Fingerprinting and De-anonymizing Mobile Users

Gene Moo Lee
University of Texas at Austin
gene@cs.utexas.edu

ABSTRACT

Mobile devices are the avatars in our daily lives. We surf webs, play mobile apps, search new information, enjoy location-based services, and do social networking with small but powerful devices. All these activities reveal a lot about ourselves, which makes big privacy concerns.

This paper conducts a passive attack against an anonymized mobile trace from an operational cellular provider. First, we profile each mobile user by extracting various features in mobile data activities. Then we analyze the feasibility to construct user fingerprints that can serve as quasi-identifiers. Lastly, we conduct two de-anonymization attacks using Facebook as an auxiliary public source.

1. INTRODUCTION

Mobile devices are the avatars in our daily lives. We surf webs, play mobile apps, search new information, enjoy location based services, and connect with friends with the powerful personal devices. All these activities in mobile networks reveal a lot about ourselves, which makes big privacy concerns.

Problem statements: In this project, we study mobile user activities by analyzing a mobile trace from an operational cellular service provider.

First we measure the diversity of mobile user activities to check the feasibility to extract fingerprints of individual users. The mobile activities in consideration include mobile web browsing activities, mobile app usages, search keywords, geographic locations, social networking activities, and so on.

Once we have mobile fingerprints from users, the next step is to link these mobile users to the identities in the publicly available auxiliary data sources. We will use social network data which provide various user information publicly.

2. DATA DESCRIPTION

In this section, we describe the datasets used in this project. Our main target is a cellular network trace from an operational cellular service provider. We try to extract various features to profile mobile users. For the auxiliary data source, we use Facebook, the most popular social network, which data can be obtained by web crawling. Lastly, we will describe how to build the ground truth of user mapping between the two datasets.

2.1 Mobile Traces: Our Target

First, we have collected real traces from an operational cellular service provider based in Chicago. This provider is one of the top ten largest wireless telecommunication networks in the US, serving 5.8 million customers in 26 states.

Currently, we have HTTP sessions of 222K users from one day in August, 2012. Based on the RADA [11] sessions, which keep track of the base station each user is accessing, we can map each HTTP session to the corresponding user. The user ID comes with the format of `phone# @provider.com`.

The 2.6 TB trace includes all the HTTP sessions generated by customers. Encrypted HTTPS sessions are not included. In this project, we will focus on HTTP GET requests each device has generated, because the information we can obtain from HTTP GET requests overwhelms other sources such as POST requests and GET responses. We have not conducted deep packet inspection (DPI) to analyze payload information due to the computation overhead.

Schema in HTTP session: For each HTTP session, we have the following header information: (1) session ID, (2) timestamp, (3) protocol, (4) client IP, (5) server IP, (6) host, (7) path, (8) key-value information. In addition to the schema we had in the proposal, we now have HTTP cookie information as well, which is a valuable source to understand the HTTP sessions.

Mobile activities: The list of mobile user activities we focus is the following:

- hosts (H): full domains, second-level domains
- mobile apps (A): app names, categories
- location (L): latitude and longitude, zip code
- facebook (F): Facebook numeric IDs

Table 1 summarizes the patterns used to extract various high level user activities from the cellular trace. We will describe the detailed methods to extract these activities below.

Category	Host	Keys	Value
Host	*	N/A	domain name
App	*	msid	app name
App	*	app_name	app name
App	*	appname	app name
App	*	bundle	app name
App	*	bundleId	app name
Location	*	lat, lng	latitude, longitude
Location	*	lat, lon	latitude, longitude
Location	*	lat, long	latitude, longitude
Location	*	latitude, longitude	latitude, longitude
Facebook	*.facebook.com	id	Facebook ID
Facebook	*.facebook.com	a	Facebook ID
Facebook	*.facebook.com	tpi	Facebook ID
Facebook	*.facebook.com	actorid	Facebook ID
Facebook	*.facebook.com	_user	Facebook ID
Facebook	*.facebook.com	ids[?]	Facebook ID

Table 1: Patterns used to extract mobile activities in the cellular trace.

Hosts (H): As each HTTP header specifies the host information, we can clearly see which domain the user accessed. Note that host information is even available in HTTPS session. Even though a passive listener cannot *look inside* the HTTPS sessions, still she can learn who is talking to whom.

We might need to differentiate hosts that are manually visited by web browsing from those accessed by machine codes from mobile apps. For example, if a user visits `austin.craigslist.org`, it is obvious that she is interested in selling or buying goods. However, other hosts accessed by mobile apps (*e.g.*, `data.flurry.com`, `androidsdk.ads.mp.mydas.mobi`) do not clearly reveal the nature of user activities. To handle this, we can analyze host collocation. Assume we know some sessions are from mobile apps for sure. Then the temporally neighboring hosts are likely to come from apps as well. If we aggregate the collocation information, we can find out the set of hosts are very likely to come from mobile apps. We may also use top sites ranked by Alexa [1] for the same purpose.

We can use host information in two ways: to use the host as it is or to extract the second level domain. In case a service is depending on multiple redundant servers for load balancing, the second level domain will group multiple server instances while giving enough information about the service. For example, `static.ak.fbcdn.net` and `profile.ak.fbcdn.net` will be mapped to `fbcdn.net`.

Mobile apps (A): From the studies about Android apps [6] and iOS apps [4], we know that over 80% of the free mobile apps reveal personal information of the mobile users to advertisement/analytics networks (*e.g.*, `doubleclick.net`, `admob.com`, `data.flurry.com`). In order to identify themselves to the ad networks, mobile apps use specific keys (`msid`, `app_name`, `bundle`) in the key-value chain. We take advantage of this to identify which mobile apps users are using.

We can treat each mobile app as an individual attribute. Another option is to use category information to group individual apps together. Google Play categorizes each app into games or applications. We will leave this as a future direction.

Figure 1 shows a HTTP GET request to `doubleclick.net`, which contains the name of mobile app the user was running (`msid=com.igoldtech.an.swiped`), which is a puzzle game according to the category from `play.google.com`. In addition, it reveals other personal information such as gender (`cust_gender=2`), language (`hl=en`), and time zone (`u_tz=300`).

```
sessionID: 3743896369240254493
timestamp: 1340161434|06-20-2012|CST|CDT
protocol: HTTP|URLP|GET
client IP: 10.221.4.49
server IP: 74.125.225.154
host: googleads.g.doubleclick.net
path: /mads/gma
key-value: msid=com.igoldtech.an.swiped
&format=320x50_mb&net=ed&hl=en
&cust_gender=2&u_audio=1&u_so=p
&output=html&region=mobile_app&u_tz=300
```

Figure 1: Trace example: mobile application

Locations (L): For location based services such as local search and weather updates, mobile devices should reveal geographic locations to the service providers. We can take advantage of this aspect to extract the trajectories of mobile users. Especially, weather services even reveals user locations periodically for frequent updates.

Figure 2 is a HTTP GET request to search using `google.com`. In the key-value information, we know that the search keyword (`q=chris bosh`), platform (`platform=android`), latitude (`41.801719`), longitude (`-87.616753`), region (`gl=US`), and language (`hl=en`).

```
sessionID: 10610718480380941993
timestamp: 1340161434|06-20-2012|CST|CDT|
protocol: HTTP|URLP|GET
client IP: 166.226.44.25
server IP: 74.125.225.209
host: www.google.com
path: /m/gne/suggest/v2
key-value: q=chris+bosh+&hl=en&app=iss
&appv=133247963&platform=android&gl=US
&sll=41.801719,-87.616753&acc=962
```

Figure 2: Trace example: search keyword and location.

Once GPS coordinates are extracted, we convert this fine-grained information to the ZIP codes using a public ZIP code database [3]. We may choose different levels of granularities such as city or state. However, we will focus on ZIP codes for this study.

Facebook IDs (F): Analyzing the trace, we know that more than 20% of cellular traffic is related to Facebook, and more than 90% of users are using Facebook. As our auxiliary data is Facebook, understanding Facebook traffic from inside data will be useful in our analysis. Specifically, we are interested in the Facebook IDs that appear in Facebook sessions.

There are a variety of key-value pairs in the Facebook HTTP sessions and we need to figure out which key-value chains corresponds to Facebook IDs. By cross checking with

the ground truth mapping and Facebook friend list data, we came up with a few host-key patterns described in Table 1.

Figure 3 shows an example of a HTTP GET request to Facebook. The key `__user` indicates the Facebook numeric ID.

```
sessionID: 223514164345848579
timestamp: 1340161454|06-20-2012|CST|CDT
protocol: HTTP|URLP|GET
client IP: 166.221.26.179
server IP: 66.220.149.97
host: m.facebook.com
path: /ds/search.php
key-value: filter%5B0%5D=user&filter%5B1%5D=page
        &filter%5B2%5D=group&filter%5B3%5D=event
        &filter%5B4%5D=app&max_results=30
        &context=mobile_search_m_site
&viewer=AfMqvQ9KrLpG9vmu
        &q=crystal&m_sess=&__user=100001573141234
        &__ajax__=true&__metablock__=23
```

Figure 3: Trace example: Facebook ID

Note that the Facebook IDs we have collected may or may not be the IDs of the cellular account owner. They may correspond to the identity of the mobile user herself or that of her friend. But with high probability, the Facebook ID will be closely related to the cellular user.

2.2 Facebook: Auxiliary Data

Now we describe the Facebook data that will serve as the auxiliary information that contains the user identities.

2.2.1 Data Collection

We have developed two crawlers to collect Facebook data: profile crawler and friend list crawler. Unlike other social network services (*e.g.*, Twitter, Foursquare), Facebook APIs have a lot of restrictions in the purpose of data collection.

Screen scraping: Therefore we chose an alternative approach to use screen scraping. The idea is to pretend to be a normal Facebook user who is using a standard browser to access the service. We implement Ruby scripts to initiate a browser and login to Facebook as a normal Facebook user. Then we make HTTP requests with the following two URLs to download profile pages and friend list pages:

```
http://www.facebook.com/profile.php?id=#{FBID}&sk=info
http://www.facebook.com/ajax/browser/list/allfriends
/?uid=#{FBID}&infinitescroll=1&location=
friends_tab_tl&start=#{offset}&__a=1"
```

Figure 4: Facebook URLs to collect profile pages and friend lists.

Facebook IDs: Now the question is how do we get the list of Facebook IDs to collect. For this project, we have used the Facebook IDs obtained from the HTTP cookies, which will be described in Section 2.3.

The caveat is that collected data is from known targets with the help of the ground truth. In reality, we can use *seeds*, a handful of Facebook IDs that we know the true mapping to our target cellular users. We can first start from the seed Facebook users to collect their profiles and friends.

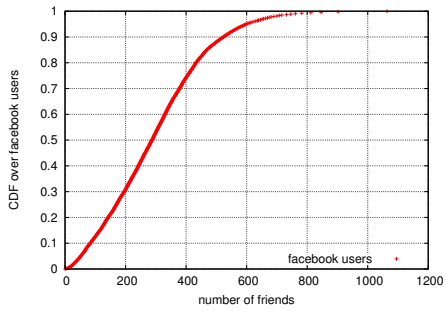


Figure 5: Node degrees on Facebook graph.

Once the seed crawling is done, we can collect from the one degree friends of seeds in a depth first search manner. In our current analysis, we only have seed data now and we are in the process to collect more information from the friends of seeds.

2.2.2 Facebook Graph

Basic statistics: We have crawled 16439 Facebook friend lists. Then we construct a graph $G_{FB} = < V_{FB}, E_{FB} >$, where V_{FB} is the set of all Facebook IDs appearing in the friend lists and E_{FB} is the set of Facebook friendship. Table 2 shows the basic statistics on the Facebook graph.

$\|V_{FB}\|$	2,627,000
$\|E_{FB}\|$	6,724,000
# of friend lists	16,439
average node degree	205
median node degree	300

Table 2: Basic statistics on collected Facebook Graph

Node degree: The average node degree is 205, which is much higher than the numbers from other social network data (*e.g.*, Twitter, LiveJournal, Flickr) in [10]. The CDF of node degrees is shown in Figure 5.

2.2.3 Facebook Profiles

In a Facebook profile page, we can collect the following user information: name, gender, current city, hometown, employers, education, languages, etc. However, due to privacy concern, many Facebook users reveal limited information to the public. Our analysis is based on the public information we can access.

Overview: From the 7740 collected profile pages, 6821 (88%) reveal the gender information. Facebook users tend to let gender, high school, current city, hometown information to be public, whereas other features are mostly private such as languages, websites, relationship status. Table 3 shows the statistics on top five public information.

Gender and relationship: Out of 6820 public gender information, 3973 (58%) and 2847 (42%) are female and male, respectively. From the public relationship information, *single* ranked first (46%), followed by *married* (23%) and *in a relationship* (23%) as shown in Table 4.

Information	Count	Percentage
# profiles	7,740	100%
w/ gender	6,821	88%
w/ high school	3,954	51%
w/ current city	3,902	50%
w/ hometown	3,538	46%
w/ history	3,164	41%

Table 3: Public information in Facebook profiles.

An interesting experiment would be to predict the gender or relationship status based on the cellular trace we have, which may be a follow-up study.

Relationship	Count	Percent
single	844	46%
married	430	23%
in a relationship	426	23%

Table 4: Relationship in Facebook profiles.

Location: As the service provider is headquartered at Chicago, a majority of the customers are geographically close to the greater Chicago area. Top ranked are major cities in the states of Illinois, Wisconsin, Iowa, North Carolina, Tennessee as shown in Table 5.

Rank	Current City	Hometown
1	Chicago, IL	Chicago, IL
2	Milwaukee, WI	Milwaukee, WI
3	Madison, WI	Des Moines, IA
4	Greenville, NC	Knoxville, TN
5	Knoxville, TN	Tulsa, OK

Table 5: Location information from Facebook profiles.

2.3 Ground Truth

Now we discuss about the ground truth of mapping between cellular users and social network users. Let $G_{MT} = (V_{MT}, E_{MT})$ be the graph constructed by the mobile trace where V_{MT} is the set of nodes (*e.g.*, cellular users) and E_{MT} is the set of edges (*e.g.*, relationship between users) in the graph.

Node mapping ($V_{MT} \rightarrow V_{FB}$): As explained earlier, we have HTTP cookie information in the trace. So we can extract the exact Facebook ID used to login to the service. This gives us a ground truth knowledge to connect Facebook IDs to the cellular users.

Based on the collected HTTP cookies, we have identified $16,162$ cellular users to their corresponding Facebook IDs. Note that some cellular accounts are being used by multiple people (*e.g.*via tethering). And there are cases where multiple Facebook IDs are mapped to a single cellular account. In this case, we accept the mapping as long as one Facebook ID is not mapped to multiple cellular users.

Notes on node mapping: We want to do some back-of-the-envelope calculation on the mapping between the cellular trace and Facebook data. Our target cellular provider is serving 6 million customers in the US and total US population is 311 million according to Google Public data [14]. Given a random US person, the probability the the person uses the target cellular provider is about 5%. Recall that the average Facebook node degree is 205, then the number of friends who is using our target cellular service will be about 4.

Edge mapping ($E_{MT} \rightarrow E_{FB}$): Based on the node mapping between mobile traces and Facebook, we can also infer the relationship (V_{MT}) within mobile traces. Table 6 shows the information about G_{MT}.

As our node mapping only covers a small subset of Facebook users, the edge coverage is also limited. The average node degree of G_{MT} is around 2, whereas that of G_{FB} is 200. It seems that this affects the performance of de-anonymization process, which will be discussed in Section 4.

$\|V_{MT}\|$	6,950
$\|E_{MT}\|$	15,131
average node degree	2.18

Table 6: Basic statistics on mobile user graph.

3. MOBILE USER FINGERPRINTING

In this section, we first describe the construction of user attributes from the cellular trace. Then we check the feasibility of extracting user fingerprints from the attributes. Lastly, we conduct another feasibility study to check if the attributes can be served as methods to reconstruct social relationship within the cellular users.

3.1 Constructing User Profiles

In our user database, each record corresponds to a mobile user in the trace. Then the attributes are generated from the raw data (H, A, L, F). Based on the properties of each information, we will transform raw information to the attributes in the records.

Counting vs. binary: When we construct the attribute vector for each user, we may make the attribute value to be binary (just counting the existence) or weights (to emphasize the frequency of each attribute). For this study, we will ignore the frequency but just consider the existence of non-zero attributes.

Example user: Figure 6 shows an example of a user profile extracted from the cellular trace. Based on the majority voting, we can guess the Facebook ID: `100000325429162`. We can also confirm the Facebook ID by the HTTP cookie. Based on the search keywords, we know that this user is interested in used cars, credit reports, Mike Duman auto sales, and drug (metformin hcl). From apps, we know that the user plays fishing game and likes military wallpapers. From location, the user is related in Virginia and Oregon.

Statistics: Table 7 shows the overall statistics on the constructed user profiles. In total, we have 222 thousand profiles and 99% of them have host information, which has the

```
Cellular ID: 2079312010@uscc.net

Keywords (K):
blue+book+value+for+cars+used 4
free-annual-credit-report.com 3
mikeduman 3
metformin+hcl 2

Apps (A):
jp.pascal.bassfishingfree 8
TN_NAVSERVICE 3
com.best.wallpapers.Military 2

Locations (L):
36.44,-76.76 6
36.62,-76.66 6
42.34,-122.84 5

Facebook ID (F):
100000325429162 38
583963896 2
100000500469179 2
```

Figure 6: User profile example

highest coverage in terms of the number of users. Only 10% of profiles include mobile app information, 26% has location features, and 6% has at least one Facebook ID in the trace. 25% of users have accessed a Facebook domain and 7% of them left cookie information, which is used as a ground truth of identity mapping (see Section 2.3).

	Count	Percentage
# total users	222,874	100.00%
H: # users with hosts	222,680	99.91%
A: # users with mobile apps	24,468	10.98%
L: # users with location	58,389	26.20%
F: # users with FB IDs from traffic	14,083	6.32%
# users with Facebook traffic	55,842	25.06%
# users with Facebook cookies	16,162	7.25%

Table 7: Statistics on user profiles.

Number of attributes per user: Figure 7 shows the CDFs of the number of unique attributes per cellular user. We can see that host is again the richest source of information, followed by Facebook IDs, mobile apps, then location information. Facebook IDs will later be used to de-anonymize the cellular users to the Facebook ID based on the friendship information in Section 4. Mobile app information is quite limited in terms of variety, which may limit the expressiveness as a source of fingerprinting. Interestingly, location information is also quite limited. This may be due to the short duration of the trace we have processed.

3.2 User Identification by Features

Once we have constructed the user profiles, we want to check if the $HALF$ information can serve as the quasi-identifier. First, we will define the similarity measure to quantify how a pair of user profiles are close to each other. Then we will see which attributes can be strong differentiators.

Similarity measures: We use the same definition by [9] to

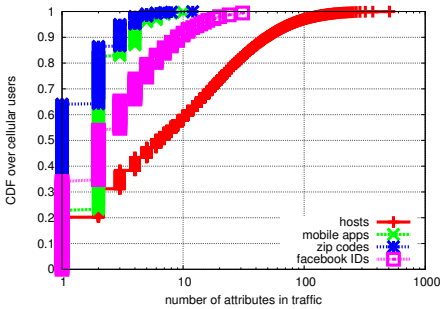


Figure 7: Number of support features for cellular users.

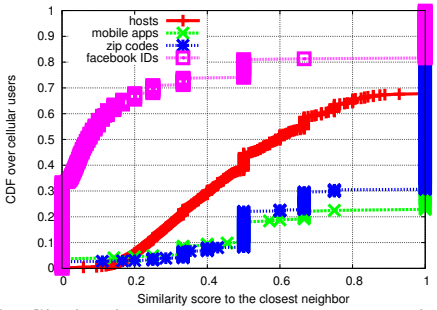


Figure 8: Similarity scores to the closest neighbors for cellular users.

measure the similarity between two records in sparse databases. We need to first define the similarity measure SIM between two individual attributes. The similarity between two individual attributes will be binary, where the function returns 1 if both are non-zeros; it returns 0 otherwise. Then the similarity measure SIM between two profiles is defined as:

$$SIM(prof_1, prof_2) = \frac{\sum_i SIM(attr_{1i}, attr_{2i})}{|supp(prof_1) \cup supp(prof_2)|} \quad (1)$$

where $supp(prof)$ is the set of non-zero attributes in the profile $prof$.

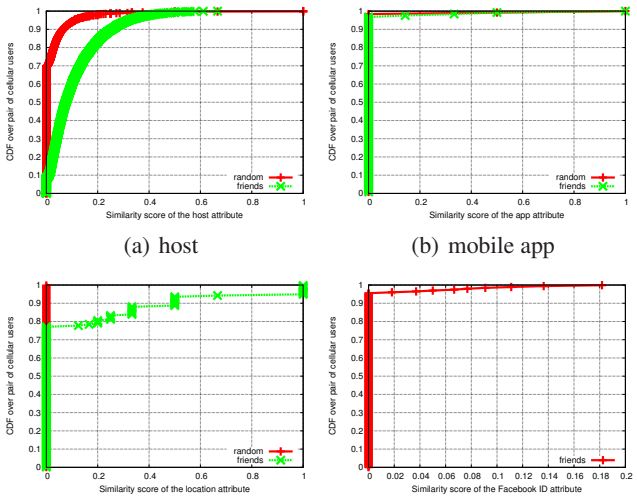
Similarity of each attribute: In order to check how each profile is distinguishable against other profiles, we measure the similarity measures to the closest neighboring profile (which gives the highest similarity score). Figure 8 shows the CDF of similarity scores to the closest neighbors.

We can see the Facebook IDs give the largest distinguishability with smallest similarity scores against the closest profiles. Then host information follows next. On the other hand, ZIP codes and mobile apps features do not provide enough expressiveness to create user fingerprints.

3.3 Reconstructing Social Graph

In order to conduct a node de-anonymization attack using graph structures, we need to re-construct the social graph of cellular users. If we had the call/SMS records, it could be easily constructed. However, our trace only includes the mobile data traffic, so the question is how we can infer the social relationship based on our constructed HALF information.

The idea is that if two users have very similar profiles, it is likely that they are related to each other. For example, if two



(a) host (b) mobile app

(c) location (d) facebook ID

Figure 9: CDF of similarity scores between random pairs and friends.

users co-located in multiple ZIP codes, they may be friends. Similarly, we may think that socially closed users are likely to use similar mobile apps and visit similar hosts.

In order to check this hypothesis, we first calculate the similarity scores between friends in cellular traces using the graph G_{MT} from Section 2.3. Then we randomly pick two cellular users to measure the similarity. We use the same similarity measure from Section 3.2.

Figure 9 shows the CDF of similarity scores of random pairs and Facebook friends. For Facebook IDs and ZIP codes, all randomly chosen pairs do not share any common attributes, resulting scores of 0, whereas Facebook friends share some common Facebook IDs or ZIP codes. However, friends do not share much mobile apps, so apps cannot be used to reconstruct social relationship between cellular users. Table 8 shows the averages, maximum values, and standard deviations for each attribute.

Feature	Avg	Max	Std
Host (Random)	0.02	1.00	0.07
Host (Friend)	0.11	1.00	0.10
App (Random)	0.01	1.00	0.10
App (Friend)	0.02	1.00	0.10
Location (Random)	0.00	0.00	0.00
Location (Friend)	0.12	0.00	0.26
Facebook ID (Random)	0.00	0.00	0.00
Facebook ID (Friend)	0.12	0.00	0.26

Table 8: Comparing similarity scores between random pairs and friends.

4. DE-ANONYMIZATION ATTACKS

In this section, we conduct two de-anonymization attacks against the mobile trace. First, we will compare the Facebook IDs appearing in the trace against the friend list in G_{FB}. Next, we use graph structures to conduct a node de-anonymization attack. In doing so, we modify the algorithm from [10].

4.1 De-anonymizing by Facebook IDs

Algorithm: Our first de-anonymization attack is using a very straightforward approach to use Facebook IDs appearing the the mobile trace. We collect Facebook IDs ($attr_{FB}$) based on the method from Section 2.1. Our assumption is that those Facebook IDs will closely related to the originating cellular user. It could be her own Facebook ID or those from her friends.

For each set of Facebook IDs ($attr_{MT}$) from a mobile user, we compare it against the friend lists ($friends_{FB}$) in the Facebook graph G_{FB}. If $attr_{MT}$ is a subset of $friend_{FB}$, we save this Facebook ID in the candidate set. If there is only one such Facebook ID, we conclude that the mobile user can be mapped to the Facebook ID. Figure 10 shows the detailed algorithm.

```
for mobile_user in V_MT:
  confusion_set = {}
  FBID_traffic = (facebook IDs from mobile traffic)
  if | FBID_traffic | < THRESH: continue
  for facebook_id in V_FB:
    FB_friends = (facebook IDs from friends and self)
    if FBID_traffic is a subset of FB_friends:
      confusion_set.add(facebook_id)
  if |confusion_set| == 1:
    mapping[mobile_user] = confusion_set
```

Figure 10: De-anonymization algorithm based on Facebook IDs.

Precision: We evaluate the performance using two metrics: precision and coverage. Precision is defined as $\frac{TP}{TP+FP}$, where TP is the number of true positive mappings and FP is the number of false positive mappings (*i.e.*, mapped to a wrong Facebook ID). Note that precision is only calculated over the ground truth mappings, where we know the true answers by HTTP cookie information. Figure 11 shows the results. As we have more numbers of Facebook IDs, our precision goes up. With one Facebook IDs the precision is 80%, which is still a good number, we can achieve up to 97% with three Facebook IDs.

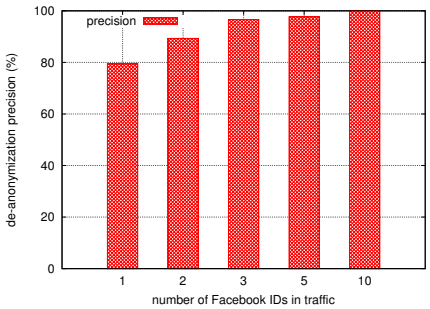


Figure 11: Precision of de-anonymization by Facebook IDs.

Coverage: Coverage is defined as $\frac{|Mapping \cap TrueMap|}{|TrueMap|}$, where $Mapping$ is our constructed mapping and $TrueMap$ is the ground truth mapping achieved by HTTP cookies. Figure 12 shows that with lower threshold of 1, we can de-anonymize

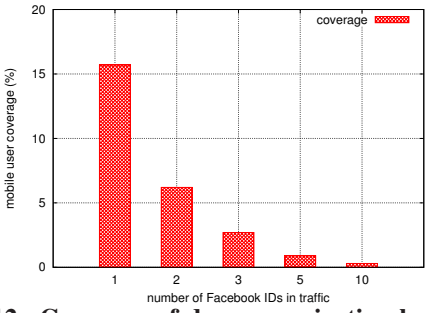


Figure 12: Coverage of de-anonymization by Facebook IDs.

more than 15% of our ground truth. However, increased thresholds will greatly reduced the coverage.

4.2 De-anonymizing by Graph Structures

Next, we will conduct a more sophisticated de-anonymization attack using the graph structure of two graphs G_{MT} and G_{FB}. This work is based on the algorithm from [10].

Algorithm: Briefly explaining the algorithm, it starts with *seed* mappings. Those seeds are true mappings between two graphs, which can be obtained by other channels such as user IDs, user real names, etc. In our evaluation, we use HTTP cookie to construct seeds.

Once seeds are generated, we try to *propagate* the knowledge to the neighboring nodes. If the link structure of two nodes are very similar with high *eccentricity*, we map them together. We repeat this propagation step until convergence.

One difference between the original problem setting from ours is that we know the Facebook IDs appearing in the traffic. Essentially, we know the set of Facebook nodes that are possible friends of a given mobile user. We can take advantage of this to update the mobile graph by adding an edge of two mobile nodes if new mapping reveals that two mobile users are friends over Facebook network.

The detailed algorithm is described in Figure 13.

Constructing mobile graphs: In the problem definition from [10], we know the graph structures of both sanitized and auxiliary ones. However, in our context, it is not clear how cellular users are socially connected to each other.

We can create mobile graphs based on the attributes we constructed. A simple approach is to create an edge between mobile users if the similarity measure of two profiles are above some threshold. However, the constructed graphs did not give good de-anonymization performances, so it seems that constructed graphs do not reflect the true relationship. We need to find a better way to re-construct social relations based on the mobile activities.

For now, we use the true social graph of G_{MT} where the edges are constructed based on the friendship from Facebook graph G_{FB}.

Evaluation: Again, we evaluate our de-anonymization attack with two metrics: precision and coverage. Figure 14 shows the precision results. In all experiments varying the number of seeds, the precision was perfect.

```
// new function
function updateGraph(lgraph, lgraph_cand_nbr, map)
  for lnode in map:
    rnode = map[lnode]
    for (rnode, rnbr) in rgraph.edges:
      if rnbr not in invert(map): continue
      lnbr = invert(mapping)[rnbr]
      if (lnode, lnbr) not in lgraph.edges:
        lgraph.add_edge(lnode, lnbr)
    for (rnbr, rnode) in rgraph.edges:
      if rnbr not in invert(map)[rnbr]: continue
      lnbr = invert(map)[rnbr]
      if (lnode, lnbr) not in lgraph.edges:
        lgraph.add_edge(lnode, lnbr)

function propagationStep(lgraph, rgraph, map)
  for lnode in lgraph.nodes:
    scores[lnode] = matchScores(lgraph,rgraph,map,lnode)
    if eccentricity(scores[lnode]) < theta: continue
    rnode = (pick node from rgraph.nodes where
               scores[lnode][node] = max(scores[lnode]))
    scores[rnode] =
        matchScores(rgraph, lgraph, invert(map), rnode)
    if eccentricity(scores[rnode]) < theta: continue
    reverse_match = (pick node from lgraph.nodes where
                scores[rnode][node] = max(scores[rnode]))
    if reverse_match != lnode:
      continue
    map[lnode] = rnode
  return map

function matchScores(lgraph, rgraph, map, lnode)
  initialize scores = [0 for rnode in rgraph.nodes]
  for (lnbr, lnode) in lgraph.edges:
    if lnbr not in map: continue
    rnbr = map[lnbr]
    for (rnbr, rnode) in rgraph.edges:
      if rnode in map.image: continue
      scores[rnode] += 1 / rnode.in_degree^0.5
  for (lnode, lnbr) in lgraph.edges:
    if lnbr not in map: continue
      rnbr = map[lnbr]
    for (rnode, rnbr) in rgraph.edges:
      if rnode in map.image: continue
      scores[rnode] += 1 / rnode.out_degree^0.5
  return scores

function eccentricity(items)
  return (max(items) - max2(items)) / std_dev(items)

// main starts here.
until convergence do:
  updateGraph(lgraph, lgraph_cand_nbr, map)
  map = propagationStep(lgraph, rgraph, map)
```

Figure 13: De-anonymization algorithm based on graph structures.

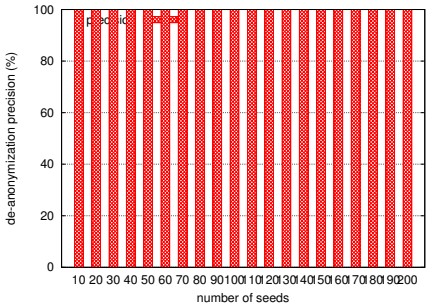


Figure 14: Precision of de-anonymization by graph structures.

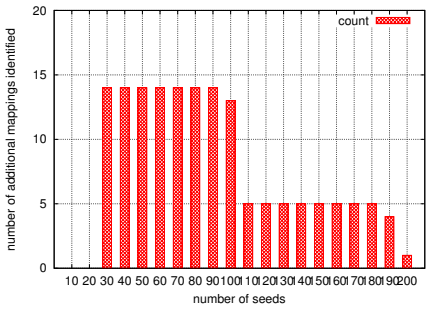


Figure 15: Number of additional mappings of de-anonymization by graph structures.

However, the performance is not satisfactory in terms of coverage. We totally have more than 16 thousand true mapping by HTTP cookies. However we could only de-anonymize up to 13 additional mappings in all experiments, regardless of the number of seeds used. We can not experience any large scale cascading yet. Figure 15 shows the results.

Note on the results: We think that the low node degree is the reason why massive cascading was not possible. The average node degrees are from 29.3 to 37.7 in graphs from [10]. Even our Facebook graph has very high node degree of 205, the mobile graph has very low average node degree of 2.18. So the remaining question is how we can de-anonymize graphs in case of substantial differences in sizes.

5. RELATED WORK

De-anonymization: A series of works have been done in the field of de-anonymization: [2], [15], [10], [13], [5], [7], [16], [8], [9], [12]. Detailed description and comparison will come later.

6. CONCLUSION

In this project, we conduct passive attacks against a mobile trace from an operational cellular service provider. Based on the HTTP session information, we can profile mobile users in various dimensions: hosts, apps, location, social networks. We show that very detailed user profiles can be obtained by simply observing the mobile sessions. We show the feasibility to use the constructed user profiles as quasi-identifiers and to re-construct social relations between cellular users. Lastly, we conduct two de-anonymization attacks using Facebook as an auxiliary data source. The first attack uses Facebook ID to achieve high re-identification accuracy but with limited coverage. We also show preliminary results on the second attack using the graph structures.

7. REFERENCES

[1] Alexa top sites. `http://www.alexa.com/topsites`.

[2] L. Backstrom, C. Dwork, and J. M. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *WWW*, 2007.

[3] Civic Space US ZIP code database. `http://www.boutell.com/zipcodes/zipcode.zip`.

[4] M. Egele, C. Kruegel, E. Kirda, and G. Vigna. PiOS: Detecting privacy leaks in iOS applications. In *NDSS*, 2011.

[5] J. Freudiger, R. Shokri, and J.-P. Hubaux. Evaluating the privacy risk of location-based services. In *Financial Cryptography*, 2011.

[6] P. Hornyack, S. Han, J. Jung, S. E. Schechter, and D. Wetherall. These aren't the droids you're looking for: retrofitting Android to protect data from imperious applications. In *CCS*, 2011.

[7] A. Masoumzadeh and J. B. D. Joshi. Anonymizing geo-social network datasets. In *SPRINGL*, 2011.

[8] A. Narayanan, H. Paskov, N. Z. Gong, J. Bethencourt, E. Stefanov, E. C. R. Shin, and D. Song. On the feasibility of Internet-scale author identification. In *S&P*, 2012.

[9] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *S&P*, 2008.

[10] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *S&P*, 2009.

[11] RADIUS Accounting RFC 2866. `http://tools.ietf.org/html/rfc2866`.

[12] R. Shokri, G. Theodorakopoulos, J.-Y. L. Boudec, and J.-P. Hubaux. Quantifying location privacy. In *S&P*, 2011.

[13] M. Srivatsa and M. Hicks. Deanonymizing mobility traces: Using social network as a side-channel. In *CCS*, 2012.

[14] U.S. Population. `http://goo.gl/w7zvw`.

[15] G. Wondracek, T. Holz, E. Kirda, and C. Kruegel. A practical attack to de-anonymize social network users. In *S&P*, 2010.

[16] H. Zang and J. Bolot. Anonymization of location data does not work: a large-scale measurement study. In *MobiCom*, 2011.