UnLinked:
Private Proximity-based Off-line OSN Interaction

Sky Faber
Computer Science Dept.
UC Irvine
fabers@uci.edu

Ronald Petrlic
Commission for Data Protection
Baden-Württemberg, Germany
petrlic@lfd.bwl.de

Gene Tsudik
Computer Science Dept.
UC Irvine
gene.tsudik@uci.edu

ABSTRACT

The recent decade has witnessed a rapid increase in popularity of mobile personal devices (notably, smartphones) t hat function as all-purpose personal communication portals. Concurrently, On-line Social Networks (OSNs) have continued their impressive proliferation. Meanwhile, the notion of "OSN privacy" remains elusive and even self-contradictory. Centralized nature of prominent OSNs is unlikely to change, which does not bode well for OSN users' privacy. However, some user privacy can be gained from making certain OSN functionality available **off-line**, such as discovering common contacts and other features, as well as establishing affinity-based connections. OSN providers stand to gain from this, since users could avail themselves of OSN functionality in scenarios where none currently exists, e.g., whenever Internet connectivity is unavailable, expensive or insufficient. At the same time, OSN users benefit from increased privacy because off-line interactions can be made opaque to OSN providers.

This paper explores off-line private proximity-based use of OSNs. Although our approach is quite general, the proposed system (called *UnLinked*) is grafted atop a specific and popular OSN – *LinkedIn*. One key challenge is how to ensure authenticity and privacy of users' information (e.g., connections and other profile data) when they engage in off-line interactions. This is addressed by designing an efficient technique for authorized two-way private set intersection (ATW-PSI), which allows two OSN users to jointly learn **only** the intersection of their input sets, while being assured of the authenticity of each others' input. The paper describes and evaluates a practical prototype that allows physically proximate LinkedIn users to commit to a connection if they have a mutually acceptable number of common connections.

1. INTRODUCTION

Building trust in previously unfamiliar people based on common factors – such as interests, backgrounds, friends, or co-workers – has been practiced by the human race since time immemorial and has been thoroughly studied as a subject. In the last decade, due to the popularity of on-line social networks (OSNs), the process of finding and connecting to other people (based on something in common) has become easier due to OSNs' integrated search functionality and ability to trawl through public profiles and friends lists of one's own friends.

At the same time, despite very broad appeal, OSNs have encountered certain limitations to their proliferation. One reason is the fundamental connectivity requirement, i.e., the "O" in OSN: in order to use an OSN, one must be connected to the Internet and logged into the OSN provider. This is not surprising since most OSNs – including Twitter, Facebook, LinkedIn, VK and RenRen – are centralized.[1] In other words, there is no option for disconnected OSN usage. Consider the following scenarios:

- Low bandwidth or intermittent Internet connectivity, e.g., due to lossy and/or error-prone wireless links.
- Expensive Internet connectivity, e.g., abroad, on trains, planes and cruise ships.
- Complete lack of Internet access, e.g., in planes, under water, under ground or in remote locations.

We believe that such scenarios are fairly common for OSN users who travel. and they share a common feature in that OSN access is difficult: too slow, too expensive or simply impossible. However, there is no fundamental reason why two nearby OSN users – who either have no OSN access or do not want to connect to the OSN, could not have some *limited* OSN functionality. This observation is the premise and one of the motivating factors for this paper.

Our second motivating factor is privacy. In general, lack of privacy is not a fair complaint against OSNs. Most people join an OSN for social reasons and privacy is not their primary concern. Although privacy advocates often decry brazen collection, marketing, mining and selling of OSN-derived user information, expecting OSNs to behave in a privacy-friendly manner is unrealistic. On one hand, it seems reasonable to observe and retain behavior (i.e., actions) and locations of users connected to the OSN. On the other hand, if OSN users are communicating off-line, i.e., without involving the OSN infrastructure, it is no longer clear whether the OSN ought to have access to user behavior and location.

We note that there are two types of off-line user (inter-)actions: (1) those that lead to direct consequences to the OSN, and (2) those that do not. For example, consider two

[1]Although decentralized OSNs exist, e.g., Diaspora [9] and Safebook [5], they have not managed to attract many users.

WPES'15, October 12, 2015, Denver, Colorado, USA.

ACM 978-1-4503-3820-2/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2808138.2808149 .

OSN users: Alice and Bob, who interact *verbally and in-person* while being disconnected from the OSN. During the chat, they exchange information about their friends, work history and educational background. If they have nothing in common, they do not subsequently connect on the OSN. However, if they discover some common factors (e.g., some number of shared friends), they might decide to connect later. In the former case, the OSN clearly learns nothing about their encounter. In the latter, the OSN observes spontaneous establishment of their subsequent connection.

By analogy with the above example, suppose that Alice and Bob interact electronically while being off-line (with respect to the OSN) and their interaction leads to some impact on the OSN, e.g., they later connect or "friend" each other, thus changing their profiles. In this case, the OSN will rightfully learn about their prior off-line interaction. Otherwise, if Alice's and Bob's off-line activity does not lead to anything and there is no reason for the OSN to learn about their off-line interaction.

To summarize, this work is prompted by the need to support limited off-line interaction between nearby OSN users. We believe that supporting this type of interaction would be beneficial for OSN users, for two reasons: (1) they would engage in social networking in a wider range of settings, and (2) they would do so knowing that positive outcomes can lead to new OSN connections, while inconsequential activity remains private. Furthermore, off-line user interaction is advantageous to OSN providers, since it would extend the reach of social networking. At the same time, we recognize that not all privacy issues stem from the OSN itself. If users communicate directly with no OSN involvement, their mutual privacy is very important in cases that do not lead to a later OSN connection. We make this one of the main design goals.

Another key goal is information authenticity. When two OSN users interact on-line (via an OSN provider), information in their profiles can not be changed arbitrarily. For example, friends in Facebook or connections in LinkedIn are not added gratuitously. In the context of off-line interaction, we need to make sure that OSN profile information exchanged as part of that interaction is authentic and corresponds to the appropriate users.

In this paper, we design an architecture and a system, called *UnLinked*. The main idea is to combine users' social proximity with their physical proximity to privately discover common factors and later possibly establish OSN relationships. *UnLinked*supports private off-line discovery of nearby users with authentic common friends or connections, without direct user interaction. Although conceptually applicable to many current OSNs, *UnLinked* is grafted onto one specific OSN, *LinkedIn* aimed at professionals who use their profiles as a sort of an online CV.

1.1 Contributions

This work makes several technical contributions in addition to the overall *UnLinked* system design. As part of *UnLinked*, we come up with an efficient *Authorized Two-Way Private Set Intersection (ATW-PSI)* protocol and demonstrate its security. Current protocols are one-way, which means that only one party learns the intersection of the two input sets. Furthermore, they only certify one party's inputs. In our ATW-PSI, both participants learn the intersection only if each has a valid authorization issued by a *trusted third party (TTP)*. This has been identified as a major problem with prior techniques, e.g., [7]. Moreover, in contrast to prior work on protocols with linear complexity [8, 16, 20], participants in our protocol can not transfer authorizations for individual set elements to others. We also present a new approach to the well-studied *friend of friends (FoF)* discovery problem. By using *LinkedIn* as a concrete OSN platform and developing a fully functional prototype[2][3] of *UnLinked*, we show that ATW-PSI protocols are usable in realistic settings.

2. DESIGN GOALS

Before proceeding with the system design, we overview and motivate key desired features that are mainly derived from the above discussion:

- D0 **Off-Line Interaction**: support for efficient communication among two physically proximate OSN peers.We restrict the number of peers to two since the ultimate outcome of a fruitful off-line interaction is a two-way OSN connection.
- D1 **Peer Anonymity:** persistent user identifiers (names, user ids, email addresses) associated with OSN members must be kept confidential in off-line interaction, unless the two decide to connect later by jointly revealing their identities at the end of off-line interaction. We require anonymity despite user's apparent proximity, since, in many situations the peers may be physically close, yet still unaware of each other's precise location or other identifying information.
- D2 **Profile Privacy wrt Peers**: ability to perform certain OSN profile operations (e.g., compare respective sets of friends/connections) with mutual privacy. In other words, information learned from such operations must be limited to what is common to both peers.
- D3 **Interaction Privacy wrt OSN:** non-disclosure to the OSN of off-line peer interactions and their locations. This specifically applies to interactions that have no impact on OSN peers' profiles, i.e., no eventual connection establishment.
- D4 **OSN Connection Spillover:** ability to later establish actual OSN connections, based on prior off-line interaction that resulted in mutual agreement to connect. That is, it should be possible for two peers to connect via the OSN at some point when they are online, if they have decided to do so as a result of sufficient degree of commonality among their profiles, e.g., at least k shared friends.
- D5 **OSN Profile Authenticity and Owner Authentication:** authenticity (including timeliness) of OSN profile information as well as authentication of each peer as part of off-line interaction. This is needed to protect OSN members from impostors (non-members) as well as malicious members.
- D6 **OSN-Independent Operation:** ability to operate along-side (or on top of) an existing OSN, i.e., no requirement to introduce changes within an OSN; also, no restriction against an OSN implementing proposed functionality.

[2]The prototype Android app can be downloaded from https://db.tt/XQXE9pqF.

[3]Due to a recent change in the LinkedIn developer agreement, our latest prototype uses Twitter as an example OSN.

D7 **OSN-Agnostic Design:** minimal reliance on OSN-specific features, i.e., applicability to the broad spectrum of OSNs.

D8 **Voluntary Participation:** OSN users must *opt in* to participate in off-line interaction.

Note that off-line interaction between users of *different* OSNs, while desirable in the long term, is not among our goals for now.

3. SYSTEM DESIGN

Based on the goals outlined above, our system architecture (called *UnLinked*) includes two main software components:

1. *UnLinked* App (ULA): an application that runs on the OSN user's personal device, such as a smartphone or a laptop, that takes part in off-line interaction, and performs auxiliary tasks on-line. ULA primarily supports D0, D2, D3, described in Section 2 above.
2. *UnLinked* Server (ULS): a stand-alone server program that supports D1, D5, D6, D7 and D8.

There are several practical reasons for ULS to be stand-alone, as opposed to being a component of an OSN. First, it allows us to support desired off-line functionality without any involvement of – or permission from – any specific OSN provider (D6). Second, ULS can be expanded to support multiple OSNs (D7). Third, ULS can operate as a registration portal where OSN users can enroll to participate in off-line interaction (D8). Fourth, acting as a neutral and independent trusted third party (TTP), ULS can certify (authenticate) profile information of OSN users (D5). Last but not least, ULS serves as a sort of a *privacy buffer* between the OSNs and ULA users. Although it can be argued that, from the complexity perspective, it makes more sense to integrate ULS into the OSN itself, independent operation of ULS insulates the OSN from its natural lack of motivation to respect user privacy (D3). That said, most privacy guarantees still hold if ULS is integrated into the OSN.

3.1 OSN Requirements

Following D7, we need to minimize assumptions about the underlying OSN. The only requirement of *UnLinked* is that the OSN must offer a secure automated way to export member profiles in some well-defined format, i.e., something that can be parsed by ULS. This feature is supported, via REST APIs, by most major OSN providers, such as Twitter, Facebook, Google+ and *LinkedIn*.

3.2 Types of Communication in UnLinked

UnLinked involves several types of communication, shown in Fig. 1. To start, a user who chooses to use *UnLinked* needs to download and install ULA. Then, a user needs to interact with the OSN and facilitate secure export of her profile to ULS (Fig. 1b). This involves user-OSN and ULS-OSN communication. Finally, the main purpose of *UnLinked* is direct communication between off-line users, i.e., a pair of end-devices running ULA (Fig. 1c). If two users decide to later connect or become OSN friends, *UnLinked* helps facilitate this interaction, per Fig. 1d.

3.3 Communication Channels

UnLinked does not restrict the means of communication between the OSN and its users, or between ULS and OSN. We assume that both transpire over the Internet. Off-line interaction between users is assumed to involve a wireless broadcast communication medium that facilitates seamless discovery of peers, e.g., via periodic beaconing. We believe that this is a natural requirement for several reasons. First, per D0, two users need to be physically near each other. Second, virtually all modern personal devices (from laptops to smartphones) communicate over broadcast wireless channels.

Obvious candidates for off-line communication between OSN users are: WiFi, Bluetooth and NFC. Of these, NFC is less appealing than others since it requires the two devices to be very *near* each other. This might be "too close for comfort" in scenarios where users are separated by some distance – e.g., on planes, trains and ships – or whenever they prefer to maintain some physical space and personal privacy. Further, broadcast based user discovery would be impossible via NFC. Both Bluetooth and WiFi are available on a wide range of devices types.

3.4 Cryptographic (Privacy) Requirements

Based on design goal D2, we need to support private computation of common factors in OSN profiles of two users. These common factors could include: friends, friends-of-friends, educational and employment histories, as well as various group memberships. Furthermore, D5 requires authenticity of computed common factors.

While online, the OSN allows users to restrict profile information visible to others. Most OSNs only allow friends or friends of friends to view one's profile content. For example, *LinkedIn* requires that two users have some shared profile information before allowing one of them to request a connection. We aim for the same amount of peer privacy but in off-line operation where the OSN itself can not provide it. Thus, we clearly can not stipulate that users simply download and exchange ULS-certified copies of their OSN profiles. Instead, we need to run a set of cryptographic protocols that maintain a comparable (to on-line) level of privacy and allow users to learn only their common factors. This motivates the use of 2-party protocols that perform private set operations, such as Private Set Intersection (PSI) and Private Set Intersection Cardinality (PSI-CA). Furthermore, based on D5, these protocols must operate on user-bound and authenticated input (i.e., profile information), thus guaranteeing authenticity of the result.

4. LINKEDIN

Most popular OSNs offer users a personal *profile page* to describe themselves: provide information on their place of residence, educational and professional backgrounds, memberships, interests, as well as share photos, videos and other information. Generally, two users can connect to each other by becoming *contacts* or *friends*. Depending on the individual's privacy settings, one typically sees more information about friends, contacts or connections. (We use these three terms interchangeably.) Most OSNs allow users to hide profile information from those who are not direct contacts.

Why *LinkedIn*? Although, based on D7, the general design of *UnLinked* is OSN-agnostic, we needed to make an initial choice of a specific OSN platform. *LinkedIn*'s strong emphasis on validity and integrity of user's connections is important for *UnLinked*. Also, *LinkedIn* appeals mainly to adult professionals who generally tend to value privacy, serious communication and maintaining real world links more than the younger and/or more socially-minded population

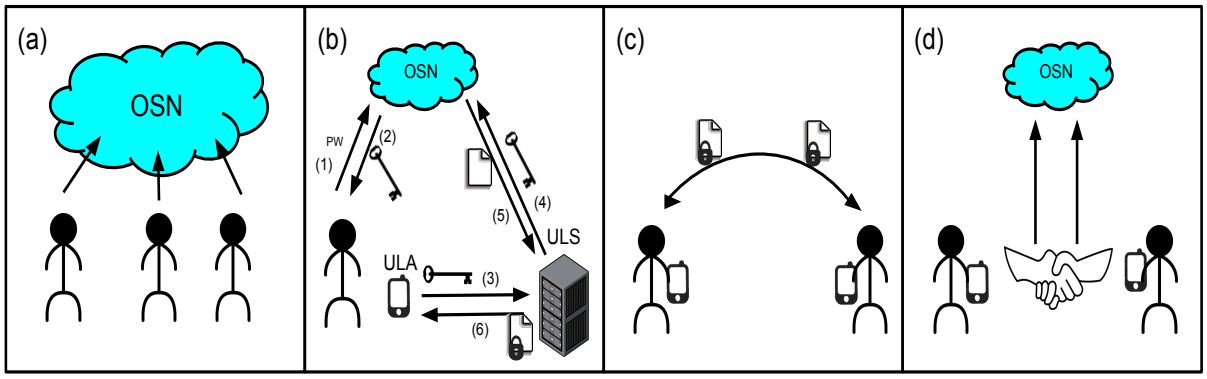

Figure 1: Interaction in *UnLinked*: (a) Regular OSN interaction, (b) Setup, (c) Offline, (d) OSN Spillover

of Tumblr or Facebook. Furthermore, considering D7 and Section 3.1, *LinkedIn* facilitates secure export of user profiles via OAuth and a REST API. Further background information on *LinkedIn* can be found in Appendix B.

5. CRYPTOGRAPHIC TOOLS

As discussed in Sect. 3.4, in order to support off-line interaction, we need privacy-preserving protocols that operate over generic and authenticated input. Furthermore, following design goal D5, a participant's input must accurately reflect its real OSN profile. To accomplish this, we rely on a trusted third party that verifies participant's input and issues a signed certificate of authenticity. ULS functions as this trusted third party: it validates user profile information by direct communication with OSN. We stress that ULS is only needed at setup time and it is not involved in any off-line interaction.

A *setup* phase is performed while A is on-line and connected to the OSN. In this phase, A inputs its profile information a and obtains the corresponding certificate $auth_A$ issued by ULS. This certificate is later used in off-line interactions with other OSN users, i.e., any B that receives $auth_A$ in the off-line phase, can validate a without learning its value.

Generally speaking, our goal in the off-line phase is for A and B to privately compute a function $P(a, b)$ via a two-party protocol P^* that realizes $P()$. To do so, we need to construct $auth_A$ such that a is bound to a specific evaluation of P^* on partial input a. That is, $auth_A$ bootstraps evaluation of $P(a, \cdot)$. Later, when B presents $auth_B$ to A, continued execution of P^* can only be used to compute $P(a, b)$. The basis for our constructions is that ULS signs the initial message of P^* which is later used as certified input to the off-line phase of the protocol. For this to work, P^* must have the following properties:

1. **Limited-Round:** ULS only signs the initial message sent by each party. All subsequent messages are open to manipulation. By minimizing the number of rounds, we also minimize potential impact of malicious participants who might deviate from the protocol.
2. **Mirrored:** All protocol actions are identical for A and B. This includes cryptographic operations and message transmission order. Thus, allows the same certificate can be used to both initiate and respond to P^*.
3. **Two-Way:** Both parties learn the same correct result $P(a, b)$. We note that this does not hold for two separate instantiations of a one-way protocol, since a malicious participant could provide different input in each one-way instance.

For the two certified private set operation protocols (ATW-PSI and ATW-PSI-CA) used in *UnLinked* we denote the first signed message as $BAS(\cdot)$, and $BAS^{CA}(\cdot)$, respectively. Description of our cryptographic building blocks and notation can be found in Appendix A.

5.1 Adversarial Model

Our initial goal is security in the so-called *Honest-but-Curious (HbC)* (aka semi-honest) model. In it, the *adversary* is a protocol participant who follows the protocol while passively trying to learn as much information as possible. However, as discussed in Sect. 5.3.4, we can achieve security in the stronger *Malicious* model with some simple extensions. In this model, the *adversary* can arbitrarily deviate from the protocol.

ULS is assumed to be trustworthy, i.e., it honestly and correctly certifies a user's input. We do not consider external adversaries, since standard network security techniques (e.g., TLS and IPSec) can mitigate outsider attacks. Finally, due to the use of hash functions, our constructs are secure in the Random Oracle Model [2].

5.2 Security Properties

We informally summarize desired security properties:

Correctness. We say that a protocol is *correct* if, whenever both A and B are honest, each outputs $P(a, b)$ at the end of protocol execution, except with negligible probability.

Peer-Privacy. We require secrecy of elements of a and b, unless they appear in the output of $P(a, b)$. For the case of P^* computing a private set intersection, secrecy holds only for elements not in $\{a \cap b\}$. In other words, no information beyond $P(a, b)$ is learned by either party.

Pseudonymity. By executing P^*, A and B must not learn each other's identities.

Authenticity. P^* aborts if a or b are not certified by ULS or if either $auth_A$ or $auth_B$ is expired. Thus, the adversary can not learn $P(a, b)$ for any a or b for which it does not hold $auth_A$ or $auth_B$.

Binding. To prevent transferability, an authorization provided to U on input $u = u_1, \ldots, u_n$, must be bound to all of u. That is, U can not transfer or omit any u_i-s.

Output Integrity. If a and b are certified by ULS, then P^* outputs $P(a, b)$. In the HbC model, output integrity is directly implied by authenticity and correctness.

Early Termination Resistance (ETR). If either party aborts the protocol, both must learn nearly[4] equal amounts of information. For an HbC adversary, termination may occur accidentally, e.g., A and B lose connectivity in the middle of the protocol or one of their devices dies. In the malicious model, no protocol can be guaranteed to be *Two-Way*. One party always learns the result first, at which point it can simply abort execution. *ETR* attempts to remedy this imbalance by limiting information conveyed by the final message.

5.3 Two-Way Private Set Intersection

Authorized Two-Way Private Set Intersection (ATW-PSI) is a protocol between users A and B on respective inputs: $(a = \{a_1, \ldots, a_m\}, auth_A, R_A)$ and $(b = \{b_1, \ldots, b_n\}, auth_B, R_B)$:

$$\mathcal{F}_{ATW-PSI}(a,b) \rightarrow \begin{cases} \perp & \text{iff } \neg VER_{PK}(BAS(a), auth_A) \\ & \text{or } \neg VER_{PK}(BAS(b), auth_B) \\ \{x_i | x_i \in a \cap b\} & \text{otherwise} \end{cases}$$

We claim that the combination of the *setup* and *offline execution* phases described below yields a concrete instantiation of the *ATW-PSI* protocol. This construction is based loosely on the One-Way PSI variant from [16].

5.3.1 Setup

Certification of user input is shown in Fig. 2. For user U, let $u = \{u_1, \ldots, u_n\}$ denote the set of U's inputs. ULS, on input of its private key SK, generates a random value $R_U \leftarrow \mathbb{Z}_p^*$ (step 1) and signs all hashes of elements in u, masked with R_U (step 2). Both $auth_U$ and R_u are transferred to U (step 3). Note that all exponentiations are *mod p*.

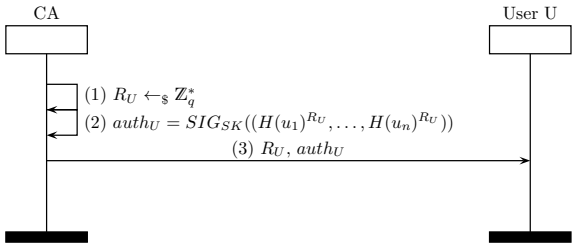

Figure 2: Setup

5.3.2 Offline Execution

User A on input $(a = \{a_1, \ldots, a_m\}, auth_A, R_A)$ and user B on input $(b = \{b_1, \ldots, b_n\}, auth_B, R_B)$ execute the protocol shown in Fig. 3. First, A computes a set $Z = \{z_1, \ldots, z_m\}$ where each $z_i = H(a_i)^{R_A}$. A forwards Z along with $auth_A$, to B. This allows B to verify whether A sent valid z_is by computing $VER_{PK}(z_1, \ldots, z_m, auth_A)$ in step (2). In concurrent steps (3) and (4), B computes Y (mirroring A's computation of Z) and forwards Y and $auth_B$ to A. Next, A exponentiates B's y_js with R_A and returns the results to B (step 5). B does the same with A's z_is and its own R_B (step 6). Note that protocol steps are executed concurrently when applicable. e.g., (1) and (3), (2) and (4). Also, in steps (5) and (6), each element sent by A is followed by one element sent by B. In step 7, both A and B output the intersection of a and b, i.e., a set of elements a_t (resp. b_s) where $y_s^{R_A} = z_t^{R_B}$ for $1 \leq s \leq m$ and $1 \leq t \leq n$.

[4]The meaning of *nearly* depends on the specific P^*. The less information revealed in each message, the closer the outputs will be.

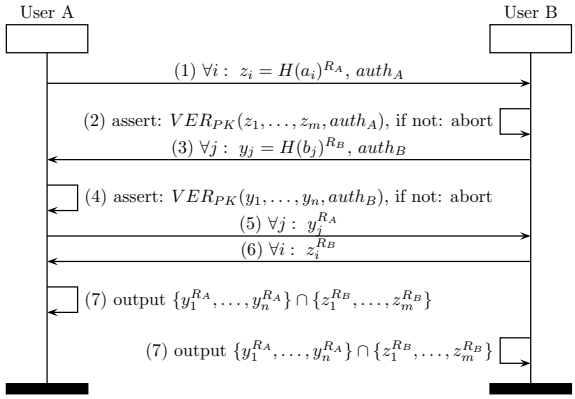

Figure 3: Offline Execution Phase

5.3.3 Security Considerations

We claim that ATW-PSI is a peer-private, pseudonymous instantiation of PSI, with binding in the presence of malicious adversaries in the Random Oracle Model [2]. Furthermore, ATW-PSI provides protocol execution integrity in the HbC model. We sketch the proof below. Since ATW-PSI is *Mirrored*, we assume w.l.o.g. that B plays the role of the adversary B^*.

Correctness follows trivially from the protocol description. If both parties are honest, both compute:
$\{H(a_1)^{R_A R_B}, \ldots, H(a_m)^{R_A R_B}\} \cap$
$\{H(b_1)^{R_A R_B}, \ldots, H(b_n)^{R_A R_B}\}$
This is equivalent to: $\{H(a_i)^{R_A R_B} | s.t. \ \forall i \ \exists j \ a_i == b_j\}$. Thus, both parties output the intersection $a \cap b$.

Peer-Privacy. For an HbC adversary, privacy is provided by the One-More Gap Diffie Hellman Assumption (OMG-DH) [11]. Informally, this provides indistinguishability of exchanged elements of the form $H(x)^R$, even with adversarial access to a Diffie-Hellman oracle. Only matching elements are learned in step (7). All privacy arguments (in the HbC model) from [16] apply to ATW-PSI with minor adjustments.

However, our protocols offer privacy in the *malicious* model with no further modifications. Since it can not deviate from the protocol without forging ULS's signature, B^* can only manipulate its response to A's initial message, in step 6. However, this message is completely de-coupled from any of A's responses. Thus, no matter what it sends, B^* learns no any additional information.

Pseudonymity. *Peer-Privacy* implies privacy of A's and B's identities during one protocol instance. However, further information can be learned from multiple protocol executions. As long as A re-uses the same $auth_A$, all off-line

interactions are initiated with this value, and B can easily correlate multiple encounters with A.

However, whenever it is on-line, A can always contact ULS and request a fresh certificate, i.e., a new pseudonym $auth'_A$, thus preventing B from linking future interactions with $auth'_A$ to any prior interactions involving $auth_A$. This is because any two protocol transcripts, both involving A on input a, with different blinding factors R_A and R'_A, are indistinguishable.

As discussed in Sect. 7.3, pseudonymity can be strengthened if a user obtains a batch of pseudonyms for each off-line epoch. In a single on-line interaction with ULS, A can request a batch of certificates and use each one as few or as many times as it wants. Clearly, if each $auth_A$ is only used once, pseudonymity transitions into anonymity.

Authenticity follows directly from unforgeability of the underlying signature scheme SIG. If either $auth_A$ or $auth_B$ does not verify in steps (2) or (4), execution is aborted.

Binding. Each $auth_U$ is bound to the entire set u. Therefore, binding is trivially achieved due to unforgeability of SIG.

Output Integrity. In HbC, this follows directly from security of SIG used by ULS and correctness of ATW-PSI. However, in the malicious model, A can convince an honest B that the protocol output is any subset of $a \cup b$.

5.3.4 Malicious Security

In its current form, ATW-PSI does not provide *Integrity of Output* in the malicious model. To achieve this stronger security, we need to modify steps (5) and (6) in Fig. 3 to force users to adhere to the protocol. This can be done using techniques similar to [16], i.e., by introducing two Schnorr-based signature of knowledge (SoK) constructs: SK-LOG and SK-EQ-LOG [1], described in Appendix A.

First, we modify *setup* to include g^{R_a} within $auth_A$. During *Offline Execution*, A provides a single (Schnorr) signature SK-LOG proving knowledge of R_a. Later, in step (5), A provides a proof of correctness that shows, via SK-EQ-LOG, that each exponentiation ${y_j}^{R_a}$ is performed with the same R_a supplied earlier in g^{R_a}. These measures demonstrate that: (1) A indeed uses the same R_a in all exponentiations of y_js, and (2) A uses the R_a as signed by ULS and included in $auth_A$. For B's part, Step (6) is modified similarly.

Early Termination Resistance (ETR). Even with the Integrity feature, a malicious participant can abort the protocol at any time. We counter this by enforcing concurrent execution of steps (5) and (6). Specifically, messages between A and B are "interleaved", i.e., each party alternates between transmitting and receiving a specific set element ${z_i}^{R_b}$ or ${y_j}^{R_A}$. The two will only swap once the computation has been verified, using SoK. This way, at any point during protocol execution, parties compute a equal portion of the intersection. A malicious participant learns at most one additional element; thus, its advantage is very low.

Clearly, ETR does not come for free. The modified protocol incurs $O(max(n, m))$ rounds, instead of the previous two. Although in some scenarios this added complexity might make the protocol impractical, recall that *UnLinked* users are expected to be near each other and transmission latency is therefore expected to be negligible.

6. SYSTEM ARCHITECTURE

UnLinked includes two phases: Setup and Off-line. This section describes basic system assumptions and the details of each phase.

6.1 Requirements

Although *UnLinked* is OSN-agnostic, there are a few requirements:

- In order to support meaningful off-line interactions, OSN profile information must contain at least a unique owner's user id and a list of connection's ids. Also, OSN must provide the ability to securely export a user profile once authorized by the user. This is offered – via OAuth2 API [13] – by several popular OSNs, e.g., *LinkedIn*, Facebook, Twitter, and Google+. In our setting, an OAuth2 interface allows ULA and ULS to communicate to OSN on behalf of the user.
- ULS requires a public key signature scheme $[SIG_{SK}(\cdot), VER_{PK}(\cdot, auth)]$ with a public/private key-pair: $[PK, SK]$. ULS retains no state information about its interactions with any ULA or OSN.
- ULA needs access to a broadcast medium, in order to support nearby peer discovery. ULA is also responsible for storing all user's private cryptographic information, including BAS_U. In addition, ULA must trust ULS's PK; this is provided by including PK in the ULA installation package.

6.2 Setup Phase Details

The goal of this phase is for ULA to obtain enough information to be able to later operate independently from OSN and ULS. Since a typical OSN user's profile can change fairly often, e.g., on a daily basis, we need to ensure that users have up-to-date profile information. For this reason, ULS issues BAS_U with a relatively short lifespan, e.g., one week in the current implementation, as described in Sect. 7.4. Setup proceeds as follows:

1. Via local web browser, Alice logs in to OSN, retrieves and stores an OAuth token *tok*. Alice's password **is not revealed** to ULA.
2. Alice's ULA presents *tok* to OSN, retrieves her profile P and stores it locally.
3. Alice's ULA presents *tok* to ULS.
4. ULS relays *tok* to OSN
5. OSN sends Alice's profile P.
6. ULS parses P into basic components:
   $[c = \text{connections}, \text{identity}, \text{misc}]$
   where identity consists of the user-name or identifier as well as a profile picture, if any. The misc field is partitioned into OSN-specific sub-fields, i.e., education, employment and residence.
7. ULS computes and sends to ULA: $\sigma_{id} = SIG_{SK}(\text{identity})$, along with $\sigma_1 = SIG_{SK}(BAS(c))$, $\sigma_2 = SIG_{SK}(BAS(\text{misc}))$ and the corresponding random blinding factors Ru_1, Ru_2[5].

6.3 Off-line Phase Details

This phase handles communication between peer ULAs. Once connected, ULAs privately compute common information of their profiles. Depending on user policy, this may

[5]If requested by ULA, ULS uses $BAS^{CA}(\cdot)$ (Blinded Attribute Set for ATW-PSI-CA) instead of $BAS(\cdot)$

entail multiple rounds of communication and executions of both ATW-PSI (defined in Sect. 5.3) and ATW-PSI-CA (defined in Appendix C). We now describe the interaction between two nearby ULAs: Alice and Bob, assuming that the former starts the interaction.

1. Without involvement of the actual human users (i.e., in the background) Alice and Bob emit broadcast "pings" or beacons at fixed time intervals[6], advertising their membership in *UnLinked*.
2. Alice detects Bob's broadcast and responds with a PID. PID selection is discussed in Section 7.2 below.
3. If Bob wishes to continue communication, it responds with its own PID.
4. Using standard techniques (TLS, in our case), ULAs establish a secure channel.
5. ULAs execute a series of ATW-PSI-CA protocols, privately computing the size of their common connections and miscellaneous fields. Comparing connections means exchanging σ_1, $BAS(c)$, followed by a series of short computation and transmission rounds. See Sect. 5.3, and Appendix C for more details.
6. Next, ULAs perform one or all of the following, depending on their mutual desire to continue interaction, either by explicit user action, or by policy. Their choices are conveyed across the secure channel.
   (a) ULAs perform a series of ATW-PSI protocols to compute the actual set intersection of their profiles, analogous to computation and communication in Step 5. Each instance of ATW-PSI is performed over a different type of data, e.g., connections, employers and educational institutions.
   (b) ULAs exchange authenticated identities of the form (identity, σ_{id}). Upon receipt, identity is validated and results are displayed to the user.
   (c) ULAs exchange messages via a simple chat interface.
   (d) If both decide to later connect through OSN, each ULA stores the peer's identity. At a later time, when it is on-line, ULAcan use the OSN to request a connection to the peer.

6.4 Notification Policy

As mentioned earlier, during the off-line phase, ULAs perform a series of cryptographic tests, each on one of various profile fields. A ULA determines explicitly which tests are conducted and, for every test, based on its results, whether it wants to continue interacting with a particular peer. However, such fine-grained control over every interaction can be cumbersome. We envision numerous scenarios where large numbers of OSN users, all with something in common, are in physical proximity. For instance, on a university campus, most users will have the same employer (or educational institution) and several connections in common. In such settings, most users would rather not to be notified of every nearby peer. In other settings (e.g., on a plane, or simply far from home) a user may wish to be notified of any nearby peer even with very little in common between their profiles.

To increase flexibility, we introduce the notion of personal *policy*. Each policy includes a set of conditions for ULA to initiate each of the following actions:

1. Execute a more revealing protocol. i.e., ATW-PSI instead of ATW-PSI-CA.
2. Alert the user to the presence of a peer.
3. Exchange identifying information, e.g., name, image, or OSN profile id.
4. Send a message to a peer.
5. Receive and display messages and requests to exchange ids.

Criteria for these actions can be any combination of the following:

1. A threshold or a range based the size of the intersection of connections or any $misc$ sub-field.
2. Presence or absence of a specific value in the intersection.

Note that criteria of the first kind can be applied using only the result of ATW-PSI-CA. Whereas, the second type of criteria relies on successful ATW-PSI execution for that field.

While our policy language for *UnLinked* is flexible, ULA is pre-configured with a few default policies. All such policies require explicit user actions to exchange identifying information and send messages. Furthermore, criteria for notifying the user of a message or identity exchange are identical to that for ULA to be alerted to the presence of a nearby peer. This simplifies effects of policy to the user.

The most permissive supported policy, *Open*, allows ULA to run ATW-PSI with any *UnLinked* peer. This policy is useful for maximizing off-line interactions. A slightly more restrictive policy, *Low*, is designed for use away from one's typical locations, e.g., on travel. In this case, the user is alerted of a nearby peer's presence if any of the following conditions are satisfied:

1. At least one friend in common, and one school or employer.
2. At least three friends in common.
3. A common current employer or current academic institution.

We also provide a third default policy, *Medium*, designed for use at everyday locations, such as work-place or school. In this case, having an employer or a school in common is not very meaningful. Instead, we add extra weight to the value of past employers and connections. Users are notified if any of the following occurs:

1. At least three connections in common, and at least one common employer or academic institution, other than the current one.
2. At least five connections and one institution in common.
3. More than seven connections in common.

Preventing SPAM: User policy also controls the amount of communication with peers. If Alice and Bob have different policies, Bob might wish to contact Alice but not vice versa. Suppose that Alice is not alerted of Bob's presence. Should Bob's contact attempts be transmitted to Alice? This decision is based upon Alice's policy. If desired, Alice can specify a different threshold for receiving messages from a peer than that for being initially alerted to the peer's presence. Without this feature, Bob could bombard Alice's ULA with spam messages or connection requests simply by acquiring many fake OSN identities.

7. DISCUSSION AND EXTENSIONS

We now discuss some system considerations and extensions.

[6]Typical interval between pings is 30-60 secs.

7.1 Minimizing Irrelevant Connections

In some settings, ULA may encounter the same peer multiple times. Being notified of each such encounter is burdensome to the user and costly in terms of device resource consumption. To this end, ULA has a mechanism for preventing repeated interactions. However, a given user's profile information might change frequently and to ensure new information is considered in future encounters this process is time-dependent. To this end, the prevention mechanism operates on two levels. First, ULA maintains a list of recently encountered device MAC addresses. Entries in this list have a lifespan of one day. We consider this to be an acceptable granularity, since user profiles rarely change dramatically within a single day. Second, each ULA broadcasts its own public identifier PID. Upon connecting to a new device, ULAs exchange $PIDs$. If a PID has been seen recently, communication is terminated. When an off-line interaction completes, the peer device address is added to the *avoid list.*[7]

In addition to avoiding recently seen peers, we consider a likely scenario where two users who are already each other's connections discover each other anew and attempt to connect. Clearly such unnecessary interactions should be avoided. For this purpose, we provide a simple mechanism that allows users to learn, and to be optionally alerted, when current connections are nearby. As part of every interaction with ULA, ULS inserts a "dummy" tuple corresponding to that ULA into the set of connections returned to ULA. Upon interacting with a peer, this tuple will appear in the result set if and only if the two are already connected.

7.2 Authenticated Channels

There are many ways to select $PIDs$. To easily establish an authentic channel we take advantage of the trusted ULS. The basic idea is that each PID is the public portion of a Diffie-Hellman key exchange. Specifically, for a user U, $PID = g_u^R$. To fully prevent man in the middle attacks, ULS includes this PID in all authorizations returned to U. This effectively binds all certificates to the corresponding user.

7.3 Unlinkability

For a privacy conscious user, *UnLinked* can also operate in an *unlinkable* mode. In it, ULA contacts ULS and receives a new authorization along with a new private key R_u, whenever possible. This provides the user with a unique pseudonym linked to the specific R_u and allows the user to control the degree of unlinkability. Of course, other factors influence both privacy and performance. To be completely unlinkable, a user might wish to avoid redundant connection optimizations discussed previously. Using the mechanism for detecting pre-existing connections would significantly lower the level of anonymity to their connections, and could trivially reveal their identity in some cases. Furthermore for each pseudonym, ULA needs to reset the list of previously discovered peers. Otherwise unwillingness to participate in an interaction may itself be used to link pseudonyms. Finally, ULA may need to make device configuration changes dependent on the medium, e.g., change MAC addresses.

Nonetheless, even with these safeguards, peers always learn the outcome and input size of cryptographic protocols. This information could be used to help identify the other user. Exact severity of this de-anonymization is dependent on the uniqueness of one's contact graph and common profile traits.

[7] $PIDs$ are refreshed when new profile information is downloaded.

7.4 Freshness of Credentials

In order to operate offline, *UnLinked* relies on authenticated profile information. Since profiles can change with arbitrary frequency, ULAs should be assured of freshness of peer profiles. Otherwise, misbehaving users with stale or outdated information could claim connections they no longer have. Intuitive ways of dealing with this problem rely on either time-stamped signatures or revocation lists. Since *UnLinked* operates off-line and aims to minimize overhead, distributing periodic revocation information is not feasible. Instead, all ULS signatures include a global timestamp and a relatively short validity interval of one week. We believe that this is sufficiently long for ULA to regain connectivity to ULS. At the same time, it is short enough to minimize the impact of malicious peers. To support this feature, we assume that each ULA's clock is loosely synchronized with that of ULS. This is a reasonable assumption for most modern devices, even when operating off-line.

7.5 Detecting Misbehavior

Since ATW-PSI and ATW-PSI-CA do not offer output integrity from a malicious peer by default, *UnLinked* provides an alternative to handle malicious behavior. This is achieved by allowing ULS to audit communication traces from offline interactions. A ULA records every protocol transcript. Later, when it re-gains connectivity, ULA sends a transcript to ULS along with its current profile. ULS audits every message in the transcript and determines whether peers behaved correctly. ULS can then blacklist misbehaving users or take other measures to prevent further cheating. However, auditing interaction transcripts imposes considerable storage and computation burden on ULS and exposes ULA's off-line interactions to ULS. Consequently, auditing is an optional feature, disabled by default. Only a user who suspects fraud should submit transcripts for auditing.

8. IMPLEMENTATION & EVALUATION

We developed and benchmarked a fully functioning prototype of *UnLinked*, consisting of an Android application implementing the functionality of the ULA and a Ruby on Rails application implementing ULS.

For the signature scheme ULS computes "attached" RSA signatures (those that include the signed data in the signature) using PKCS7 implemented over OpenSSL. Currently, ULA supports a wide range of communication technologies: WiFi Direct, Bluetooth, and WLAN based Network Service Discovery. Any subset of these can be used concurrently. The current prototype only supports ATW-PSI without extensions for malicious security. For the purposes of benchmarking, ULA was tested on two Nexus 5 smartphones communicating over Bluetooth. Each phone has a 2.26GHz processor and 2GB of RAM. ULS was tested on a desktop running Ubuntu 12.10, with an Intel i7-3770 3.4GHz quad-core CPU and 16GB of RAM.

Performance: *UnLinked* was tested on several input sizes: 10, 100, 1000, and 10,000 friends. As expected, protocol execution time was linear in the number of friends. For 100 friends, ULS signature computation completed in 0.83s, and used 25.2kb of storage. Also, each ULA completed the Of-

fline Phase of the ATW-PSI protocol in under 1 second, with 526kb exchanged. See Tab. 1 for detailed results. We note that, even with 1,000 friends, information returned by ULS in the setup phase only consumed 243kb. One could easily batch 100 of such signatures to interact with unlinkability off line, without significant storage impact. Similarly, authorizations for ATW-PSI-CA can be batched without significant storage costs. Finally, round-trip times between devices was, on average, only 7.86ms. When using ATW-PSI with interleaving this amounts to less than 1 second of overhead for the input size of 100 friends.

Table 1: Evaluation of *UnLinked*.

Input Size	Setup Time (s)	Signature Size (kb)	Offline Time (ms)	Offline Band. (kb)
10	0.0739	3.41	143	9.2
100	0.837	25.2	508	75
1000	8.57	243	8790	2941
10000	86.4	2420	14530	7337

9. RELATED WORK

Most related work falls into two groups: (1) private discovery of common friends and (2) cryptographic protocols for private set operations.

9.1 Private Friends Discovery

In von Arb et al. [24] privacy is considered in the *friend-of-friends* scenario where two users want to learn whether they have the same contacts on their mobile phones, based on phone numbers. The proposed technique, based on [15], uses commutative encryption: each party encrypts its own elements and gets the resulting ciphertexts encrypted by the other party. By comparing encrypted ciphertexts both parties identify common elements. This approach offers no authorization of set elements. Thus, parties can use any phone numbers as protocol input and learn the other party's contacts. Moreover, there are no ETR features, meaning that a malicious party can end the protocol as soon as it learns the intersection.

De Cristofaro et al. [7] introduced the concept of *private contact discovery* that offers privacy-preserving computation of common contacts. The proposed scheme uses *index-based message encoding* [19]. Unauthorized relaying of contact set elements is addressed via contact certification which requires no trusted third party. The main idea is that each user U issues to its every contact V a contact certificate, which is essentially U's signature on V's id. Thus a certificate is bound by the issuer to a specific user (contact) and can not be relayed. As [24], this scheme is prone to early protocol termination attacks. Also, it is limited to connections and does not consider other types of profile information, e.g., educational insitutions or past employers, for which such contact certificates are not applicable.

Nagy et al. [20] construct a *Common Friends* protocol that allows two parties to privately learn whether they are already friends or share some OSN friends. The protocol uses so-called *bearer capabilities* [23] for authenticity of friends lists. Bearer capabilities constitute proofs of friendship. Other than containing IDs of OSN users, those capabilities are considered to be "high-entropy objects". Consequently [20] claims that using full-blown (standard) PSI protocols is overkill since set elements are not "predictable". This allows the usage of more efficient PSI protocols based on Bloom filters [3]. Concerning authenticity, [20] acknowledges that *re-distribution* of contacts is not addressed.

9.2 Private Set Intersection

Private set intersection (PSI) protocols [8], [16], [4], [10], [14] allow two parties, each with its own set, to privately compute a set intersection. In other words, if two parties' (private) sets include common elements, one or both learn(s) these elements and no information about other set elements (other than their number) is revealed. If both parties learn the result, the protocol is called *mutual* or two-way PSI. It is claimed in [8] that mutual PSI can be obtained by two instantiations of *one-way* PSI. However, we believe that this holds only in the semi-honest model, where protocol executions require no binding. In an *authorized* PSI protocol, set elements need to be signed beforehand by a trusted third party. In this setting, special care must be taken to disallow trading authorized inputs. This is typically not addressed.

9.3 Policy-Enhanced Private Set Intersection

Another class of related protocols is called *Policy-Enhanced Private Set Intersection* (PE-PSI) [22]. These allow a ATW-PSI to be constructed from any PSI protocol secure against a malicious adversary. The scheme from [22] offers super-linear computational complexity. Also, binding is not provided: any individually authorized set element can be omitted from the intersection. Furthermore, this scheme requires a PSI protocol secure in the malicious model, which may not be two-way. Thus, in some instantiations, output integrity is not provided. In a practical deployment, such as *UnLinked*, the added costs of such schemes may be prohibitive. In contrast, ATW-PSI and ATW-PSI-CA provide efficient binding, as well as ETR in the malicious model, and can be optionally extended to provide output integrity.

10. CONCLUSIONS

In this paper, we reported on the design of *UnLinked* that supports private off-line interaction among nearby OSN users. As part of this work, we developed a novel ATW-PSI protocol that allows two parties to learn the intersection of their pre-authorized private input sets. Pre-authorization of these input sets by ULS prevents transfer and manipulation of individual set elements. A fully functional prototype of *UnLinked* is available for Android smartphones, allowing LinkedIn users to automatically discover nearby peers who share a sufficient number of friends or other profile features.

A far as future work, we intend to integrate information from multiple OSNs, e.g. *Facebook*, *Google+* and *Twitter*. This will allow for more flexible policies and would allow us to tap into a greater user pool. Furthermore, we plan to investigate whether our approach is applicable to other application scenarios, such as mobile social services relying on encounter-based trust.

11. REFERENCES

[1] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO 2000*, pages 255–270. Springer, 2000.

[2] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73. ACM, 1993.

[3] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, July 1970.

[4] J. Camenisch and G. Zaverucha. Private intersection of certified sets. In R. Dingledine and P. Golle, editors, *Financial Cryptography and Data Security*, volume 5628 of *Lecture Notes in Computer Science*, pages 108–127. Springer Berlin Heidelberg, 2009.

[5] L. A. Cutillo, R. Molva, and T. Strufe. Safebook : a privacy preserving online social network leveraging on real-life trust. *"IEEE Communications Magazine", Vol 47, Nq12*, 12 2009.

[6] E. De Cristofaro, P. Gasti, and G. Tsudik. Fast and private computation of cardinality of set intersection and union. In *Cryptology and Network Security*, pages 218–231. Springer, 2012.

[7] E. De Cristofaro, M. Manulis, and B. Poettering. Private discovery of common social contacts. *International Journal of Information Security*, 12(1):49–65, 2013.

[8] E. De Cristofaro and G. Tsudik. Practical private set intersection protocols with linear computational and bandwidth complexity. *IACR Cryptology ePrint Archive*, 2009:491, 2009.

[9] Diaspora Foundation. Webpage. `https://diasporafoundation.org/`.

[10] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19. Springer Berlin Heidelberg, 2004.

[11] D. Freeman. Pairing-based identification schemes. *arXiv preprint cs/0509056*, 2005.

[12] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[13] D. Hardt. The OAuth 2.0 authorization framework, Oct. 2012.

[14] C. Hazay and Y. Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In R. Canetti, editor, *Theory of Cryptography*, volume 4948 of *Lecture Notes in Computer Science*, pages 155–175. Springer Berlin Heidelberg, 2008.

[15] B. A. Huberman, M. Franklin, and T. Hogg. Enhancing privacy and trust in electronic communities. In *In Proc. of the 1st ACM Conference on Electronic Commerce*, pages 78–86. ACM Press, 1999.

[16] S. Jarecki and X. Liu. Fast secure computation of set intersection. In J. Garay and R. Prisco, editors, *Security and Cryptography for Networks*, volume 6280 of *Lecture Notes in Computer Science*, pages 418–435. Springer Berlin Heidelberg, 2010.

[17] M. Lepinski and S. Kent. Additional diffie-hellman groups for use with IETF standards, January 2008. RFC 5114.

[18] LinkedIn Help Center. Account Restricted. Webpage, Mar. 2013. `https://help.linkedin.com/app/answers/detail/a_id/1386`.

[19] M. Manulis, B. Pinkas, and B. Poettering. Privacy-preserving group discovery with linear complexity. In *Proceedings of the 8th international conference on Applied cryptography and network security*, ACNS'10, pages 420–437, Berlin, Heidelberg, 2010. Springer-Verlag.

[20] M. Nagy, E. De Cristofaro, A. Dmitrienko, N. Asokan, and A.-R. Sadeghi. Do i know you?: Efficient and privacy-preserving common friend-finder protocols and applications. In *Proceedings of the 29th Annual Computer Security Applications Conference*, ACSAC '13, pages 159–168, New York, NY, USA, 2013. ACM.

[21] C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 4(3):161–174, 1991.

[22] E. Stefanov, E. Shi, and D. Song. Policy-enhanced private set intersection: Sharing information while enforcing privacy policies. In *Public Key Cryptography–PKC 2012*, pages 413–430. Springer, 2012.

[23] A. S. Tanenbaum, S. J. Mullender, and R. van Renesse. Using sparse capabilities in a distributed operating system. In *International Conference on Distributed Computing Systems (ICDCS)*, pages 558–563, 1986.

[24] M. von Arb, M. Bader, M. Kuhn, and R. Wattenhofer. VENETA: Serverless friend-of-friend detection in mobile social networking. In *IEEE Conference on Wireless & Mobile Computing, Networking & Communication*, 2008.

APPENDIX

A. CRYPTOGRAPHY BACKGROUND

Prime Order Groups

Our schemes require a prime order subgroup with a generator g, along with a full domain cryptographic hash-function $H(\cdot)$ with a range of this subgroup. Specifically, we use a Diffie-Hellman subgroup defined by primes p and q where $p = tq + 1$, for some integer t [17]. As an instantiation of $H(\cdot)$ we use the SHA-1 hash function, whereby, on input x, $H(x) = (\text{SHA-1}(x))^t \ mod\ p$.

Public Key Signature Scheme

We assume the existence of a public key signature scheme secure against existential forgery under an adaptive chosen message attack [12]. Given a public/private key-pair (PK, SK) we denote such a scheme as a pair of protocols: sign – $SIG_{SK}(\cdot)$, and verify – $VER_{PK}(\cdot,\cdot)$. These protocols operate on message m: $\sigma = SIG_{SK}(m)$ and $VER_{PK}(m, \sigma)$. Verify returns true if and only if σ was computed using knowledge of SK.

Signatures Of Knowledge

We use two popular signature-of-knowledge (SoK) schemes. They operate similar to zero knowledge proofs in that they

allow a party to demonstrate knowledge of secret without revealing any additional information (given some shared public knowledge). However, SoK can be performed without interaction, via Schnorr signatures [21]. Specifically, we rely on signatures demonstrate knowledge of (1) a discrete logarithm and, (2) two discrete logarithms (with different bases) being equal. These signatures operate over the same cyclic subgroup as discussed above.

To show knowledge of discrete log of g^x, a party computes SK-LOG as:

$$SKLOG(g^x, g, m) = (c, s) = (h(g^x||g||g^t||m), t - cx)$$

where t is chosen randomly. Upon receipt of $(c, s, y = g^x)$, one can verify that $c = h(y||g||g^s y^c||m)$.

Similarly we can construct an SoK proving equality of two discrete logs with different bases SK-EQ-LOG, i.e., given (g^x, f^y), show that $x = y$. This is done by constructing:

$$SK\text{-}EQ\text{-}LOG(y = g^x, z = f^x, g, f, m) =$$
$$(c, s) = (h(g^x||f^x||g||f||g^t||f^t||m), t - cx)$$

Verification is perfomed as: $c = h(y||z||g||f||g^s y^c||f^s z^c||m)$.

For more details we refer to [1].

B. LINKEDIN

LinkedIn, with more than 250 million users worldwide, is a global OSN that provides a networking platform for professionals. As such, it is widely considered to be more serious or "grown-up" than its more social counterparts, such as *Facebook.* To this end, a typical *LinkedIn* profile page resembles a CV or a resumè, rather than a dynamic scrapbook.

Connections play a major role on LinkedIn. One user gets connected to another by sending an invitation. Such a "direct connection" between two users is established only if the invitee accepts the invitation. The invitee can refuse it and report the inviter as unknown, or the invitation – as spam. An inviter who accumulates many invitation refusals might get her account restricted [18]. This approach constitutes an entry barrier for fraudsters who attempt to join groups of professionals and provides some level of trusted relationships among users. Moreover, users can benefit from their contact network by getting introduced to new contacts, by their existing contacts. This way, a user has "access" to *second-degree* and even *third-degree* connections.

The main idea is that this "get introduced" approach might lead a user not just to new contacts but possibly to new companies or jobs, as recommended by their connections. Thus, a major goal for *LinkedIn* users is to discover and establish new connections based on criteria such as: common connections, interests, membership and educational or professinal experience.

C. TWO-WAY PSI CARDINALITY

As the privacy and usability of *UnLinked* can be greatly enhanced by additional crypto primatives, we present here an extension of PSI-CA form [6] to construct a linear-time Two-Way Private-Set Intersection Cardinality protocol with Authenticity (ATW-PSI-CA). We follow a similar procedure to ATW-PSI with minor modifications to *setup* and *offline* phases. Resulting protocols are shown in Fig. 4 and Fig. 5 respectively. They rely on two random permutations Π, and Π'.

ATW-PSI-CA provides the same security as ATW-PSI, however, at additional cost. Unlike ATW-PSI, authorizations ($auth_U$) cannot be reused. Instead, U must obtain from CA a distinct $auth_U$ for each future off-line interaction. In Sect. 8 below, we argue that, in the context of *UnLinked*, this is a small price to pay for better privacy.

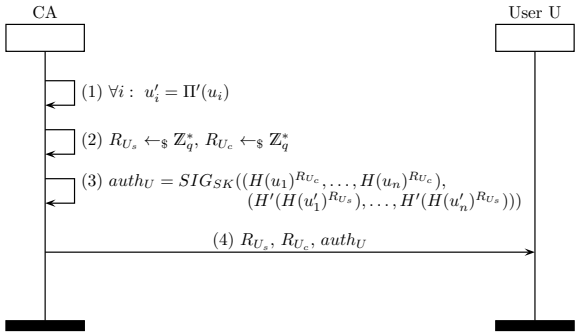

Figure 4: Setup w/ Cardinality

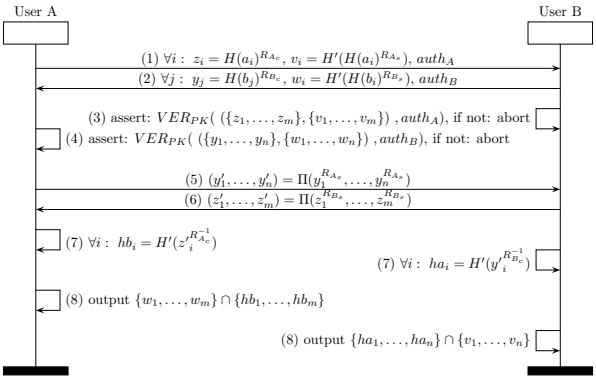

Figure 5: Offline Execution Phase w/ Cardinality