2012 IACSIT Coimbatore Conferences
IPCSIT vol. 28 (2012) © (2012) IACSIT Press, Singapore

K-Anonymity for Real-time Social Network Applications with Network Based anonymization and processing framework

Bharathi M[1], S. N. Chandrashekara[1], Anil G.N2[2], Anand R[2], Muneshwara M.S[2]

[1] Department of CSE, SJCIT, Chikaballapur-562101

[2] Department of CSE, BMS Institute of Technology, Avalahalli, Bangalore-560064

Abstract. The K-anonymity provides privacy guarantees over publicly released data sets with specified quasi-identifiers. However, the most common public releases of personal data are now done through social networks and their APIs, which do not fit the previous research-centric data set release model, nor do they allow for clear assumptions about quasi-identifiers. In this paper we proposes (i) A definition of K-anonymity that suggests a practical way in which social networks could provide privacy guarantees to users of their API.(ii) K-anonymity-based framework for privacy, termed as network-based anonymization and processing (NAP).To support as wide a range of applications as possible, the proposed privacy guarantee assumes all social-networking data may be a quasi-identifier and does not assume that data may be generalized and still be useful. Using the Facebook social networking API, we implement an application to demonstrate that providing such guarantees in real-time is feasible for real social networking data.

Keywords. Anonymizer, LocationServer, Candidate Set, Anonymity Degree.

1. Introduction

Every month at least 250 million Facebook users release their personal social networking data through Facebook's API. The recipients of this data include over 500,000 social networking applications, 80,000 external websites, 65 million mobile devices, and 180 mobile operators [1]. Due to the nature of how this data is used and accessed, the traditional definition of K-anonymity and how it is implemented does not apply to this public release of personal data. This paper suggests an alternative approach to K-anonymity that directly applies to this data and how it is accessed.

This paper introduces a new type of privacy guarantee for online social network (OSN) applications. This privacy guarantee deals with the "re-identification" problem and K-anonymity definition as discussed in [2]. However, this paper suggests a new K-anonymity problem to fit the most common usages of personal social networking data. The traditional approach to K-anonymity is to provide a guarantee over a data set such that quasi-identifiers associated with one identity in the data set are indistinguishable from (copies of) at least k-1 other identities in the data set. While interesting as related to data sets for research purposes, the traditional K-anonymity definition is not so useful when applied to the release of social networking data through publicly available social networking APIs for the following reasons:

1) Social networks do not currently anonymize their data sets and have released no plans to do so in the future. Furthermore, the social networks are the only entities capable of such anonymization.

2) It is possible that any or all social network attributes may be used as quasi-identifiers, therefore all social network data (profile data) must be considered as quasi- identifiers.

3) Social network API calls access or refer to only a small subset of the overall data set and usually only refer to a particular subset of attributes related to an individual. Whereas, traditional K-anonymity guarantees require anonymity across the entire data set.

The primary differences between this definition of K- anonymity and the traditional definition given in [3] are listed below:

All data is considered a "quasi-identifier". This paper takes a "guilty until proven innocent" approach to whether or not data could be used to re-identify an individual. The privacy guarantee describes how a subset of data relates to the entire data set. There is no privacy guarantee on the entire data set and the privacy guarantee only applies to the data specified in a particular release. If this privacy guarantee were to be extended over multiple releases it would require that all data cumulatively released be evaluated together. When applied to the entire data set the proposed K-anonymity definition is equivalent to the traditional one.

The privacy guarantee refers to a number of sets of individuals rather than a number of individuals. Since an API call may specify a set of users and and not just a single user, we suggest expressing the privacy guarantee in terms of a minimum number of sets and not just individuals.

2. K-anonymity Problem

This paper proposes a personal social networking API, that is, an API which provides privacy guarantees for the data that it releases. In particular, it guarantees than an "anonymous" release of personal data cannot be used to distinguish between a minimum number of users, designated by k. The value of k refers specifically to the number of distinct and indistinguishable sets of users. This paper chooses to define k in terms of sets of users rather than total number of users due to the latter being implied by the former but not vice versa.

This new definition of K-anonymity was motivated by the fact that the existing definition relates to the public release of data sets, usually for research purposes. However, this definition makes assumptions that are not realistic or useful when applied to the how social networks release their personal data. Social networks release their data through public APIs, through which social network applications, external web sites, or mobile applications can integrate social networking data.

2.1. Definition 1. K-Anonymity

Given two sets U and T. U is a set of all individuals and T is a set of all quasi- identifiers. There is a many-to-many mapping between T and U. t is any subset of T and u is a subset of U.

Tk is the superset of all sets tx which may be released under a K-anonymity guarantee, then$\forall$tx Є Tk, $\exists${u1, . . . , uk} Є Uk where: $\forall$ux ЄUk, ux $\rightarrow$ tx and |Uk|= k.

3. Cloaking Algorithm

In this section, we present the cloaking algorithm of our NAP framework. Our primary objective is to guarantee reciprocity- based anonymity. In NAP, the AZ anonymizes u with a set of line segments/edges instead of a spatial region (ASR).

The crux of our cloaking method is to utilize a global edge ordering, i.e., an ordered sequence that contains all network edges exactly once. The edge ordering is setting-sensitive, i.e., it specifies which end-node of the edge precedes the other. We refer to the position and setting of an edge in the ordering as the edge order and the edge setting, respectively. To avoid confusion, the setting of an edge depends solely on the ordered sequence. The number next to each edge indicates its order and the arrow its setting.

The edge ordering defines an implicit linear order among the users themselves. In particular, a user u precedes another u1 if the edge of u has smaller order than that of u1. If they fall on the same edge ni nj (with setting from ni to nj), u precedes u1 if it is closer to ni. Ties among coinciding users are resolved arbitrarily. This precedence relationship defines the order ordu of each user u. The position of a user in the defined sequence is referred to as the user order. The example in Fig. 1a contains 10 users whose subscript indicates their order (i.e., user u3 has order 3, etc.).Reciprocity in NAP is achieved by conceptually partitioning the user ordering into buckets of K users each, and forwarding to the LS the edges corresponding to the bucket of the querying user u. This set of edges is called the Anonymizing Edge List (AEL) of u. Specifically, let U be the set of users registered with the AZ and assume that a querying user requires anonymity of degree K. Set U is partitioned into B= [|U|/K] buckets, each containing K users, except the last one which may contain up to $2\cdot K - 1$; the ith bucket bi (for i < B) consists of users with order from $(i- 1) \cdot K +1$ to $i\cdot K$, and the Bth is assigned the remaining ones.

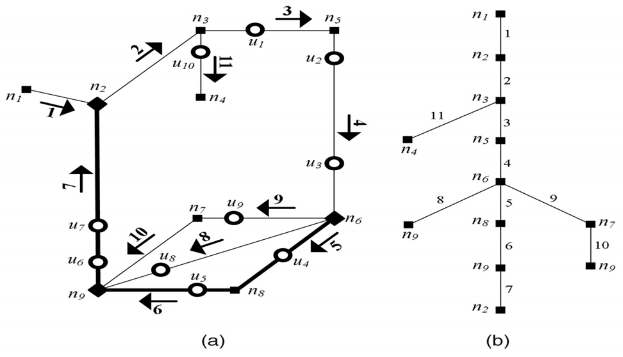


Fig. 1. Edge ordering. (a) Users. (b) DF search tree.

Consider the network in Fig. 1a, where |U| = 10, and assume that u6 poses a query with anonymity requirement K = 3. This results into three buckets b1 = {u1,u2,u3}, b2={u4,u5,u6}, and b3={u7,u8, u9,u10}. User u6 belongs to b2 and is anonymized together with the other users in it. The boundary users (i.e., first and last) of b2 are u4 and u6, whose edges have orders 5 and 7. The AEL is formed by collecting all edges with orders between 5 and 7, i.e., it comprises edges n6n8, n8n9, and n9n2, shown bold.

The AEL for any user in the same bucket is identical, and therefore, an adversary cannot pinpoint the query originator with a probability higher than 1=K (recall that each bucket contains K or more users). Hence, our cloaking method satisfies reciprocity. Reciprocity, in turn, is a sufficient condition for anonymity and thus, NAP guarantees K-anonymity to the querying users.

4. Social-K architecture

This section will outline the architecture of an example personal social network API, Social-K. Social-K was implemented on top of the Facebook API to test feasibility of such a solution.

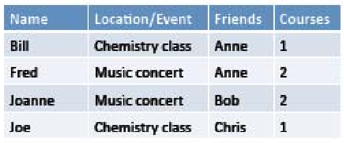
4.1. Identity Server

The Identity Server (IS) provides identity management services for social network application users. The IS generates and distributes anonymous identifiers (AIDs) to users through requests. An AID is a nonce which may be included with a query to anonymously identify the user (or users) associated with the query. A user's AID is mapped to the user's social network ID by the database on the IS.

4.2. Social Network Data Gatherer

After the IS receives a request for a user's social network profile data (we will call this user "user A") Social-K begins gathering pertinent user profile data for a large set of users. It was convenient to start with the user's friends on the social network since they were easiest to access through the Facebook API.

C. Expression Builder



Name	Location/Event	Friends	Courses
Bill	Chemistry class	Anne	1
Fred	Music concert	Anne	2
Joanne	Music concert	Bob	2
Joe	Chemistry class	Chris	1

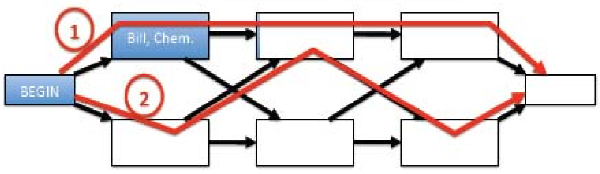


Figure 2. Example of how released data set fChemistry Class, Anne, 1g and its associated individuals can be expressed as a directed graph in which all paths are sets of individuals which could possibly account for the released data set

Consider the example shown in Figure 2. If the data set (Chemistry class, Anne, 1) is released, it could be mapped back to the distinct sets (Bill) and (Fred; Joe) implying that at least, Bill OR (Fred AND Joe) could have generated the data. This would be an example of K-anonymity where K <= 2.

For more complex data sets, the Boolean expression linking the set of users to the set of data to be released needs to be systematically derived, which can be achieved using a directed graph that models the relationship between the data to be released and the individuals who could be linked to that data. For any data set that we wish to release (d1, d2, ..., dn), we construct a column of nodes for each di, where each node in column i consists of the pair (username, di). This identifies all possible users who could be associated with the release of data item di. Next, we interconnect the nodes in a column with the nodes in the next column. This creates the directed graph. The set of all truth cases would be the superset of all paths across the graph. Each path would map to a conjunctive clause of literals (one literal per node) in the final disjunctive normal form (DNF) Boolean expression.

4.3. Logic Minimizer

Next, we apply logic minimization algorithms to simplify the Boolean expression provided by the Expression Builder. Several well known logic minimization algorithms exist, including ESPRESSO [4] and Quine-McCluskey [3]. When applied to our example graph in figure 2, the simplified Boolean expression reduces to (Bill) OR (Joe, Fred). If the number of disjunct clauses is greater than or equal to k then the data is admissible under a K-anonymity guarantee, if not the data must be filtered by the Profile Data Filter.

4.4. Profile Data Filter

If a set of data is found to not meet the required K- anonymity guarantee then the data is reduced and tested again. This project has not yet designed any advanced technique beyond random withholding and retest, however any mature solution to this problem should consider this step seriously as it involves a trade-off between response time, computational resources, and information loss.

5. Feasibility Study

We have implemented a prototype of Social-K and performed an initial evaluation of its behavior and feasibility. This section describes our implementation and some initial results from a feasibility study.

5.1. Implementation

The IS was implemented using the Java Standard Edition (SE) 5.0 platform. All IS services accessed by mobile and/or stationary devices are exposed as web services conforming to the REST architecture [5]. We expose each resource on the IS, including a user's AID and the Facebook profile information for a user, as separate URL-accessible resources supporting the HTTP GET method. The body of each HTTP request is encoded using JSON (RFC 4627). All web service network traffic between the IS and other mobile/stationary devices is encrypted using HTTPS, and access to all resources is authenticated using HTTP basic access authentication (RFC 2617). [6] Provides more information about the implementation of the IS.

We use an open source Quine-McCluskey implementation [7] to perform logic minimization in Social-K. As our evaluation results will show, this component performs reasonably well as the number of variables and terms in the Boolean expression to be minimized increases.

We use the SocialAwareFlicks application described in [8] as an example of a context-aware social networking application that queries the IS for user profile data. SocialAwareFlicks displays movie trailers that match the movie preferences of one or more users jointly watching a common large-screen display.

5.2. Feasibility Study Results

We have gathered some initial performance metrics to demonstrate the feasibility of Social-K. All performance testing was performed using a Mac book notebook running Mac OS X 10.5, with a 2.0 GHz Core2Duo processor, 2 GB of RAM, and a university-provided high-speed Internet.

Our metrics were gathered using the Facebook account of a volunteer, here called "user A", who has 222 Facebook friends and seven favorite movies listed on his Facebook profile. In a study conducted by Ellison et al. [9], the mean number of Facebook friends reported by the study participants was between 150 and 200. Therefore, we suppose that user A is a reasonable representation of a typical or average Facebook user.

We conducted our evaluation of Social-K performance by submitting a query to the IS requesting the list of favorite movies in the Facebook profile for user A. Social-K begins processing this query by first gathering the favorite movies lists for each of user A's Facebook friends. Social-K then proceeds to use the Expression Builder component to construct a Boolean expression representing the relationship between user A's friends and their favorite movies.

Figure 3 shows how the time required to minimize the unsimplified Boolean expression in the Social-K Logic Minimizer component varies with the number of terms in the unsimplified Boolean expression. We see from this plot that the Logic Minimizer component scales reasonably well for unsimplified Boolean expressions containing up to about 450 terms. We expect that 450-term Boolean expressions will account for many typical usage scenarios, although this will vary based on the number of the user's favorite movies that match with his friends' favorite movies. There is a nonlinear relationship between the number of terms in the unsimplified Boolean expression and the number of terms in the simplified expression. Based on the results of our tests, we have found that unsimplified Boolean expressions containing around 450 terms have up to about 20 terms when simplified by the Social-K logic minimizer. 20 terms in the simplified Boolean expression provides K-anonymity guarantees for k = 20. Thus, we have shown that Social-K is feasible for K-anonymity guarantees up to k = 20, which includes user groups as large as most social network friend lists (consisting of 200–300 friends).

Figure 4 shows how the time required to minimize the unsimplified Boolean expression in the Social-K Logic Minimizer component varies with the number of terms in the simplified Boolean expression (k). We see from this plot that there is minimal correlation between the value of k and the time required to minimize the unsimplified expression. We can conclude from figures 3 and 4 that the time to minimize the unsimplified Boolean expression is correlated with the size of the input to the Social-K Logic Minimizer component number of terms in the simplified Boolean expression (k) (the number of terms in the unsimplified expression), and is not correlated with the size of the output (the number of terms in the simplified expression). This is the expected behavior for a logic minimizer.

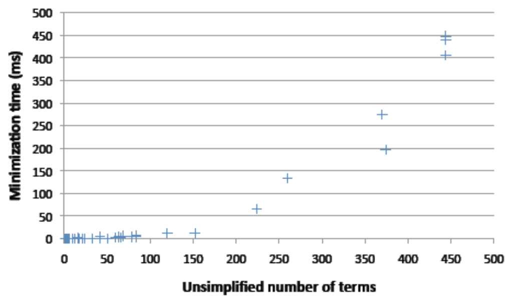


Figure 3. Time to minimize the unsimplified Boolean expression vs.number of terms in the unsimplified Boolean expression

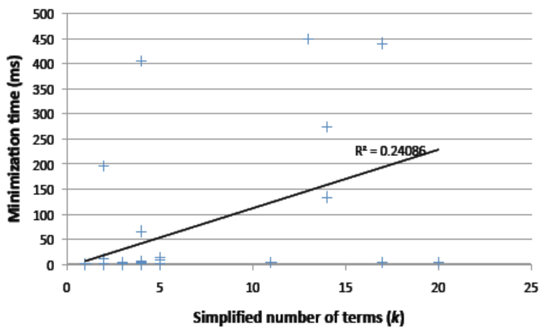


Figure 4. Time to minimize the un simplified Boolean expression vs.

Table I shows the run times of each component of Social- K for the following conditions for user A:

Number of friends: 222

Number of movie matches: 13

Number of friend matches: 7

Number of terms in unsimplified expression: 339

Number of terms in simplified expression: 7

The Social-K total run time in table I is the time for our system to return K-anonymous favorite movie preferences for a user. In our tests, the run time of the Social Network Data Gatherer component dominates the run time of Social-K, since this component spends most of its time downloading data from Facebook. We expect that running Social-K with local access to Facebook's user database would significantly reduce the run time of this component. However, the current average total Social-K run time of 1377 ms for our tests provides acceptable performance for applications such as SocialAwareFlicks.

Table I RUN TIMES FOR EACH SOCIAL-K COMPONENT

Component	Mean run time (ms)	Run time standard error (ms)
Social network data gatherer	852	145
Expression builder	11	1.7
Logic minimizer	297	8.29
Social-K total	1377	134.7

6. Conclusion

This paper presents Social-K, a new approach to K- anonymizing social network data whereby data is released without modification as long as K-anonymity constraints are met, and is otherwise selectively withheld. This contrasts to existing approaches that release modified data, either distorted or generalized, to maintain K-anonymity. Social-K further offers a new and useful definition of K-anonymity in relation to social network queries. This privacy guarantee is defined in terms of set theory, which relates sets of users to sets of data, thereby allowing the problem to be solved using equivalent logic minimization algorithms, for which many efficient solutions exist. This Social-K solution is then tested with a proof-of-concept implementation which demonstrates that it is practical to employ our logic minimization approach to K-anonymize social networking data profiles.

7. References

[1] “Facebook statistics,” http://www.facebook.com/press/info. php?statistics.

[2] L. Sweeney, “Uniqueness of simple demographics in the U.S. population,” in LIDAPWP4, 2000.

[3] S. Muroga, Logic Design and Switching Theory. New York: Wiley, 1979.

[4] R. L. Rudell, “Multiple-valued logic minimization for pla synthesis,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/ERL M86/65, 1986. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/1986/ 734.html

[5]R.Fielding,“Representationalstate transfer (rest),” http://www.ics.uci.edu/_fielding/pubs/dissertation/rest arch style.htm.

[6] A. Beach, M. Gartrell, and R. Han, “Solutions to security and privacy issues in mobile social networking,” in SMW09: Workshop on Social Mobile Web at SocialCom 2009 in CSE ’09: Proceedings of the 2009 International Conference on Computational Science and Engineering. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1036–1042.

[7] “Quine-mccluskey algorithm (java),” http://en. literateprograms.org/Quine-McCluskey algorithm (Java).

[8] C. M. Gartrell, “Socialaware: Context-aware multimedia presentation via mobile social networks,” Master’s thesis, University of Colorado at Boulder, December 2008, http://www.cs.colorado.edu/_rhan/Papers/Mike Gartrell CU MS thesis-final.pdf.

[9] N. Ellison, C. Steinfield, and C. Lampe, “The benefits of facebook “friends:” social capital and college students’ use of online social network sites,” Journal of Computer-Mediated Communication, vol. 12, no. 4, 2007, http://jcmc.indiana.edu/ vol12/issue4/ellison.html.