

Consistent Deniable Lying: Privacy in Mobile Social Networks

Sebastian Kay Belle¹ and Marcel Waldvogel¹

Distributed Systems Laboratory¹
University of Konstanz
Konstanz, Germany
<first>.<last>@uni-konstanz.de

Abstract. Social networking is moving to mobile phones. This not only means continuous access, but also allows to link virtual and physical neighbourhood in novel ways. To make such systems useful, personal data such as lists of friends and interests need to be shared with more and frequently unknown people, posing a risk to your privacy. In this paper, we present our approach to social networking, Consistent Deniable Lying (CDL). Using easy-to-understand mechanisms and tuned to this environment, it enables you to meet new friends with joint interests while limiting exposure of your private data. Not only can it be generalised to include “friends of friends” (transitivity) into interest search, it allows you to plausibly refute any allegations of your claimed interests. Unlike prior work, we focus on the application to similarity finding and include the novel aspects of transitivity and deniability, which are key to success in social networks.

1 Introduction

Mobile phones combine the benefits of networked computers and personal assistants: They can gather information from the Internet while also surveying your surroundings, not require any activity of yours until something to raise your attention has been identified. When used for building social networks, both this metaphor and the real mobile phone require access to large amounts of information. Many people willingly publish massive amounts of data into social networking platforms, unawares of the risks of identity theft, the possibility of this data becoming embarrassing to you in a few years, or other abuses of your data. However, private data can also be used in a useful manner in many cases. For instance, publishing personal data could facilitate locating like-minded people.

Thinking about mobile devices, this idea can be extended in such a way that data could be published on a mobile device such that other devices can connect to and search for some desired information, or even that a mobile device could broadcast the contained information to other devices in the vicinity through wireless P2P connections. Especially nowadays, where virtual social networks grow and gain severe impact on how people interact with each other, the extension of social networks to mobile devices discloses interesting new aspects of interaction. Thus, the demand to exchange data while preserving privacy is an interesting goal to accomplish. We envision not only an

increase in privacy-awareness, but also in the use of location-dependent mobile contact services, such as *Nokia Sensor*¹.

Before we put up the requirements, let us discuss two sample uses:

1. The *Terminator*, as a techno-savvy man from the future, is, of course, a long-term passionate user of our system. One of his goals is to find other rough boys to help him fulfil his cruel appointments. Unbeknownst even to the scriptwriter, the Terminator is one of the most sensitive men in the university. Deep down in his heart, he would like to watch romantic movies holding hands with an empathic young woman. Looking out for a fellow moviegoer without becoming the laughing-stock of his tough friends requires that he can plausibly deny his inner self.
2. *Dr. Leonard "Bones" McCoy* is visiting a medical congress of rare diseases. Many of his fellow doctors bring their patients' records, hoping to be able to run their illness theories on a larger dataset. Dr. McCoy and his colleagues are aware of the recent results of Narayanan and Shmatikov [1], knowingly that simple "anonymisation" can easily be undone, causing the doctors to lose their license due to violation of the privacy act. Here, it would be helpful if the real information was slightly perturbed to avoid reverse engineering, while still keeping many of the properties for analysis. Fortunately, the data does not need to be entirely accurate: Once a theory crystallises, it can be manually verified by the individual dataset-contributing doctors on the original data.

Examining the requirements closer, we find that we have a privacy-preserving, symmetric, transitive, approximate set-intersection problem with deniability. This is explained as follows and addressed by our approach, Consistent Deniable Lying (CDL).

Privacy-preserving: Neither party should learn too much about the other or be able to perform data mining too easily.

Symmetric: Both parties should gain similar knowledge about each other.

Approximate: There is no need for exact results, as long as persons with shared interests can be identified with high probability. Even the occasional error can be a success, as a discussion starting with "oh, you really believe that I like *this*?" can be a good start.

Transitive: When looking for a person with particular properties, you frequently ask your friends whether they know someone matching your description. Transitivity enables such queries to be answered by the system, even if the friend(s) you are asking do not share the desired properties.

Deniable: If you feel that its none of their business, you can always plausibly declare that some of the properties in your public profile are wrong.

To achieve these properties, our system under design extends your profile with additional, fake, interests. Human interests are generally clustered, so fake interests cannot be purely randomly chosen but need to model the clustering relationships from realistic interests. Today, we are in a unique position that such data and their relationship is publicly available, which finally enables this novel approach. For example, cddb.com or

¹ <http://www.nokia.ch/A4335350>

	PIR	Access Control	Inference Control	PIC	Set Interactions	Encr. Bloom Filter	CDL	
Consistency	✓	✓	✓	✓	✓	(✓)	✓	Consistency
Deniability	X	X	X	X	(✓)	(✓)	✓	Deniability
Symmetry	X	X	X	✓	✓	✓	✓	Symmetry
Transitivity	X	X	X	X	(✓)	✓	✓	Transitivity
3rd Party	X	X	(X)	◆	X	◆	X	3rd Party
Global DB	X	X	X	X	X	X	◆	Global DB

Key: ◆ Required -- ✓ Available -- X Not Available -- (•) Limited

Fig. 1: Comparison of Algorithms

imdb.com are large repositories of music and movie relationships (genre, artists, topics, ...); whereas wikipedia.org contains relationship information for encyclopaedic knowledge.

Unlike prior work, we focus on the application to similarity finding and include the novel aspects of transitivity and deniability, which are key to success in social networks.

2 Background and Related Work

Preserving privacy in information retrieval is an ongoing field of study with several distinct and promising techniques, all intended to reduce the ability of others to excessively mine data (cf. Fig. 1).

Traditionally, privacy in databases is ascertained by *access control*, completely preventing access to selected types of information. When some of the information should be made available, *inference control* (cf. [2]) can be used to ensure only aggregated information is delivered, whereas operations on multiple subsets will not reveal individual entries. The inverse is *private information retrieval* (cf. [3], [4]), where the database is unable to successfully profile the querier. Bloom filters [5] already provide some form of uncertainty: Not only are there false positives, but it is hard to identify the data originally put into the filter, unless the candidate set of members can be enumerated in practice. As shown in e.g. [6], the latter is frequently the case today, so hashing techniques ranging from Bloom filters to cryptographically strong hashes provide no privacy in many of today’s applications due to finite set sizes.

Bellovin and Cheswick [6] use encrypted bloom filters to query a data base. In a nutshell, Bellovin and Cheswick mask the intentional database query by augmenting the proper questions with fake inquiries. As the authors show, making the fake questions look plausible is very hard, when your questions are under the scrutiny of a trained eye. Their approach differs from CDL in that they modify the query, not the database and requires a semi-trusted third party.

Freedman et al. [7] provide efficient private set intersection, which solves the above problems, but lacks transitivity and, more importantly, deniability: An attacker can create a set of his “interests” which cover what it wants to learn from the victim, thereby being able to profile him with arbitrary scrutiny.

Woodruff and Staddon [8] introduce the concept of private inference control, PIC, to control the amount of information that can be obtained by a querier or the database owner.

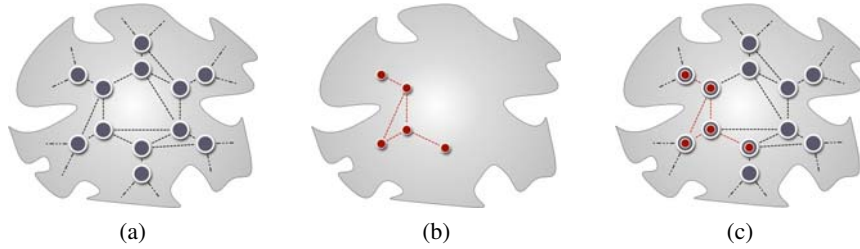


Fig. 2: Global information space mapped onto G (a) and the user-specific subset of the information space mapped onto G' , respectively (b). Thus, G' is a subgraph of G (c).

3 Consistent Deniable Lying

As Bellovin and Cheswick [6] stated – and what is a matter of common knowledge – *telling a consistent set of lies is hard*. We propose a simple method to address the requirements listed in section 1, given that a global non-user-specific dataset is on-hand, which is a superset for user-specific datasets. E.g. the cddb.org or freedb.org is a global superset of a user’s iTunes library.

Formally, let D_G be the global, non-user-specific database and let $D_U \subset D_G$ be a user’s specific database (e.g., a user’s iTunes library). Access to D_U should be restricted as processing a query presumably violates the user’s privacy. Assumed that D_G exists, we map D_G onto a graph $G(V, E)$. Each vertex $v \in V$ represents a single data element of D_G , each edge $(v, u) \in E$; $v \neq u$ relates to an arbitrary but predefined relationship between v and u (e.g., written by the same artist). Simplified, we map the information space contained in D_G onto a graph (cf. Fig 2(a)). Likewise, we create a graph $G'(V', E')$ for the user-specific database D_U , utilising the same relationship between two vertices $v', u' \in V'$ as was used to relate the vertices of G (cf. Fig. 2(b)). Thus, G' is a subgraph of G (cf. Fig. 2(c)). Note that G and G' are not necessarily fully connected.

Beside mapping the information space onto a graph, we utilise bloom filters to encode such a graph. Let B denote a bit array of length m and let $H_j(v)$; $j = 1 \dots n$, $n < m$ be n different hash functions that map a vertex $v \in V$ to an index $i \in \{0, \dots, m-1\}$ of B . Thus, to query a user’s database and check if it contains information $u \in V$ we simply check the bits in B for $H_j(u)$. As a user-specific database is mapped to $G'(V', E')$ with $V' \subset V$ it is easy to create a query utilising elements from V and compare it to the bit vector that encodes V' (cf. Fig. 3(a)). This approach is especially suited for mobile devices due to the low computational costs needed to compare two bit arrays.

For instance, *Dr. Leonard “Bones” McCoy* utilises his smartphone to create a query for some strange symptoms he encountered lately during the yearly check-up of the *Terminator*. The query is encoded into a bit array and broadcasted to his colleagues mobile devices at the conference.

However, as this simple scheme would allow set interactions to retrieve sensitive information from V' we propose to introduce fake bits in B that are generated by mutation of G' . Simplified, we add fake information by adding additional vertices to V' from $V \setminus V'$ (cf. section 3.2). Thus, encoding the modified set of vertices V' in B we generate a bit vector containing untruths (cf. Fig. 3(b)).

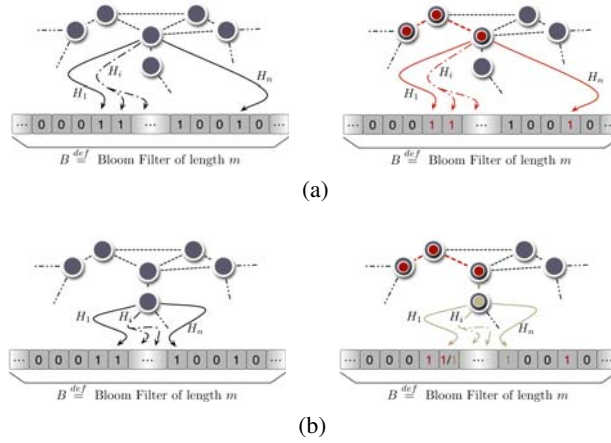


Fig. 3: Relationship between the bloom filter for G and subgraph G' (a) and relationship between the bloom filter for G and G' with introduced fake vertices (beige) (b).

Bloom filters are listed here as one likely use, thanks to their compactness and their slight gain in privacy. Nevertheless, CDL even provides necessary advantages when the set is encoded in plain-text or when private set intersection methods are used.

3.1 How to be Consistent

As we propose to introduce untruths into a user-specific database D_U we need to verify that the false information included is consistent with the true dataset of the user. To achieve consistency we make use of the precondition that $D_G \supset D_U$. By mapping D_G and D_U to G and G' , respectively, we can formalise this precondition as follows::

$$\bigcup G' \subseteq G \quad \text{thus} \quad \forall G'; G' \subset G \quad (1)$$

So, as we extract fake information not related to the user out of $G \setminus G'$ the introduced untruths are consistent with the user's true dataset.

3.2 How to Lie

Following, we explain three simple schemes to add fake data to G' . Introducing fake elements creates an extended graph G'_{lie} that contains additional, false information. Formally, we create $G'_{lie}(V'_{lie}, E'_{lie})$ by extending G' so that $G' \subset G'_{lie} \subset G$. All three schemes can easily be combined to create G'_{lie} in which the magnitude of dummy elements inserted control the reliability of G'_{lie} whenever a query is processed. Initially, let $G'_{lie} = G'$ and let $G_{lie}(V_{lie}, E_{lie})$ be the graph of all possible untruths where $V_{lie} = \{V \setminus V'_{lie}\}$ and $E_{lie} = \{E \setminus E'_{lie}\}$. We create G'_{lie} successively through an arbitrary combination of the three fake vertex insertion methods. While G'_{lie} grows, G_{lie} thins out, thus, we can define the following invariant::

$$(G' \cup G'_{lie}) \cap G_{lie} = \emptyset \quad (2)$$

Random Insertion:: The simplest method to introduce untruths is to add random vertices as illustrated in Fig. 4 (a). We calculate the number of randomly selected vertices as::

$$n_{rand} \stackrel{def}{=} \lfloor \delta_{rand} \cdot |V_{lie}| \rfloor; \quad \delta_{rand} = [0, 1] \quad (3)$$

where δ_{rand} is a user chosen threshold. Then we select and remove n_{rand} vertices $v \in V_{lie}$ and add them to V'_{lie} . Further, if we select a vertex $v \in V_{lie}$ that is adjacent to one or more vertex $u \in V'_{lie} \cup V'$, we also add the edge that defines the relation between u and v to E'_{lie} and remove the edge from E_{lie} . Note, while it is simple to select random vertices, the set of selected vertices still forms a consistent set of lies as discussed in section 3.1.

Growth Insertion:: Second, we introduce lies by growing G'_{lie} through adding neighbouring vertices. We add false vertices adjacent to vertices $v \in V'_{lie}$ as depicted in Fig. 4 (b). Let $C(V_C, E_C)$ be a subgraph of G'_{lie} , thus, $V_C \subset V'_{lie}$ and $E_C \subset E'_{lie}$ and let T be the spanning tree for C . T can easily be extended by adding vertices beneath the leaves of T . Let $\delta_{depth}, \delta_{ext} = [0, 1]$ be two user-defined thresholds, $L_p = \emptyset$ the set of *processed* leaves of T , and $L_u = \{l | l \text{ is leaf of } T\}$ the set of *unprocessed* leaves of T . We add fake vertices as follows::

1. Calculate random variable $\chi_{ext} = [0, 1]$
2. If $\chi_{ext} \leq \delta_{ext}$ and $L_u \neq \emptyset$ proceed, else **break**
3. Randomly select and remove leaf $l \in L_u$ and add it to L_p .
4. Set::

$$\delta_{ext} \leftarrow \delta_{ext} - \chi_{ext} / (|L_u| + 1)$$

5. Calculate random variable $\chi_{depth} = [0, 1]$
6. If $\chi_{depth} \leq \delta_{depth}$ proceed, else go back to step 2.)
7. Randomly select and remove vertex $v \in V_{lie}$ where $dist(v, l) = 1$ in G ,² add it to V'_{lie} and L_u , and add $(v, l) \in E$ to E'_{lie}
8. Set::

$$\delta_{depth} \leftarrow \delta_{depth} - \chi_{depth}$$

and go back to step 5.)

Cluster Insertion:: Third we propose to generate fake clusters as illustrated in Fig. 4 (c). Again, let $C(V_C, E_C)$ be a subgraph of G'_{lie} . Further, let $\delta_{dvsd} = [0, 1]$ be a user-defined threshold, $n_{cnt} = |V_C|$, and $L = \emptyset$ an initially empty set of vertices. Then we run the following steps to create a fake cluster $C'(V'_C, E'_C)$::

1. Select random vertex $v \in V_{lie}$ with $\exists u \in V_{lie}; dist(v, u) = 1$, and add it to V'_C .
2. If $\exists u \in V_{lie} \setminus V'_C$ with $dist(u, v) = 1$; $(u, v) \in E_{lie}$, add u to L , add u to V'_C , and add $(u, v) \in E_{lie}$ to E'_C . Else, go to step 6.)
3. Calculate random variable $\chi_{dvsd} = [0, 1]$. According to χ_{dvsd} set::

$$\begin{aligned} v &\leftarrow v \text{ if } \chi_{dvsd} \geq \delta_{dvsd} \\ v &\leftarrow u \text{ if } \chi_{dvsd} < \delta_{dvsd}, \text{ and remove } u \text{ from } L \end{aligned}$$

² Note that $(v, l) \notin E_{lie}$ as $E'_{lie} \cap E_{lie} = \emptyset$.

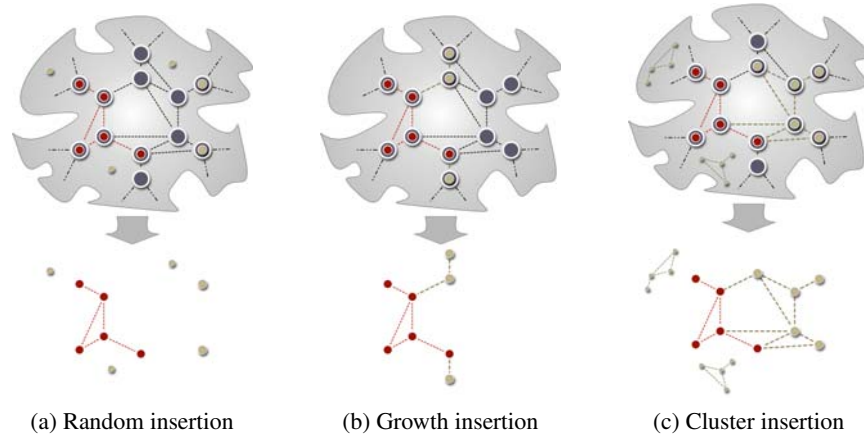


Fig. 4: The three fake vertex insertion techniques. Fake vertices are shown in beige.

4. Select a second discrete random variable $\chi_{cnt} \in 0, \dots, n_{cnt}$ and set $n_{cnt} \leftarrow n_{cnt} - \chi_{cnt}$
5. If $n_{cnt} > 0$ go back to step 2.), else discard L and go to step 7.)
6. Randomly select and remove $u \in L$. Set $v \leftarrow u$ and go back to step 2.)
7. $\forall u, v; u \in V'_{lie}, v \in V'_C$ and $dist(u, v) = 1$ add $(u, v) \in E$ to E'_{lie}

Thus, we are able to create clusters $C' \subset G'_{lie}$ to enhance G'_{lie} .

3.3 How to be Deniable

The deniability of our system implicitly emanates from the consistent set of untruths as well as the methods we utilise to choose them. By inserting fake information selected from a global dataset (cf. section 3.1, eq. (1)) all information related to a user is equally probable, thus, there are no obvious “outliers” that can easily be identified as meaningless information in the given context. Second, by introducing three different techniques that add fake information in a non-deterministic but plausible manner (cf. section 3.2), false information cannot be refined. In any case, a user can argue that information in his dataset could be introduced through random addition of false elements, thus, the user himself is the deciding factor to let someone know if the received information is true or false.

3.4 How Close are You?

As contacts grow more reliable the amount of false information inserted into G'_{lie} can be reduced on a *per-user basis* by defining access rules that reduce the thresholds that control the amount of false information in a user’s dataset. Furthermore, by introducing access rules for indirect contacts, that are (trusted) direct contacts of a user’s 1st degree contacts, we facilitate transitivity to socialise or share sensitive information.

To come back to our initial examples, suppose that our tech-savvy *Terminator*, the covert romanticist, got in (real-world) contact with *Trixie* through the CDL system. As

the *Terminator* gained more confidence in *Trixie* over time, he decides to define an access rule for *Trixies*' 1st degree contacts that enables them to gather more precise information about the *Terminator* such that he potentially extends his private social network further. Note that other users that query the *Terminator*'s profile still access the original bit vector.

Likewise, *Dr. Leonard "Bones" McCoy* could facilitate the use of indirect access rules regarding his problem with the strange symptoms he encountered during the *Terminators*' check-up. One of "*Bones*" long lasting friends who can not attend the conference "*Bones*" is visiting defined several access rules for some of his direct contacts that also attend the conference. Thus, as "*Bones*" is now enabled to gather more precise information from some indirect contacts he luckily finds a colleague that posted some similar symptoms of another patient. Getting into direct (real-world) contact with this colleague at the conference, this colleague, a psychiatrist, comes up with a plausible diagnosis: suppressed emotions!

4 Conclusion and Future Work

We proposed a technique that enables us to achieve synchronous, privacy preserving information exchange provided that a global dataset is available. By introducing consistent untruths in a user's specific dataset the user gets the deciding factor to authorise the information other users may see. Further, we defined a method to refine the untruths introduced in a dataset on a *per-user basis*, thus, giving the user control over the amount of fake information and also taking into account transitive relationships.

In the future we will investigate integration of other private set interaction algorithms as well as fuzzy geographic location using mobile devices. Further other methods to introduce lies, like deleting vertices could be an interesting field to research.

References

1. Narayanan, A., Shmatikov, V.: How to break anonymity of the netflix prize dataset (2007)
2. Farkas, C., Jajodia, S.: The inference problem: a survey. *SIGKDD Explor. Newsl.* **4**(2) (2002) 6–11
3. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. *J. ACM* **45**(6) (1998) 965–981
4. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: single database, computationally-private information retrieval. In: *FOCS '97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS '97)*, Washington, DC, USA, IEEE Computer Society (1997) 364
5. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* **13**(7) (1970) 422–426
6. Bellovin, S.M., Cheswick, W.R.: Privacy-enhanced searches using encrypted bloom filters. Draft (2004)
7. Freedman, M., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: *Advances in Cryptology — EUROCRYPT 2004*. (2004)
8. Woodruff, D.P., Staddon, J.: Private inference control. In: *Proceedings of the 11th ACM Conference on Computer and Communications Security*, Washington DC, USA (2004) 188–197