Establishing Distributed Hidden Friendship Relations

Sören Preibusch and Alastair R. Beresford

University of Cambridge · Computer Laboratory
William Gates Building, 15 JJ Thomson Avenue
Cambridge CB3 0FD, United Kingdom
{Soren.Preibusch|Alastair.Beresford}@cl.cam.ac.uk

Abstract The social Web is going mobile and needs support for friendship management in a distributed manner, while privacy concerns mandate configuring the public visibility of one's friends. In this paper we leverage existing Web standards to describe a simple P2P protocol for establishing, enforcing, and revoking hidden friendship relations and report on an implementation for a mobile platform. We examine the suitability of hidden friendship links for bilateral and delegated access control and discuss how the social connotation of friendship can be preserved when concealing the friend's identity.

1 Online Friendship Relations and Privacy

The use of on-line and social networking websites is growing, and social interaction through such systems is now part of the daily routine for many individuals [4]. Whilst the development of social networking on the Internet originated from the desire to allow individuals to update their friends or colleagues with new personal or professional information, social networking techniques are now being used to enhance the performance of many other Internet services. For example, friendship links have improved Internet search content indexing and ranking by using HTTP requests made by friends [9]; Tribler [11] is an overlay on top of BitTorrent [10], allowing users to establish friendship links and form groups in order to increase download speed or improve content discovery.

Positive privacy of friendship relations. Once established, the friendship relation often confers additional rights or capabilities to friends, such as the ability to view personal photographs or send private messages (first use-case, UC1). In this context, friendship links on social networks are seen as privacy enhancing, since they restrict access to personal information. Unfortunately, the controls available to limit access to personal data or enhanced services are often quite primitive. Most sites allow users to restrict access to personal information to friends (of first or higher degree) and some sites permit permission to be configured at an individual level. There are only few sites which allow users to privately group individuals together and apply access control at the group level directly;

such capabilities are required to provide more generic rôle-based access control facilities.

As online social networking sites increasingly become platforms on which relationships are setup rather than merely replicated from the offline world, networks provide trust metrics to guide the users in assessing other members' credibility. Despite its limitations, a user's friends count is used as a simple yet intuitive metric in contexts such as casual dating, business contacts, or electronic commerce (UC2).

Friendship links may also be used by the information consumer for incoming filtering rather than for outgoing filtering by the information producer. The access control function of friendship is replaced by an information overflow prevention function. For instance, user agents may only process broadcasted events such as status updates that originate from known friends; Sunday shoppers may only enjoy receiving promotional offers from stores that are on their favourites list (UC3).

The gate-keeping function of friendship relations is not restricted to pairwise encounters, but can be extended to multihop authorisation based on functional properties of the links, including but not limited to transitivity: friends of friends or, more generally, friends of n^{th} degree may enjoy privileged access (UC4). Active consent of the original information holder or of the involved middlemen may be required for successful privilege propagation (UC5).

Negative privacy of friendship relations. The set of attributes and identities linked with a friend's online profile introduce a privacy-endangering facet into friendship relations. Having a friendship relation with somebody may be socially detrimental. For example, investigating journalists have a professional interest in keeping their sources secret (UC6). Executive professionals may wish to maintain secret ties with friends working at competitor companies (UC7). And teenagers may feel peer group pressure in choosing who they call friends (UC8). Pitfalls also arise within the online network itself when the number and kind of friends positively and negatively influence one's social status (UC9). There are also potential negative consequences outside the social realm. Companies rely on the formalised nature of friendship relations to mine connections between users along which personality traits and socio-economic characteristics are assumed to propagate. A potential employer may refuse a candidate because of her friendship with others; and users may receive targeted advertisements based on preferences and interests their friends publicised on the network site.

The nature of the friendship relation on social networking sites therefore requires the ability to hide friendship relations. Simply hiding all of one's friends does not solve the problem, because it denies the advantages of public friendship and ignores the symmetry of friendship relations, implying that either of the involved parties may reveal the existence of a link independently. In analysing an existing social network, the authors found that more than two thirds of users who chose to conceal friendships actually had exposed at least one of these supposedly hidden relationships [12].

Mobile networking. As the performance and capability of mobile phones increases, such devices increasingly host social networking applications. This movement provides richer (yet intermittent) connectivity, encourages greater levels of data entry, and allows the automated collection of sensor data, such as location information. Intermittent connectivity encourages more application state and functionality to reside on the device itself, rather than a remote server, and it is for this reason we believe that a move to decentralised social networks will occur. In the long term, mobile devices may function without any centralised facilities at all. Brief encounters amongst humans will trigger ad-hoc connectivity between devices. For instance, human mobility and opportunistic short-range networking may allow social networks to be built on top of delay-tolerant networks, for which, in turn, stable human connectivity traces suggest reliable routing paths [6, 16]. Also, a decentralised scheme potentially provides better privacy guarantees, since trusting social network operators is no longer a prerequisite—data are kept on the device under the control of the individual.

Our contribution. The contribution of this paper is twofold. Building on previous research into the architecture of hidden friendship relations [12], we propose a simple protocol for establishing, enforcing, and revoking selectively hidden friendship relations in a P2P scenario. In addition, we describe an implementation of this protocol for mobile devices, providing details of the user interface and on the integration with the phone's existing messaging and contact management facilities. We review our protocol with regard to security and functional requirements, to resource consumption, as well as to standard compliance. Based on nine use-cases, we examine the suitability of hidden friendship links to convey privileges and we discuss how the social connotation of friendship can be preserved when concealing the friend's identity.

2 Hidden Friendship Relations Protocol

Deployment scenario and protocol requirements. In a centralised scenario, hidden friendship links can easily be implemented by the central network server removing hidden friends from its response when serving a user's list of friends, based on the credentials the requesting client presents. In particular, the network operator is in a position to evaluate any credentials with regard to a strong identity since the user is typically session-authenticated. In an otherwise secure system, it is unlikely that user B could pose for A when presenting (replaying) one of A's credentials.

In a distributed scenario, however, checking the credentials of the requesting user represents a server-like task, implying that continuous connectivity must be maintained, which is incompatible with the assumption of intermittent connectivity and prohibitively resource-consuming for mobile devices.

One of the design challenges, which occurs when removing a central authority, becomes the lack of a strong yet simple proof of identity. A traditional decentralised public key infrastructure, such as GPG, or the web of trust imply a social graph in which identity can be mined to the detriment of *hidden* friendship.

Design goals and requirements. In order to support a distributed social network, it is desirable if a user's list of friends is immutable with respect to requests being made by different parties. A distributed hidden friendship protocol is further expected to fulfil the following design goals: (a) users should be able to selectively hide a self-chosen subset of their friends; (b) a friendship relation, whether hidden or public is symmetric; (c) users should be able to establish, to revoke, and to set the visibility of their outgoing friendship links uni-laterally, i.e. without coordination efforts; (d) a friendship is public iff both friends make it public and it is hidden iff both friends hide it; (e) a friend B of user A can check whether their friendship still holds by inspecting A's list of friends; (f) everybody should see public friendship links, and nobody except the involved parties should be able to infer a hidden friendship from either list of friends; (g) the establishment of a friendship relation requires the consent of both parties; (h) hidden and public friendship links are both made public, i.e. hiding a friendship link comes not from concealing its existence (no security through obscurity).

It is outside the scope of the protocol to specify what leads to establishing a friendship. This preceding interaction pertains to the social sphere.

Threat model. We outline the security goals and the threat model for a hidden friendship in general and its distributed deployment in particular. The fundamental notion is user A calling user B a hidden friend. This shall be manifested with an encrypted entry E_{AB} in A's public list of friends. The entry can be accessed by any other user since the list of friends is public. However, it shall not be possible for a non-related third-party $\mathrm{X} \notin \{\mathrm{A},\mathrm{B}\}$ to learn who B is from the entry in the list of friends. In particular, X is unable to locate the corresponding list of friends in which E_{BA} should be listed in case the hidden friendship actually exists through symmetrically calling one another a hidden friend.

The following assumptions are made with regard to a friend entry E_{AB}: (a) $\mathrm{E}_{\mathrm{AB}} \neq \mathrm{E}_{\mathrm{BA}}$ so that the symmetry of a hidden friendship is not obvious; (b) $\mathrm{E}_{\mathrm{BA}} \neq \mathrm{E}_{\mathrm{CA}}$ so that two users having a common friend is not visible in the friend list; (c) a friends list entry corresponding to a hidden friendship can be told apart from a non-hidden friendship link; (d) the validity of E_{AB} cannot be established by X, i.e. X cannot distinguish a real friends list entry from random data made up by A. It is assumed to be beyond the capabilities of an attacker to compute E_{BA} from E_{AB} and to infer B from E_{AB}.

Regarding the integrity of a list of friends, we assume that a user has sole control over her public list of friends and that requests to this list can be made in a secure manner. A potential attacker has the following capabilities: (a) monitoring traffic of users; (b) monitoring changes in the published lists of friends; (c) re-publishing hidden or public friendship entries found in other users' lists of friends.

Regarding the system environment, the following assumptions are made: (a) devices that offer tool support for distributed hidden friendship links have sufficiently synchronised clocks to assess lifetime expiry of documents such as friendship lists. Moreover, these devices experience periods of lost connectivity. We assume that (b) users checking for the existence of a device or a user do not immediately conclude the non-existence of a user from her non-reachability. As devices may cache friends lists, these cached versions may be outdated. No means for verifying cache expiry exists at periods of lacking connectivity, during which friendship revocation may occur. We, therefore, further assume that (c) users may rely on possibly outdated cache copies, (d) will implement safeguards such as short expiry times for critical applications, and (e) further delay the execution of critical friend lookups until connectivity is restored or cache validity has been established.

Further design goals specific to social networking applications. In addition to the design goals outlined above (p. 4), and for the purpose of sensible social networking applications, an additional property is desirable: any X can learn the number of hidden friends A claims to have by counting the entries for hidden friendship links. Such a requirement is in line with the use-cases UC2 and UC9. However, this further requirement conflicts with the design assumption (d) established above that validity of one's friends list entry is undecidable for an outsider. The remedy of counting one's inbound friendship links which should equal one's outgoing friendship links under the symmetry assumption is not possible for hidden friendship links. A well-formedness requirement may be another practicable approach to tell bogus entries and valid friendship entries apart. Still, the countability goal is not be achievable by withdrawing the assumption (d) only. Any user may have several identities that she could use for establishing friendship links between seemingly different users. There are therefore at least two techniques to boost one's hidden friends count, each of which cannot be precluded in a fully distributed scenario where identities can be created opaquely at low cost. We conclude that X can only learn a lower bound for A's number of hidden friends from latter list of friends. The satisfiability of weakened countability goals is discussed in Section 2, p. 9.

Social networks also rely on friendship as an access control criterion as described in use-cases UC1 and UC4. The existence of a friendship relation is enforced when accessing a secured resource. We distinguish between two cases: first, direct enforcement and, second, delegated enforcement. Bilateral enforcement relies on the design goals and is achieved since both parties involved in a hidden friendship can verify its continued existence in their own and the respectively other's records. A third party tests the existence of a friendship between two users by pairwise associating a friendship claim with a proof of identity.

Granting access to privileged resources may not be limited to one's direct friends but also friends of friends or, more generally, friends of n^{th} degree. We discuss the compatibility of hidden friendship links with multihop friendship in Section 2, p. 9.

Leveraging existing technologies. The implementation of a hidden friendship protocol, discussed in the next Section on p. 6, can be built on top of existing Internet and security technologies, keeping the protocol itself concise while leveraging established and developing standards. In particular, the following infrastructure for publishing personal information in a semantic format and for transmitting information in a secure and concealed manner is used as a basis: (a) the vCard format extension to represent social network membership information for a single individual, in particular the publisher of the vCard file [5]; (b) the representation of latter in semantic HTML using hCard for direct embedding into personal or other Web pages, including distributed social network profile pages [1]; (c) the FOAF (friend-of-a-friend) standard to encode personal information and relationships in a machine-readable format; (d) FOAF+SSL, as an alternative to OpenId to allow for certificate-bound identities and distributed authentication across multiple social networks [15]; (e) a contact list private to the user's individual, such as a phone book, Outlook contacts or a private FOAF file; (f) means for encrypting messages and for concealing messaging interaction, using remailers such as Mixminion [3].

Implementation: establishing friendships. Public and hidden friendship links are both stored as outgoing friendship links, mutually pointing to the other party (Figure 1). For *public friendship links*, the link references the other user by her public identifier, $\mathrm{K_A}$,such as a public key, a well-known personal Web page Uri, an email address or the Uri of her own list of friends. For *hidden friendship links*, the link references a public identifier specific to this relationship only and not otherwise used by either of the parties. Each party may freshly generate such an identifier as a (public key, private key) pair, unique to a directed friendship link. The referenced party maintains the private key K_i^{-1} associated with the public key K_i in the referencing party's public list of friends. Note that there is no need for sharing a secret. If secure channels are used for concealing the messages exchanged between two prospective friends, public/private keys used therein may be recycled.

Enforcing friendships. Friendship relations are enforced upon execution of a privileged action. A *public friendship relation* between A and B is enforced by A inspecting B's list of friends for the existence of $\mathrm{K_A}$. B looks for $\mathrm{K_B}$ in A's public list of friends. In addition, any third party X may also check for the existence of a public friendship relation between A and B. A *hidden friendship relation* between A and B is enforced by A inspecting B's list of friends for the existence of the relationship-specific public key K_i that A once issued to B. Likewise, B looks for the relationship-key K_j she has issued to A. A third-party X is unable to check for the existence of a hidden friendship relation between A and B, since the relationship-specific keys do not contain a reference to the issuing party. This behaviour is intentional, as detailed in use-cases UC6 and UC7.

Revoking friendships. Friendship relations can be revoked unilaterally by either party removing the corresponding person- or relationship-specific public

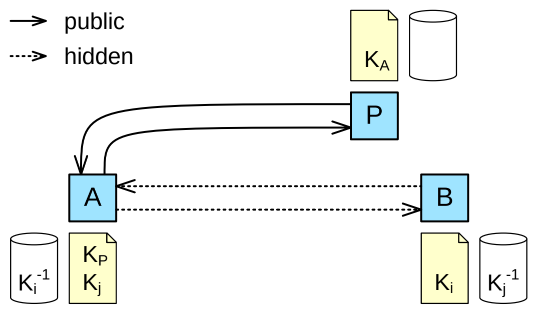


Figure 1. Encoding of public (solid lines, A $\leftrightarrow$ P) and hidden (dashed lines, A $\leftrightarrow$ B) friendship relations with symmetric links. The list of friends (document) associated with a user is public, the individual contact list (database) is private. For a public friendship, the friends store the respectively other's public identifier in their own list of friends: A stores K_P and P stores K_A. For a hidden friendship, the friends store a public key issued by the respectively other for this friendship relation only. A generates a key-pair (K_i, K_i^{-1}): she keeps the private part and sends the public part to B who incorporates it into her public list of friends. B also generates a key-pair (K_j, K_j^{-1}) and performs the same exchange and integration tasks (Figure 2).

key from her public list of friends. Revocation can be detected through unsuccessful enforcement. A unilaterally revoking party continues to have access to friendship-secures resources until the other party replicates the revocation.

Encoding in FOAF. FOAF is an established semantic Web format one can use to publish information about oneself and one's connections in standard and machine-readable manner. FOAF files are published by their author. In FOAF, `foaf:knows`-elements indicate relationships with other people. More specific variants, such as "closeFriendOf", "livesWith" or "parentOf" as exist as inheriting elements to further specify the quality of the relationship [2]. Such derived properties can be used to make the distinction between public and hidden friends explicit. However, this is not required because storing relationship-specific keys instead of person-specific identifiers is compatible with the syntax of FOAF. `foaf:nickname` or other RDF vocabulary may be used instead of `foaf:name` or `rdf:resource` may also be used to qualify the nature of the referencing identifier rather than of the reference itself.

For the friendship relations depicted in Figure 1, the relevant fragment from A's FOAF file could be:

```
<foaf:Person rdf:ID="me">
  <foaf:name>A</foaf:name>
  <foaf:knows><foaf:Person>
    <foaf:name>P</foaf:name>
  </foaf:Person></foaf:knows>
```

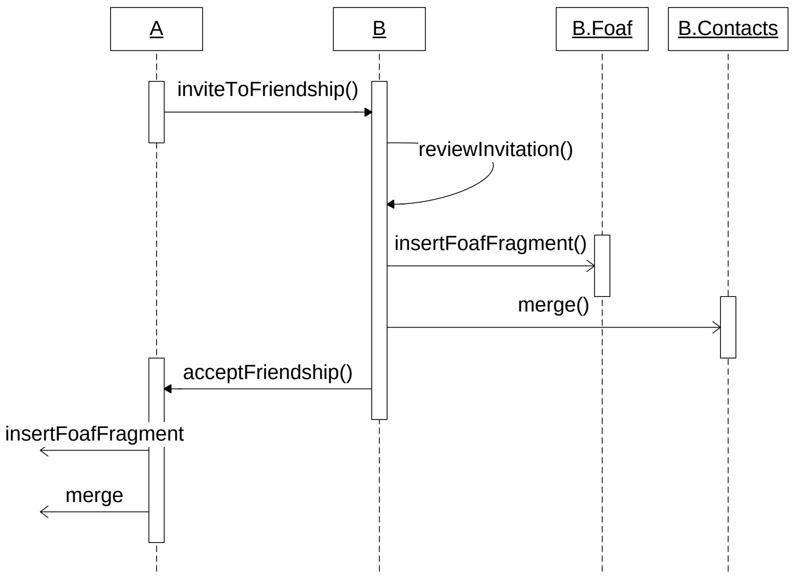


Figure 2. Generic sequence diagram for a successful establishment of a friendship relation. User A initiates the establishment by sending a friendship invitation to user B. A includes an identifiying fragment, a public key specific to this friendship link for a hidden or her persistent public key (or otherwise identfier such as a Uri) for a non-hidden friendship. Her details such as email address or other communication details are included as sender information. User B reviews the friendship request and, upon approval, inserts A's identifying fragment into her own list of friends, such a B's public FOAF file (B.Foaf)). B also updates her private contact database (B.contacts) with A's details and a pointer to A's public FOAF file for future friendship enforcement. B then replies to A with B's own identifying fragment (friendship link specific / general purpose public key). Finally, A performs the same integration tasks, by the end of which the friendship between A and B is established.

```
  <foaf:knows><foaf:Person>
    <foaf:name>ae4f281df5a5d0ff3cad6371f76d5c29b6d953ec</foaf:name>
  </foaf:Person></foaf:knows>
</foaf:Person>
```

The rôle of a central authority / delegated access control with distributed hidden friendships. As outlined in use-case `UC1`, friendship links form the basis for social access control schemes. In particular, a user may publish a digital resource such as the photographs from the wedding or a video from the last rafting trip. Only friends shall have access to this resource. Typically, A does not want to host this material herself but relies on dedicated service providers to fulfil this server-like task since these offer continuous connectivity and have sufficient bandwith and storage resources. Several social networks have grown around such media publishing. The idea of centralised content storage is compatible with and complementary to decentralised friendship control. Figure 3 depicts a scenario where A publishes a restricted document using C's services and B, a hidden friend of A, wants to access the document. C shall be able to check whether B qualifies for access without having A to reveal her friends to C. Delegated access control is a major motivation for uniformly publishing hidden and non-hidden friends.

Access to a secured resource will be determined based on a list of friends which is referenced through a Uri as part of the upload process. Linking to the friends list instead of physically uploading it implies that changes in A's list of friends will be reflected in C's decisions who shall have access with no need for A to alter the uploaded content/ACL bundle. A may want to specify a resource-dependent caching policy which allows C to base its decisions on local copies of A's list of friends. Setting such caching policies is perfectly feasible when requests to the Uri of A's FOAF file are served over HTTP(S).

As soon as friendship enforcement relies on a cached list of friends relayed by third parties, the authenticity of the caches needs to be sufficiently reliable. Signing FOAF files guarantees their integrity and can be done using standard XML Signature procedures. To ensure that the signer actually is the original publisher (and not an attacker), the signature key needs to be linked sufficiently strongly with the FOAF file author. Semi-centralised architectures such as institutional trust provide strong evidence as can a web of trust provided there are enough public friendship links. Note that once established, the belief in a signature key can be passed on over updates of the signed document.

A public friend is trivially granted access since anybody can enforce a public friendship relationship. A hidden friend can prove her entitlement without revealing her identity: access shall be granted to a requestor who can prove to have the private key associated with any public key located in the uploaders public FOAF file. This challenge-response can be done using standard procedures. It is not important under which hidden friends' public key a requestor authenticates since all hidden friends have access anyway. A leaked private friendship key is just the digital analogy to an unreliable friend.

Counting hidden friendships. As outlined in use-cases UC2 and UC9 counting one's friends is another typical social networking application. Given a user's list of friends, one can only tell how many hidden friends this user has at most (see p. 5). However, in the context of an access control scenario as the one outlined above, the content distributor C could publish statistics on how many successful authentications against different public friendship keys were made (still being unable to tell how many distinct users authenticated). A user may use a similar infrastructure to propose a "vote of confidence" and ask her friends to support her – without unveiling their identity.

Hidden friendship of n^{th} *degree.* As outlined in use-case UC4, privilege propagation beyond direct friendship links may be desirable. Users can identify public

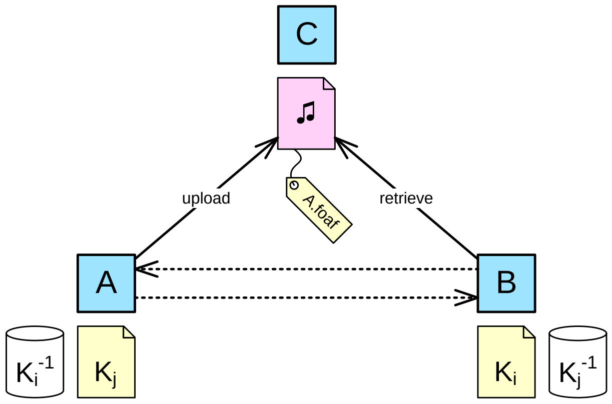
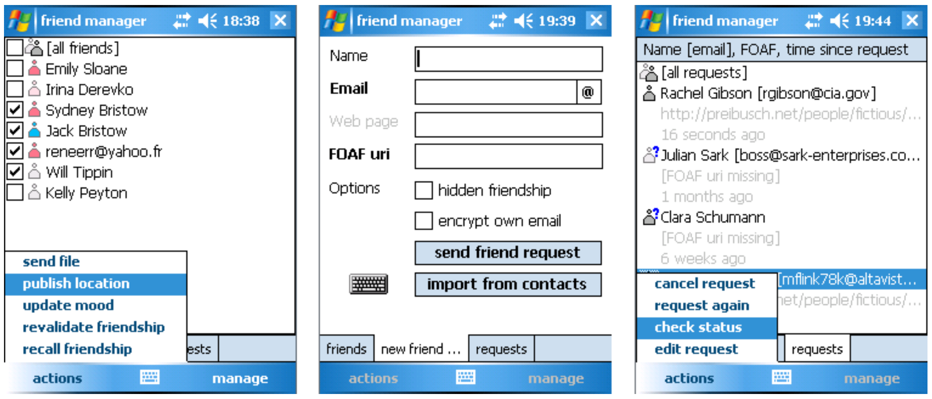


Figure 3. A publishes a digital resource using C's facilities: she uploads a document and attaches to it the Uri of her FOAF file, which shall determine who will have access to the resource. B, a hidden friend of A, then wants to access the resource.

friends (of public friends) of their own hidden friends. Verification of hidden friendship links further down in a linking sequence can be achieved on a per-case by applying the third-party enforcement / delegated access control model. As motivated in use-case UC5, successful enforcement of multihop friendship links requires cooperation from hidden friends along the friendship path. If, for instance, there was a path of friendship relations A $\leftrightarrow$ B $\leftrightarrow$ C $\leftrightarrow$ D, all of them hidden, D would require (only) C's support for proving to A there is a friendship link of third degree. In general $n-2$ cooperating intermediate friends are needed to to directly prove a hidden friendship of n^{th} degree and $n-1$ are required if A delegates the enforcement.

One-party deviation from the protocol. In a distributed friendship scenario, symmetry of friendship links is not enforced. As a result, each of two parties involved can change the friendship links in their FOAF files unilaterally. For a public friendship link, un-symmetric friendship links are obvious and may just be interpreted as unilateral appraisal ("fan") or as a transitionary period during which one party has already established a link and is waiting for the other party to follow or one party has revoked the friendship.

The interpretation of unilateral hidden friendship relations is more subtle. A party deviating from the protocol may alter the visibility status of an outgoing friendship link or delete it. For all scenarios, it holds that the other party may detect such changes in the link structure to at least the same extent as a third party observer. Unilateral alteration of friendship visibility does not change the publicity of the relationship but breaks the relationship (as does revocation). The other party may respond by a corresponding action.



[all friends]
Emily Sloane
Irina Derevko
Sydney Bristow
Jack Bristow
reneerr@yahoo.fr
Will Tippin
Kelly Peyton

Name
Email
Web page
FOAF uri
Options
hidden friendship
encrypt own email

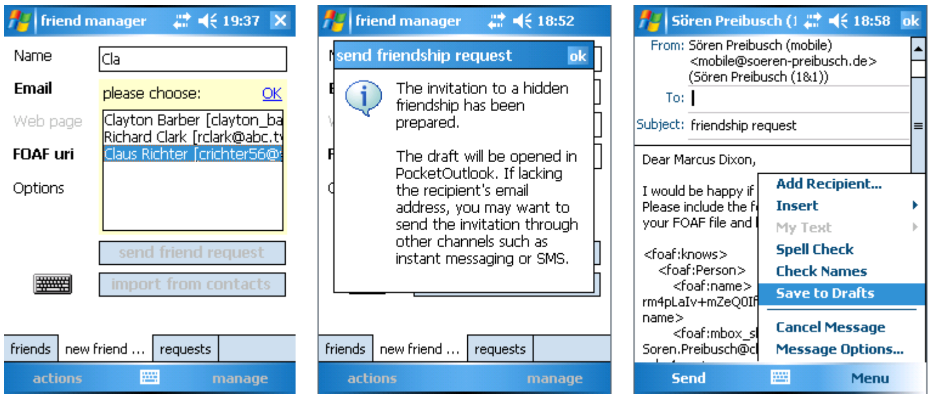
Name [email], FOAF, time since request
[all requests]
Rachel Gibson [rgibson@cia.gov]
http://preibusch.net/people/fictious/...
16 seconds ago
Julian Sark [boss@sark-enterprises.co...
[FOAF uri missing]
1 months ago
Clara Schumann
[FOAF uri missing]
6 weeks ago
[mflink78k@altavist...
net/people/fictious/...

Figure 4. Using a tabbed user interface, the "friend manager" application gives a list of the existing friends, provides a form for generating new friendship requests, and allows a review of pending requests. **(left)** If a friend can be matched to the user's contacts, a real name is displayed instead of a communication identifier (e.g. reneerr@yahoo.fr). Shade and colour of the icons communicate the visibility of the friends (light: hidden; dark: publicly visible) and another property such as sex, which may be not available for some friends (grey icons). Multiple friends can be selected at once to perform a privileged operation such as informing them about one's current location. **(middle)** The form for creating new friendship invitations offers options for enhancing the privacy of the friendship. **(right)** Currently pending requests are summarised with the name of the potential friend, her contact details, and her FOAF URI if available. The visibility of the friendship is encoded in the icon. The time elapsed since the request was filed is displayed in a human-readable format. Requests with the FOAF URI missing are highlighted since their status cannot be checked. Filed requests can be edited to add such missing information at a later point in time.

3 Tool Support for Mobile Devices

The feasibility and sustainability of the developed hidden friendship protocol is demonstrated by a mobile-device platform implementation of a "friend manager" application for which Figure 4 depicts the core functionality. The application builds on the Microsoft .Net Compact Framework, allowing deployment on devices running Windows [8], Symbian [14], or other various operating systems such as Linux on diverse architectures, Apple operating systems, SUN Solaris, and non-mainstream operating systems (for instance Nintendo Wii) via Mono.

The friend manager tightly integrates with the existing PIM infrastructure for messaging and contact management, Figure 5, hence improving usability and avoiding overlapping data silos. Close integration is of paramount importance for user acceptance as the contact book is the private manifestation of a person's social network and a source for finding opportunities for communication and interaction [13].



Name: Cla
Email
Web page
FOAF uri
Options
please choose: OK
Clayton Barber [clayton_ba
Richard Clark [rclark@abc.t
Claus Richter [crichter56@

send friendship request

The invitation to a hidden friendship has been prepared.

The draft will be opened in PocketOutlook. If lacking the recipient's email address, you may want to send the invitation through other channels such as instant messaging or SMS.

From: Sören Preibusch (mobile) <mobile@soeren-preibusch.de> (Sören Preibusch (1&1))
To:
Subject: friendship request

Dear Marcus Dixon,

I would be happy if
Please include the f
your FOAF file and

<foaf:knows>
<foaf:Person>
<foaf:name>
rm4pLaIv+mZeQ0IF
name>
<foaf:mbox_s
Soren.Preibusch@c

Figure 5. The "friend manager" application integrates with the existing messaging infrastructure on the device. **(left)** Upon creating a new friend request, user details can be imported from the contacts so that contacts matching in name will be displayed and updated as the user types. **(middle, right)** Friendship invitations are sent via PocketOutlook where a draft is generated for further edit prior to sending the message so the user can benefit from the entire existing messaging infrastructure. The application is prepared to handle alternative communication channels in addition to email.

4 Discussion

The proposed protocol of encrypting identifiers for hidden friendship relationships presents several theoretical and technical advantages which are particularly valuable for deployment in distributed social networking. Public and hidden friendship relations can be encoded uniformly in a public FOAF file with no need to serve different friends lists based on the credentials a requesting client presents. Hidden friendship identifiers cannot be traced back to the actual users and several hidden friendship links of the same user cannot be merged. Semantics and syntax of FOAF are preserved as well as the various alternatives for publishing FOAF files – including third party repositories and distributed caching.

Hidden friendship relations are compatible with typical social networking applications such as direct or delegated access control schemes based on friendship. Friendship enforcement can be realised for multihop links. The number of one's outgoing hidden friendship links is an unreliable social metric since these links can be made up easily. Still, it is possible to anonymously count one's encrypted friend entries that are not bogus data.

The proposed protocol builds on top of encryption and the concealment of the friendship-establishing messaging. Nonetheless, should an encrypted identifier be intercepted, it cannot be used to establish a hidden friendship under a false name. However, challenges remain:

- Does the observable execution of a privileged action reveal enough information to detect the existence of a friendship link?
- Can users attach an individually negotiated privacy policy to a friendship link to govern the use of data received over this friendship channel?
- Is access control based on friendship relations of n^{th} degree as sketched in this paper viable?
- To which extent can one leverage hidden friendship networks for anonymising the retrieval of Web resources?
- Can we build usable rôle-based access-control (RBAC) schemes for social networking?
- From an economic perspective what level of support for privacy settings does a social network operator optimally offers to its users? Do social network operators compete on privacy support anyway?
- How can metrics over hidden and public friendship relations still be computed in a useful manner?
- Is an escrow service (for instance dbpedia) required in order to ensure the same list of friends is presented to all users?
- Does the implicit understanding of what a friendship relation conveys differ across different networks to such an extent that joining friends throughout several networks would contravene the users' access control intentions?

References

1. Tantek Çelik; Brian Suda. *hCard.* `http://microformats.org/wiki/hcard`
2. Ian Davis; Eric Vitiello Jr. *RELATIONSHIP: A vocabulary for describing relationships between people.* `http://purl.org/vocab/relationship/`
3. George Danezis; Roger Dingledine; Nick Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. *IEEE Symposium on Security and Privacy*, 2003. `DOI>` 10.1109/SECPRI.2003.1199323
4. Martin Fisch; Christoph Gscheidle. Mitmachnetz Web 2.0: Rege Beteiligung nur in Communitys. *Media Perspektiven* **7**:356-364, 2008.
5. Robins George; Alexey Melnikov. *vCard Format Extension : To Represent the Social Network Information of an Individual.* IETF, Internet-Draft, 2009 `http://www.ietf.org/internet-drafts/draft-george-vcarddav-vcard-extension-00.txt`
6. Pan Hui; Jon Crowcroft; Eiko Yoneki. BUBBLE Rap: Social-based Forwarding in Delay Tolerant Networks. *9th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, 2008. `DOI>` 10.1145/1374618.1374652
7. Microsoft Corporation. *Microsoft Launches MSN Messenger Service: Online Instant Messaging Service Enables Consumers to Communicate With More People Than Any Other*, 1999. `http://www.microsoft.com/presspass/press/1999/jul99/messagingpr.mspx`
8. Microsoft Corporation. *Devices and Platforms Supported by the .NET Compact Framework*, 2009. `http://msdn.microsoft.com/en-us/library/ms172550.aspx`
9. Alan Mislove; Krishna P. Gummadi; Peter Druschel. Exploiting Social Networks for Internet Search. *5th Workshop on Hot Topics in Networks (HotNets'06)*, 2006. `http://www.mpi-sws.org/~amislove/publications/PeerSpective-HotNets.pdf`

10. Johan Pouwelse; PawełGarbacki; Dick Epema; Henk Sips. The Bittorrent P2P File-Sharing System: Measurements and Analysis. *Lecture Notes in Computer Science* **3640**:205-216, 2005. DOI> 10.1007/11558989_19
11. J. A. Pouwelse; P. Garbacki; J. Wang; A. Bakker; J. Yang; A. Iosup; D. H. J. Epema; M. Reinders; M. R. van Steen; H. J. Sips. TRIBLER: a social-based peer-to-peer system. *Concurrency and Computation: Practice and Experience* **20**(2):127-138, 2008. DOI> 10.1002/cpe.1189
12. Sören Preibusch; Alastair R. Beresford. Privacy-Preserving Friendship Relations for Mobile Social Networking. *W3C Workshop on the Future of Social Networking*, 2009. http://www.w3.org/2008/09/msnws/papers/Preibusch-Beresford_Privacy-Preserving-Friendship-Relations.pdf
13. Mika Raento; Antti Oulasvirta. Designing for privacy and self-presentation in social awareness. *Personal and Ubiquitous Computing* **12**(7):527-542, 2008 DOI> 10.1007/s00779-008-0200-9
14. Red Five Labs. *Net60 - Opening Symbian devices to .NET development*, 2009. http://www.redfivelabs.com/content/datasheet.aspx
15. Henry Story. FOAF & SSL: creating a global decentralised authentication protocol. *W3C Workshop on the Future of Social Networking*, 2009. http://www.w3.org/2008/09/msnws/papers/foaf+ssl.html
16. Eiko Yoneki; Pan Hui; ShuYan Chan; Jon Crowcroft. A Socio-Aware Overlay for Publish/Subscribe Communication in Delay Tolerant Networks. *10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems*, 2007 DOI> 10.1145/1298126.1298166