Front.Comput.Sci.
DOI

SUPPLEMENTARY

A privacy – enhancing scheme against contextual knowledge – based attacks in location – based services

Online Resource 1 (Pseudocode of Algorithms)

1 Pseudocode of *DCA* Algorithm

According to the division of work, we implement our scheme in three sub-algorithms that will call each other during the process. Algorithm 1 and Algorithm 3 run on clients, and Algorithm 2 runs on Wi-Fi APs.

Algorithm 1: Client: DCA Sub-algorithm (issuing a query)

Input: target user U_t's standardized preference vector $\mathcal{W}_t^{'}$, an LBS query $Q(qtype,\ qdetail)$, real location $cell_t$, privacy profile k_t, distance preference μ, # of sets ns
Output: an optimal k-anonymity set AS

1 send $(\mathcal{W}_t^{'}, Q, cell_t, k_t, \mu)$ to AP (run Algorithm 2);
2 wait until AP returns CS to it; //Alg. 2 (Line 9) shows data structure of CS
3 **for** $(i = 0; i < \min(ns, \binom{3k_t}{k_t-1}); i++)$ **do**
4 construct set C_i with U_t and $k_t - 1$ other users (in set CS) at random;
5 $score_{C_i} = \sum_{j=1}^{k_t} (index\_prdiff_{ij} \cdot index\_dis_{ij} \cdot r_{ij})$;
6 **return** $\arg\max_{C_i}(score_{C_i})$;

Algorithm 1 demonstrates *DCA* Sub-algorithm which runs on the client of target user U_t (The target user refers to the user who issues the LBS query actually).

Firstly, it requests a candidate set CS from AP (Lines 1-2, run Algorithm 2). Then, it randomly generates several subsets of CS with U_t and $k-1$ other users in each one, and computes the score of each according to the differences in query probabilities, correlation coefficient and indexes of distance (Lines 3-5). Finally, the target user will filter out the most optimal anonymity set locally (Line 6).

Next, we present Algorithm 2 running on APs. After an AP receives U_t's query, Algorithm 2 will search for nearby cells with similar query probabilities of the same query type[1] (Line 3). Then, AP broadcasts U_t's standardized preference vector $\mathcal{W}_t^{'}$ continually to users who are located in those cells (Line 4). After that, AP processes each tuple returned from nearby users (Lines 5-9). The index of differences in type-j query probabilities between the real location $cell_t$ and other cells can be achieved by function $index\_prdiff = 1 - \frac{|pr - p_t^{qtype}|}{\beta}$. In addition, the size of a cloaking region is closely related to location privacy and QoS (which should be balanced by users). Consequently, we use the index of

Algorithm 2: AP: DCA Sub-algorithm (forwarding information)

Input: U_t's standardized preference vector $\mathcal{W}_t^{'}$, an LBS query $Q(qtype,\ qdetail)$, real location $cell_t$, privacy profile k_t, distance preference μ
Output: a candidate set CS

1 CS =NULL;
2 **for** $(d = 1; CS.size() < 3k_t; d++)$ **do**
3 searching for $cell_x$ in d-hop area around $cell_t$, $s.t.\ \forall x,\ |p_t^{qtype} - p_x^{qtype}| < \beta$;
4 send $\mathcal{W}_t^{'}$ to users who are located in these found cells (run Algorithm 3);
5 **while** $\exists$ tuples $(\widetilde{user},\ r)$ returned from users **do**
6 $index\_dis = e^{-\frac{(d-\mu)^2}{8}}$;
7 pr = getPr($user,\ qtype$); //retrieve (in database) the query probability of a cell where the user is located
8 $index\_prdiff = 1 - \frac{|pr - p_t^{qtype}|}{\beta}$;
9 add tuples $(\widetilde{user},\ r,\ index\_dis,\ index\_prdiff)$ to CS;
10 **return** CS;

distance $index\_dis = e^{-\frac{(dis-\mu)^2}{8}}$ to describe that preference [2]. Finally, all processed tuples are added into the candidate set CS (Line 9). If there aren't enough candidates in CS, AP will

[1] Threshold β (and θ in Alg. 3) is a system parameter, which can't be set by users.
[2] Parameter μ measured by # of hops on the grid-based map is determined by users.

E-mail: *cqjin@dase.ecnu.edu.cn*

extend searching areas (Line 2).

Algorithm 3, which is in charge of computing correlation coefficient between users' query preferences, is finally presented.

Algorithm 3: Client: compute_corr

Input: target user U_t's standardized preference vector $\mathcal{W}_t^{'}$, other user's own preference vector $\mathcal{W}_a$
Output: Pearson correlation coefficient between U_t and himself(herself)

1 standardize the vector $\mathcal{W}_a$ as $\mathcal{W}_a^{'}$;
2 $r = cov(\mathcal{W}_t^{'}, \mathcal{W}_a^{'})$;
3 **if** $r > \theta$ **then**
4 **return** $(\widetilde{user}, r)$; //user's ID will be replaced by a pseudonym by himself

2 Pseudocode of *enDCA* Algorithm

Algorithm 4: Client: enDCA Sub-algorithm (issuing a query)

Input: target user U_t's standardized preference vector $\mathcal{W}_t^{'}$, an LBS query $Q(qtype, qdetail)$, real location $cell_t$, privacy profile k_t, distance preference μ, # of sets ns
Output: an optimal k-anonymity set AS (or a cached set CAS)

1 send $(\mathcal{W}_t^{'}, Q, cell_t, k_t, \mu)$ to AP (run Algorithm 5);
2 wait until CS or CAS returned from AP ;
3 **if** CAS != $NULL$ **then**
4 /*An appropriate cached anonymity set is found*/
5 **if** $CAS.k == k_t$ **then**
6 return CAS;
7 **else**
8 /*$CAS.k > k_t$*/
9 randomly select $k_t - 1$ other users from CAS, and construct AS;
10 return AS;
11 **else**
12 run Lines 3-6 in Algorithm 1 (Clinet: DCA Sub-Algorithm);

We implement the enhanced scheme in three sub-algorithms as well according to their division of work. Algorithm 4 runs on clients, and Algorithm 5 runs on Wi-Fi APs. Algorithm 3 is still used to compute the correlation coefficient.

Algorithm 4 presents *enDCA* Sub-algorithm which runs on clients. Firstly, it requests a candidate set CS or a cached anonymity set CAS from AP (Lines 1-2, run Algorithm 5). If there exists an appropriate cached set, Algorithm 4 will construct the formal anonymity set AS with it (Lines 3-10). Otherwise, it follows the ordinary steps to construct an optimal anonymity set (Line 12).

Algorithm 5: AP: enDCA Sub-algorithm (forwarding information)

Input: U_t's standardized preference vector $\mathcal{W}_t^{'}$, an LBS query $Q(qtype, qdetail)$, real location $cell_t$, privacy profile k_t, distance preference μ
Output: a candidate set CS or a cached anonymity set CAS

1 CS =NULL;
2 T=NULL; //a variable used to store cached anonymity set temporarily
3 **foreach** t in $cache[qtype]$ **do**
4 **if** $t.k \geq k_t$ and $(\exists i \in [1, t.k],\ t.U_i == U_t)$ **then**
5 $T = T \bigcup \{t\}$;
6 **if** T != $NULL$ **then**
7 **return** $\arg\max_{t \in T}(\frac{\xi_t}{\log_2 t.k})$; //A cached anonymity set with high confusion degree is returned
8 run AP: DCA Sub-Algorithm($\mathcal{W}_t^{'}$, Q, shiftLocation($cell_t$), k_t, μ); //run Algorithm 2

Algorithm 5 illustrates *enDCA* Sub-algorithm running on APs. After an AP receives U_t's query, it will check in cache whether there exist appropriate anonymity sets. If several cached sets are found, a set with highest confusion degree will be returned (Lines 3-7). Otherwise, Algorithm 5 shifts U_t's real location first (Line 8), and then follows ordinary steps to construct a candidate set CS (Line 9).