of a Bouncy Castle crypto library v. 147 (written in Java). For the implementation of cryptographic primitives, we used RSA 1024 and AES 128. We used standard SSL for the establishment of the secure channel used in provider discovery and OpenVPN (from the Cyanogenmod image) for the protection of the data channel. Our Android app has a modular design. Particularly, MeshNetwork is implemented as a separate component with well-defined interfaces which can be replaced when necessary or re-used in other applications. The overall implementation excluding the MeshNetwork component has 9 441 LoC.

We adapted the implementation of the Serval open source project [6] for the instantiation of the MeshNetwork component. Serval allows mobile devices to establish mesh networking on top of ad-hoc WiFi connections. It integrates BATMAN [10], a proactive distance vector routing protocol for wireless mesh networks. Further, it supports voice calls and text messages between mesh modes (hence, this functionality is also inherited by our implementation), but it does *not* provide Internet connectivity sharing and does *not* address possible security threats, which are our main focus.

Generally, stock Android devices cannot be configured to operate in WiFi ad-hoc mode without root access. Root access is required for loading a WiFi driver, configuring it to operate in ad-hoc mode and for configuring IP settings. However, root access is not required for the usage of our `FoF Finder` service. To support this claim, we implemented a simple (one-hop) tethering app which uses `FoF Finder` service for access control. The app uses Bluetooth to run `FoF Finder` protocols and WiFi for tethering and does not require root privileges.

6 Performance Evaluation

For our performance tests we used a HTC Desire device as a resource provider P and Nexus One devices for the resource consumer C and the relaying node F.

Multihop Re-transmissions. Fig. 4 illustrates the performance with and without multihop re-transmissions. To perform this test, we sent a ping packet to a remote server (*www.google.de*) and estimated the delay of the received response. The test was done for the direct Internet connection (i.e., no mesh re-transmissions are required), as well as for 1 hop and 2 hop indirect connections[1]. We sent 200 ping packets for each case. The delay increases with rising hop counts in the multi hop connection, which is reasonable, as each additional hop imposes additional packet delay due to re-transmissions. Further, the context switch between 3G and WiFi transmissions also adds overhead. The several peaks for the 2 hop tethering are imposed by packet loss and subsequent re-transmissions required to perform packet delivery successfully. To summarize, a hop count of 2 introduces a little delay in the range of milliseconds, which is acceptable.

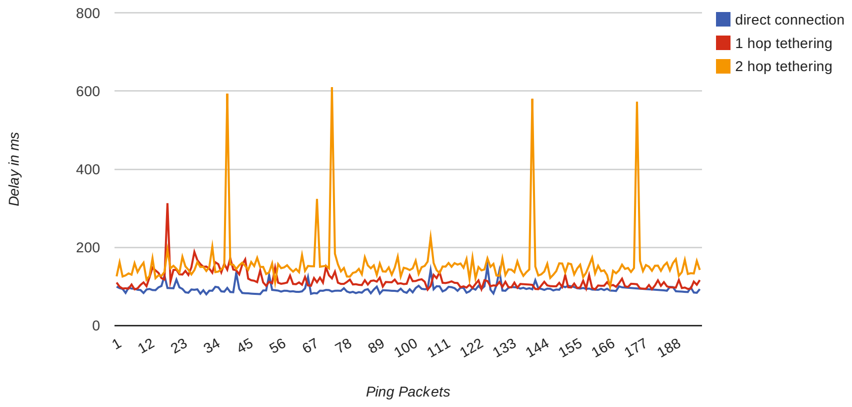

Fig. 4. Performance with and without multihop re-transmissions

7 Related work

VENETA [3] is a mobile social networking platform that, among other features, allows decentralized SMS-messaging via Bluetooth (up to 3 hops) and privacy-preserving matching of common entries in the users' address books using private set intersection. The combination of privacy-preserving profile matching and establishment of a secure channel was considered recently in [13]. Their solution allows a user to establish a shared key with another user only if their profiles match in a pre-determined set of attributes. Privacy-preserving discovery of common social contacts was considered in [5], where friends issue mutual certificates for their friendship relation. Our setting is different to all these works as we want to perform access control based on relationships in an existing social network.

[1] Our tests were limited by the number of available devices.

A number of projects developed ad hoc communication and resource sharing on top of mesh networks like Serval [6] and OpenGarden [1]. SCAMPI [12] provides generic discovery and a routing framework for opportunistic networks for developing versatile applications and services on top of it. Ad hoc communication has also found use cases in extreme situations where normal infrastructures are inaccessible, e.g., mines [7] and disaster-recovery scenarios [8]. In addition, we focus on privacy and security.

Acknowledgements. This work was supported in part by the German Federal Ministry of Education and Research (BMBF) within EC SPRIDE, and by the Hessian LOEWE excellence initiative within CASED.

References

[1] OpenGarden project, `http://opengarden.com/ourstory.php`

[2] Serval - comunicate anywhere, anytime, `http://www.servalproject.org/` (visited July 26, 2012)

[3] von Arb, M., et al.: VENETA: Serverless friend-of-friend detection in mobile social networking. In: WiMob, pp. 184–189. IEEE (2008)

[4] Asokan, N., Dmitrienko, A., Nagy, M., Reshetova, E., Sadeghi, A.-R., Schneider, T., Stelle, S.: Crowdshare: Secure mobile resource sharing. Technical Report TUD-CS-2013-0084, TU Darmstadt (April 2013), `http://www.trust.informatik.tu-darmstadt.de/publications/publication-details/?no_cache=1&tx_bibtex_pi1`

[5] De Cristofaro, E., Manulis, M., Poettering, B.: Private discovery of common social contacts. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 147–165. Springer, Heidelberg (2011)

[6] Gardner-Stephen, P.: The serval project: Practical wireless ad-hoc mobile telecommunications (2011)

[7] Ginzboorg, P., et al.: DTN communication in a mine. In: ExtremeCom (2010)

[8] Hossmann, T., et al.: Twitter in disaster mode: Security architecture. In: CoNEXT. ACM (2011)

[9] JoikuSpot (2007), `http://joikusoft.com/`

[10] Neumann, A., et al.: B.A.T.M.A.N.: Better approach to mobile ad-hoc networking. IEFT Draft (2008)

[11] Nokia. Nokia Instant Community. Article in Nokia Conversations Blog (May 2010), `http://conversations.nokia.com/2010/05/25/nokia-instant-community-gets-you-social/`

[12] Pitkänen, M., et al.: SCAMPI: Service platform for social aware mobile and pervasive computing. Computer Communication Review 42(4) (2012)

[13] Zhang, L., et al.: Message in a sealed bottle: Privacy preserving friending in social networks. CoRR, abs/1207.7199 (2012)