Privacy-Preserving Mobility-Casting in Opportunistic Networks

G. Costantino, F. Martinelli, P. Santi

IIT TR-23/2013

Technical report

Dicembre 2013
Abstract—In this paper, we introduce the notion of mobility-cast in opportunistic networks, according to which a message sent by a node $S$ is delivered to nodes with a mobility pattern similar to that of $S$—collectively named place-friends. The motivation for delivering a message to place-friends stems from the fact that current social acquaintances are likely to be place-friends. Most importantly, it has been recently found that a large fraction of new social contacts comes from place-friends.

After introducing mobility-cast, we present a privacy-preserving mobile-cast protocol based on the MobileFairPlay platform for secure two-party computation in mobile environments. The effectiveness of the protocol in delivering messages to place-friends is demonstrated by means of simulations based on a real-world GPS trace.

Finally, in the last part of the paper we present an implementation of mobile-cast on the Android platform, and test its computational performance on a number of different smartphones. Overall, the results presented in this paper show that privacy-preserving mobile-cast can be effectively implemented with current mobile phone technology.

I. INTRODUCTION

While mobile social networks have attracted considerable interest in the research and industrial community in recent years, some issues are still to be solved before they gain full acceptance in the user community. First, if opportunistic communications are used to drive the information dissemination process, forwarding mechanisms should be designed that are able to deliver information to all and only the interested users, so to reduce spamming of messages throughout the network. Furthermore, privacy issues should be carefully considered when designing a mobile social application. On the one hand, knowledge of private user information such as interest profile, mobility pattern, social ties, etc., has been proved very useful in improving the information propagation process within the network [5], [7], [9], [11]. On the other hand, users are increasingly reluctant in sharing such sensible information with strangers, which motivate the need for protocols that consider user privacy in the design cycle by exploiting the privacy-by-design concept.

In this paper, we try to address the above described issues by introducing an innovative information dissemination mechanism, called mobility-cast, and by presenting a privacy-preserving implementation of a mobility-cast protocol. The idea at the core of mobility-cast is delivering a message $M$ generated by a user $U$ to users who display a mobility pattern similar to $U$’s one. Following [12], in this paper we call such set of users the place-friends of user $U$. As described in greater detail in Section III, mobility-cast finds its motivation in the observation that not only current social acquaintances are likely to be place-friends, but also a large fraction of new social contacts comes from place-friends.

The mobility-cast protocol that we introduce, which we call $2H$ since information is propagated only up to the second communication hop, is built upon a secure function to estimate place-friendships between two users. As carefully analyzed in Section VI, the function is secure in the sense that, after its execution, a party only acquires minimal information about the other party’s mobility profile. The amount of information disclosed to the other party can be controlled through a design parameter of the protocol.

In this paper, we show $2H$ effectiveness not only in preserving user privacy, but also in actually delivering information precisely to place-friends: the results of simulation experiments performed on a real data trace show that $2H$ strikes the best compromise between coverage, precision, and cost, amongst the protocols evaluated in the experiments—see Section VII for details.

Finally, we report the results of measurements we have performed on different smartphones with the goal of estimating the running time of the secure place-friendship estimation function at the core of $2H$. The results have shown that running times are acceptable (as low as 10 sec) even with current smartphone technology.

II. RELATED WORK

A number of recent studies have shown that the effectiveness of the information propagation process can be improved by having users exchange some type of personal information to drive the process. In [5], [7], the authors show that the effectiveness of unicasting a message to destination is improved by considering user social metrics (e.g., centrality in the social network graph) in the forwarding process. The authors of [11] instead proposed exchanging user interest profiles in order to deliver messages only to interested users. A work which is closer in spirit to ours is [9], where the authors propose to use the user mobility profile to drive message forwarding. However, the focus in [9] is in unicasting a message to a specific destination, while in this paper we aim at implementing a novel communication primitive, namely, mobility-casting. Furthermore, privacy issues are not considered in [9], and mobility profiles are exchanged in clear between users.

Another line of research which is related to our work is privacy-preserving protocols for opportunistic networks. Most of existing approaches focus only on securely computing whether two mobile users are “friends”, where the specific definition of friendship depends on the approach at hand.
A few protocols consider network-wide information propagation protocols built on top privacy-preserving “friendship” estimation. In [1], the authors introduce a privacy-preserving protocol for geo-casting a message to a specific geographic location. In [2], the authors of this paper present a privacy-preserving approach for implementing the interest-casting primitive introduced in [11], and analyze the privacy-preservation vs. forwarding accuracy tradeoff which is inherent in the design. The interest-casting protocol has been implemented within the MobileFairPlay platform for secure two-party computation in mobile devices [3].

III. Motivation

We want to implement a communication primitive for opportunistic networks which delivers a copy of message $M$ generated by user $A$ to all nodes in the network with a mobility pattern similar to $A$. In accordance with [12], we call the set of nodes with a mobility pattern similar to node $A$ the place-friends of node $A$. How to formally define a user’s mobility pattern and a similarity metric between mobility patterns (and, hence, the set of place-friends) is deferred to later sections. We call the communication primitive that delivers a copy of the message to place-friends mobility-cast.

Mobility-cast is motivated by the observation that social interactions often occur between individuals with similar mobility patterns: e.g., colleagues who work in the same place, friends attending the same fitness class, etc. This intuitive observation is quantitatively evaluated in [4], where the authors show that social ties between people can be inferred with a good accuracy from co-occurrence in time and space. Hence, delivering a message to node $A$’s place-friends is likely to reach many relevant social ties of node $A$. More importantly, it has been recently shown [12] that a large fraction (about 30%) of new social interactions arise between place-friends. Similar observations have been done in [16], where it is shown that individuals with similar mobility patterns are likely to be close in the social network graph formed of the phone calls between users. Thus, delivering a message to place-friends is useful not only to reach current social ties, but also forthcoming social ties. Indeed, we can imagine that a mobile-cast primitive might even increase the fraction of social interactions between place-friends well beyond the 30% value observed in [12]. Suppose individual $B$, who is a stranger but place-friend of node $A$, receives an interesting message $M$ from $A$ (e.g., announcing a special event in which $B$ is very interested); having received $M$, node $B$ might be stimulated to initiate a direct, social interaction with node $A$.\footnote{Notice, though, that this would require node $A$ to include his/her identity in $M$, which might be at odds with the need of preserving privacy.}

IV. Defining Place-Friends

In order to define place-friends, we need to formally define a notion of individual mobility pattern, and a similarity metric between mobility patterns. Concerning definition of mobility pattern, two approaches are typically used in the literature: a point-of-interest based approach, or a partition-based approach. In the former approach, a number of points-of-interest (shopping centers, touristic attractions, public parks, etc.) are identified within the area of interest (typically, a city). A user’s mobility profile is then defined by the visiting frequency of the points-of-interest. This notion of mobility profile is used, e.g., in [12]. In the partition-based approach, the area of interest is partitioned into a number of non-overlapping regions, and a user’s mobility profile is given by the visiting frequency of each sub-region. Sub-regions typically are defined as the coverage area of a cell-tower (see, e.g., [14]), or based on a square cell partitioning (see, e.g., [1], [4], [9]).

While in principle our ideas can be applied to any definition of mobility pattern, for the sake of definiteness in the following we use a square grid partition-based approach. More specifically, we assume the mobility region $R$ is a square of side $ℓ$, which is logically partitioned into $m = h^2$ square cells of side $\frac{ℓ}{h}$, where $h$ is a tunable parameter. Assuming an arbitrary ordering of the $m$ cells, the mobility pattern of a user $A$ is defined as an $m$-dimensional vector $M_A = (x_1^A, \ldots, x_m^A)$ of real numbers $x_i^A \in [0,1]$, where $x_i^A$ denotes the relative visiting frequency for the $i$-th cell, and $\sum_i x_i^A = 1$.

Given the above definition, and in accordance with [9], a user’s mobility pattern can be represented as a vector (point) in an $m$-dimensional vectorial space. Different similarity metrics can be used to compare two mobility patterns. In [9], the authors suggest to use Euclidean distance between the two points corresponding to the individual mobility patterns. Alternatively, one can use the cosine similarity metric used in [11] to quantify similarity between user interests, where interest profiles are represented as points in a vectorial space as well. However, we have to consider that vectors representing mobility patterns are likely to be highly skewed, with most cells visited with near zero frequency, and only a few cells visited on a regular basis. This observation comes from recent studies showing that individuals tend to spend most of the time in a few locations: more specifically, the visitation frequency of locations follows a Zipf’s law with exponent 1.2 [14], corresponding to having an individual spending about 60% of the time in the 5 most popular locations. Thus, similarity metrics that consider all coordinates in the vectorial space such as Euclidean distance and cosine metric tend to shallow the relative difference/similarity between mobility patterns, due to the many close-to-zero coordinates which are present in the overwhelming majority of mobility patterns.

To get around this problem, in this paper we use a similarity metric based on comparing the $k$ cells most frequently visited by users. More specifically, let $F_A = \{i_1^A, \ldots, i_k^A\}$ and $F_B = \{i_1^B, \ldots, i_k^B\}$ be the set of most frequently visited cells of user $A$ and $B$, respectively, where $i_j^X$ denotes the ordinal number in the cell ordering of the $j$-th most frequently visited cell of user $X$. We say that users $A$ and $B$ are place-friends if and only if $|F_A \cap F_B| \geq \hat{\lambda}$, where $\hat{\lambda}$, with $1 \leq \hat{\lambda} \leq k$, is a tunable integer parameter.
representing the minimal degree of similarity needed to declare two users place-friends.

Notice that, differently from other metrics such as Euclidean distance and cosine metric, the notion of similarity defined above is apt to a scenario in which most of the $x_i$ values in a mobility profile are near-zero, since only the most frequently visited cells are accounted for in the similarity metric. Furthermore, the notion of similarity defined above is apt to a privacy-preserving implementation, using well-known secure two-party protocols for secure set intersection computation (see below).

V. THE MOBILE-CAST PROTOCOL

Participants in Mobile-Cast are users who have a GPS-equipped device, like smartphones or tablets, that can keep track of their mobility pattern during the daily life. In our protocol, we consider that the map of a zone, e.g., a city, is split into cells. Cells can assume different size, for instance they can be represented by a square where each side is long 10, 100 or 1000 meters. Clearly, there is a tradeoff between location accuracy (lower with larger cells), and memory requirements on the device (larger with smaller cells).

The current location of a user is collected at regular time intervals (e.g., a few minutes), and it is stored in a file. At regular intervals (say, every few hours or a day), all the collected data are processed to calculate the visiting frequency $f_{new}(C_i)$ for each cell $C_i$ in the map. The new frequency values are combined with the previously stored frequency value $f_{old}(C_i)$ to compute the current mobility profile of the user. We use the typical exponentially weighted moving average (EWMA) to compute the mobility profile of the user, i.e., we update the frequency value $f(C_i)$ for cell $C_i$ as follows:

$$f_{old}(C_i) = f(C_i), \quad f(C_i) = \alpha f_{new}(C_i) + (1 - \alpha) f_{old}(C_i),$$

where $0 < \alpha < 1$ is the degree of weighting decrease.

Starting from the vector $f(C_i)$ of frequency values for each cell $C_i$, our protocol builds the user mobility profile by maintaining a list of the IDs of the $k$ most visited cells, i.e., the index set $F_A$ for user $A$. Notice that, since our protocol is based on computing the set intersection between the $F$ sets, elements in $F_A$ are not ordered.

In order to provide a privacy-preserving comparison between user mobility profiles, we propose a solution based on Secure Two-party Computation functions [21]. We recall that in secure two-party computation we have two parties (Alice and Bob), each holding some private data $x$ and $y$, respectively. The goal of secure two-party function computation is allowing Alice and Bob to jointly compute the outcome of a function $g(x, y)$, without disclosing to the other party the own input. The straightforward way to solve the above problem would be to have a Trusted Third Tarty (TTP) to which Alice and Bob securely send the data, and to have the TTP compute $g(x, y)$ and separately send the outcome to Alice and Bob. The business in secure two-party computation amounts to securely compute $g(x, y)$ without the need of a TTP.

We adopt our recently developed MobileFairPlay framework [3], which is a Android-based implementation of the FairPlay framework to run secure functions [10]. FairPlay has been proven to be secure against a malicious party; in particular $i)$ a malicious party cannot learn more information about the other party’s input than it can learn from a TTP that computes the function; and $ii)$ a malicious party cannot change the output of the computed function [10]. Notice that, as customary in secure two-party computation, there is an asymmetry on the provided security guarantees: in particular, there is no way to prevent Alice from terminating the protocol prematurely, and not sending the outcome of the computation to Bob. This situation can be detected by Bob, but cannot be recovered from.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{image}
\caption{Protocol flow to discover similarity of mobility profiles.}
\end{figure}

In Fig. 1 we pictorially present our protocol, which makes use of Secure-two party computation to compare Alice and Bob mobility profiles. The protocol assume a threshold value $\lambda \leq \hat{\lambda}$, known to all participants, is used to control the message forwarding process. The protocols starts when Alice and Bob are in close proximity: for instance, using a Bluetooth connection, when they are less than 20m apart. Initially, Alice starts a connection to Bob; once received the connection request from Alice, Bob starts the secure computation of the function:

$$g(F_A, F_B) = \begin{cases} 
\text{True} & \text{if } |F_A \cap F_B| \geq \lambda \\
\text{False} & \text{otherwise}
\end{cases} \quad (1)$$

If the profiles are found to be similar, Alice and Bob are estimated as place-friends, and they start comparing the content of their buffers and exchange files. Otherwise, the connection is terminated. Notice that, if $\lambda < \hat{\lambda}$, function $g(F_A, F_B)$ might evaluate at True even if Alice and Bob are not place-friends. This situation can be avoided by setting $\lambda = \hat{\lambda}$. However, as we shall see in the next section, increasing the value of $\lambda$ is detrimental for privacy preservation since more information about the own mobility pattern is disclosed.
to the other party during the computation. For this reason, using a value of \( \lambda \) lower than \( \lambda \) is often preferable in practice.

The message forwarding policy is as follows. If Alice is the sender of a message \( M \), or if Alice received the message directly from the sender, \( M \) is delivered to Bob if Alice and Bob are estimated to be place-friends according to function (1). In all other cases, including the case in which Alice and Bob are estimated to be place-friends but only \( M \) is not delivered to Bob. Notice that this forwarding policy, which can easily be implemented including an hop-count field in the message, ensures that any message travels at most two-hops to reach a place-friend. For this reason, we name our protocol 2-hops mobility-cast (2H for short).

Restricting message forwarding to two hops finds its motivation in the fact that Alice and Bob are not necessarily place-friend, but only estimated to be so (recall discussion above). Thus, extending forwarding beyond the second hop would likely reduce the precision of the forwarding protocol in delivering messages exclusively to place-friends of the sender. A similar design choice has been done, e.g., in [2].

VI. PRIVACY ANALYSIS

While not disclosing user mobility profiles, a certain leakage of private information is unavoidable when using secure two-party computation. In particular, at the end of the protocol computation, the following information is leaked to the other party:

- if the outcome of \( g(F^A, F^B) \) is True, the party (say, Bob) knows that at least \( \lambda \) of his most popular locations are in common with Alice. However, he does not know the exact number of common locations (can be any number in the \([\lambda, k]\) interval), nor which they are exactly. Only in the case that \( \lambda = k \) Bob knows that Alice has the same exact mobility profile as the own profile.

- If the outcome of \( g(F^A, F^B) \) is False, Bob knows only that less than \( \lambda \) of his most popular locations are in common with Alice. However, he does not know the exact number of common locations (can be any number in the \([0, \lambda - 1]\) interval), nor which they are exactly.

To quantitatively evaluate privacy leakage, we use the entropy-based privacy preservation metric introduced in [2]. In particular, we want to quantify the privacy leakage caused by the protocol execution, under the assumption that the attacker’s goal is discovering the other party’s mobility profile, i.e., his/her \( k \) most frequent locations. Taking w.l.o.g. Alice’s perspective, Bob’s profile is a set of \( k \) cell IDs, which can be modeled as a random variable \( Y = (y_1, \ldots, y_k) \). Each specific realization of r.v. \( Y \) is denoted \( Y_i \), and corresponds to a set of \( k \) cell IDs chosen amongst the \( m \) possible cell IDs. Hence, the number of possible values of r.v. \( Y \) is \( \binom{m}{k} \).

The bit entropy of a random variable \( Y \) with possible values \( \{y_1, \ldots, y_n\} \) is defined as [13]:

\[
H[Y] = -\sum_{i=1}^{n} p(Y_i) \log_2 p(Y_i),
\]

where \( p(y) \) is the probability mass function of random variable \( Y \). The privacy preservation metric of a certain protocol \( P \) is defined as

\[
pp(P) = \frac{H[Y_{after}]}{H[Y_{initial}]},
\]

where \( Y_{initial} \) and \( Y_{after} \) are the r.v. modeling Alice’s uncertainty about Bob’s profile initially and after the execution of protocol \( P \), respectively. The \( pp \) metric takes values in \([0, 1]\), with 0 indicating that after \( P \)’s execution Alice knows exactly Bob’s mobility profile (zero privacy preservation), and 1 indicating that after \( P \)’s execution Alice has the same knowledge about Bob’s profile he had before executing the protocol (maximal privacy preservation).

To quantify the \( pp \) metric, we need to make same assumptions about the distribution of r.v. \( Y \). In the following, we quantify privacy leakage under the assumption that all locations have the same probability of being included in a node’s mobility profile. In other words, we assume that all \( \binom{m}{k} \) possible subsets of \( k \) out of \( m \) possible cell IDs are equiprobable. Notice that this assumption is not necessarily in contrast with the observation made in [14] that people tend to frequently visit only a few locations. In fact, people in general have different more frequently visited location, and the resulting aggregate location popularity (which is the one that determines the distribution of r.v. \( Y \)) might be relatively uniform. On the other hand, analyzing privacy leakage under a non-uniform location popularity assumption (e.g., assuming Zipf’s law) is cumbersome, due to the need of computing each single \( p(Y_i) \) value in the definition of bit-entropy. This further justifies our working assumption of uniform location popularity.

Let us first compute \( H[Y_{initial}] \). If locations (cell IDs) have uniform popularity, from Alice’s perspective any of the \( \binom{m}{k} \) possible Bob’s mobility profiles has the same probability \( \frac{1}{\binom{m}{k}} \) of occurrence. Hence, we get:

\[
H[Y_{initial}] = -\sum_{i=1}^{\binom{m}{k}} p(Y_i) \log_2 p(Y_i) = -\sum_{i=1}^{\binom{m}{k}} \frac{1}{\binom{m}{k}} \log_2 \frac{1}{\binom{m}{k}} = \binom{m}{k} \log_2 \binom{m}{k}.
\]

Let us now compute \( H[Y_{after}] \). We recall that the value of \( \lambda \) used to estimate place-friendship is fixed and known to both parties. We distinguish the case of protocol execution with outcome True or False.

If the outcome is True, after the protocol execution Alice knows that Bob’s mobility profile has at least \( \lambda \leq k \) locations in common with the own profile. Fixed a value \( h \), with \( h \leq k \), of possible common locations, the number of possible choices for Bob’s profile with \( h \) locations in common with Alice can be computed as follows:

\[
\binom{k}{h} \cdot \binom{m-k}{k-h}.
\]
In fact, the first binomial coefficient accounts for all possible choices of the $h$ locations in common with Alice’s profile, taken amongst the $k$ locations in Alice’s profile. The second binomial coefficient accounts for all possible choices of the remaining $k-h$ locations in Bob’s profile, which are taken amongst the $m-k$ locations which are not in common with Alice’s profile.

Given the above, we can compute $H[Y_{after}]$ in case of True outcome as follows:

$$H[Y_{after}^T] = \log_2 \left( \sum_{h=\lambda}^{k} \binom{k}{h} \cdot \binom{m-k}{k-h} \right).$$

If the outcome is False, Alice knows that all possible Bob’s profiles with at least $\lambda$ locations in common with the own profile should be excluded from the universe of possible profiles, i.e.,

$$H[Y_{after}^F] = \log_2 \left( \binom{m}{k} - \sum_{h=\lambda}^{k} \binom{k}{h} \cdot \binom{m-k}{k-h} \right).$$

The value of the $pp$ metric for increasing values of $k$ and $\lambda = 3$ is reported in Figure 2. As expected, a True outcome of the protocol’s execution discloses more information to the adversary. However, the amount of information disclosed to the other party can be reduced by increasing the value of the number $k$ of locations in the mobility profile. Notice, though, that increasing the value of $k$ beyond a reasonable value has a negative effect on the accuracy of the mobile-cast operation, indicating a tradeoff between networking performance and privacy already observed in [2] for the case of interest-cast.

Figure 3 reports the $pp$ metric for increasing values of $\lambda$, with parameter $k$ fixed to 10. In case of True protocol outcome (the most critical case for privacy leakage), we can reduce privacy leakage by reducing the value of $\lambda$. This dataset recorded a broad range of users’ outdoor movements, including not only life routines like go home and go to work, but also some entertainments and sports activities, such as shopping, sightseeing, dining, hiking, and cycling. The vast majority of the data was collected in Beijing, China.

Since the original dataset is huge and covers a very long time interval, we decided to consider only a time interval of one year. More specifically, we used the trajectories saved in 2008, which is the denser year in terms of total number of active users in the trace (77).

A. Data pre-processing

In order to derive user mobility profiles, we have divided the central area of the city of Beijing into square cells. Each cell is $1 \text{Km}$ wide, and in total we consider an area of 340 $\text{Km}^2$, corresponding to 340 cells.

VII. EXPERIMENTAL EVALUATION

In order to estimate the performance of $2H$, we have used the real-world mobility trace available through the Microsoft Research GeoLife GPS Trajectories project [17][18][19]. The dataset provides GPS trajectories of 182 people collected in a period of over five years (from April 2007 to August 2012). A GPS trajectory is represented by a sequence of time-stamped points, each of which contains the information of latitude, longitude and altitude. These trajectories were recorded by different GPS loggers and GPS-phones. Positions were stored every 1 to 5 seconds or every 5 to 10 meters per point. This dataset recorded a broad range of users’ outdoor movements, including not only life routines like go home and go to work, but also some entertainments and sports activities, such as shopping, sightseeing, dining, hiking, and cycling. The vast majority of the data was collected in Beijing, China.

Since the original dataset is huge and covers a very long time interval, we decided to consider only a time interval of one year. More specifically, we used the trajectories saved in 2008, which is the denser year in terms of total number of active users in the trace (77).
Preliminary to the experiments, we calculated the frequency with which each user visits the 340 cells using the entire, one-year long data trace. We then recoded for each user her 10 most visited cells, which forms her mobility profile. Then, in order to set the value of $\lambda$ and $\lambda$ to reasonable values, we have computed, for each user in the dataset, how many cells on average she has in common with each of the other 76 users. The results of this preliminary evaluation showed that, on the average, a user has 1.85 cells in common with another user, but the variance is high (2.39). This indicates that mobility profiles have very different degrees of similarity between themselves, which is a good scenario to evaluate the ability of protocol $2H$ to deliver messages precisely to place-friends.

B. Simulation experiments

We implemented a Java simulator that uses the Beijing data trace to estimate mobility-cast performance. During a simulation experiment, a user $U$ is selected as the message sender. For the given user $U$, the simulator identifies within the year-long trace a two-months portion with the largest density of encounters. The message $M$ is then generated at node $U$ at the beginning of the identified two-months stretch of the trace, and the forwarding process starts according to one of the forwarding policies described in the next section. At the end of the experiment, the ID of users that received $M$ is recorded, as well as the ID of $U$’s place-friends (computed using the mobility profiles). For each timestamp in the dataset trajectory, we consider all possible pairs of generic users Alice and Bob. If they are within a distance of 1000 m, Alice checks whether she has in her queue packets that she may forward to Bob according to the chosen forwarding policy. If this is the case, depending on whether place-friendship is required by the forwarding policy, Alice starts comparing the number of cells in common with Bob’s mobility profile, and, in case this number is larger than a threshold $\lambda$, message forwarding takes place. Notice that the relatively large value of the transmission range of $1Km$ has been chosen to cope with the very low density of users per unit area in the data trace.

C. Forwarding protocols

In the simulations, $2H$ performance is compared against that of the following forwarding protocols:

- **DD**: strictly speaking, DD is not a forwarding protocol, since no packet forwarding takes place. In DD, it is only the sender (say, Alice) of a message $M$ who can deliver it to other users. In particular, Alice delivers $M$ to another user (say, Bob) subject to the condition that Alice and Bob have at least $\lambda$ common cells in their mobility profile. Since threshold $\lambda$ is used (instead of threshold $\lambda$ as in protocol $2H$), only actual place-friends of Alice can receive $M$ under protocol DD.

- **EP**: it is the classical Epidemic forwarding protocol [20]. Whenever Alice, who has a copy of $M$, meets another user (say, Bob) who does not hold $M$, she forwards a copy of $M$ to Bob. Forwarding occurs independently of whether Alice and Bob are place-friends.

- **PF**: it is a 2-hops probabilistic forwarding protocol. Whenever the sender of message $M$ (say, Alice) encounters another user (say, Bob), she forwards a copy of $M$ to Bob with fixed probability $p$. Similarly to EP, forwarding occurs independently of whether Alice and Bob are place-friends. The same forwarding rule is used by any user who received $M$ directly from Alice. No message forwarding is allowed beyond the second hop of communication.

D. Results

The results reported in this section are computed averaging across 77 simulation experiments, one for each user in the data trace. The following performance metrics have been estimated:

- **Coverage**: it is the ratio between the number of place-friends of the sender user $U$ that received the message $M$, and the total number of $U$’s place-friends. Notice that coverage is computed using threshold $\lambda$ to define place-friends. The coverage metric measures how good a protocol is at delivering messages to $U$’s place-friends.

- **Precision**: it is the ratio between the number of $U$’s place-friends that received $M$, and the total number of users that received $M$. Precision measures the accuracy of a protocol in delivering messages only to $U$’s place friends.

- **Cost**: it is the total number of copies of $M$ circulating in the network at the end of the protocol execution. Cost measures the communication overhead of a protocol.

Figure 5 reports the coverage experienced by the various forwarding protocols as a function of $\lambda$.

![Coverage vs. lambda](image-url)
As expected, EP has the best coverage performance, due to the epidemic forwarding process. However, EP pays the fee in terms of cost, which as much as 2.3 times higher than that of 2H – see Figure 6. EP performs poorly also in terms of precision, due to the fact that the forwarding process is oblivious to place-friendship – see Figure 7. DD is at the other end of the scale with respect to EP: it has poor coverage performance but minimum cost due to the lack of forwarding, and it has optimal precision due to the fact that M is delivered only to place-friends of the sender node U.

**Table I**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>( \lambda )</th>
<th>Var (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2H</td>
<td>1</td>
<td>5.3</td>
</tr>
<tr>
<td>2H</td>
<td>2</td>
<td>5.3</td>
</tr>
<tr>
<td>2H</td>
<td>3</td>
<td>5.4</td>
</tr>
<tr>
<td>PF</td>
<td>1</td>
<td>7.7</td>
</tr>
<tr>
<td>PF</td>
<td>2</td>
<td>8.2</td>
</tr>
<tr>
<td>PF</td>
<td>3</td>
<td>9.4</td>
</tr>
</tbody>
</table>

VARIANCE IN COVERAGE PERFORMANCE OF PROTOCOLS 2H AND PF.

 Protocols 2H and PF strike a compromise between coverage and cost, improving coverage performance considerably with respect to DD, while considerably reducing the cost with respect to EP. It is interesting to compare 2H and PF performance. While the two protocols display comparable coverage performance, 2H achieves a much better precision than PF – see Figure 7. This is due to the fact that, while the average coverage performance of the two protocols is very similar, the variance in the achieved coverage is very different for the two protocols – see Table I. The variability in coverage performance displayed by randomized forwarding explains the much better performance displayed by 2H with respect to PF in terms of precision.

Summarizing, we can conclude that 2H proved effective in delivering messages only to place-friends, striking the best compromise between coverage, precision, and cost, amongst the considered protocols. The optimal setting of the parameter \( \lambda \) in protocol 2H is a design choice that should account not only for the performance metrics considered herein, but also for the privacy preservation properties of the protocol as analyzed in Section VI.

**VIII. Prototype Implementation of 2H**

In order to verify whether protocol 2H can be efficiently executed with current smartphone technology, we have implemented the most computationally intensive task of the protocol on the Android platform. More specifically, we have implemented the secure computation of function \( g(F^A, F^B) \) as defined in equation (1). Function \( g(F^A, F^B) \) secure computation is implemented in MobileFairPlay [3], an Android-based implementation of FairPlay [10]. FairPlay functions must be written with the language Secure Function Definition Language (SFDL), which is a high level language that allows developers to write simple function that are then converted into garbled boolean circuits. Only a limited number of commands and operations are available in SFDL. For instance, it is not possible to use text values in a function, but only integers or simple types are allowed. MobileFairPlay then transforms an SFDL program into a Java program executable on Android smartphones.

The SFDL function that we have written to securely compute similarity between Alice’s and Bob’s mobility profiles, which is reported in the Appendix, is quite simple and works by comparing each cell ID in Alice’s profile with those in Bob’s profile. If the same cell ID appears in both profiles, then a counter is increased. At the end of the function, the value of the counter (common frequently visited cells) is compared to the threshold \( \lambda \) known to both parties. If the comparison is positive, then the output for both Alice and Bob is True (represented by integer 1), otherwise it is False (represented by integer 0).

The APP that we have built is a prototype of the 2H protocol including secure estimation of place-friendship. User mobility is traced every minute. Once per day, the application takes all saved GPS coordinates and calculates frequencies per cells. When two users Alice and Bob meet each other, Alice starts challenging Bob on the number of common cells that they have in their profiles in a privacy-preserving manner. The current version of the APP considers mobility profiles composed of the five most frequently visited cells. Once Alice starts the connection with Bob through the Bluetooth interface, the devices are automatically paired for the first time. Then, the Secure-Two party computation of function \( g(F^A, F^B) \) begins. At the end of the computation, both users know the value of \( g(F^A, F^B) \). If Alice and Bob recognize each other as place-friends, they start an interaction phase consisting in sharing files (txt, pdf, jpg, etc.).

To evaluate the computational time of the application, we
have executed the APP on five smartphones, which are listed in Table II together with their technical specifications.

Fig. 8 reports the time needed to compute \( g(F^A, F^B) \), including communication through the Bluetooth interface. In the reported results, the Samsung Galaxy S2 always played Alice’s role, while the Bob’s role is played by the other smartphones. We can see that running times range between less than 10 and 12.5 seconds, with the best running time achieved by the fastest smartphone, i.e. Sony Xperia T. We can then conclude that our proposed 2H protocol can be effectively executed also with current smartphone technology.

![Graph showing time needed to compute function \( g(F^A, F^B) \).](image)

**Fig. 8.** Time needed to compute function \( g(F^A, F^B) \).

### IX. Conclusions

In this paper, we have put forward the idea of delivering messages to place-friends in opportunistic networks. Delivering messages to place-friends (which we called mobility-cast) has the potential to reach not only current social contacts of a user, but also forthcoming social contacts as observed in recent studies. We have introduced a privacy-preserving design of a mobility-cast protocol called 2H, and analyzed its privacy-preservation properties, as well as evaluated its performance through experiments based on a real-world mobility trace. The results of the experiments have shown that 2H strikes the best balance between coverage, precision, and cost, amongst the considered protocols. Finally, we have shown that secure place-friendship estimation, which is at the core of 2H, can be effectively implemented with current smartphone technology, experiencing running times below 10 seconds.

As directions for future work, we mention extending the experimental evaluation using more data traces to better quantify 2H benefits, and realizing a full-fledged mobility-cast APP to be tested with real users.

### REFERENCES


program MobileCast
{
    const Size = 5;

type Data = Int<10>;
type AliceInput = Data[Size];
type BobInput = Data[Size+1]; // one more cell for the threshold
type AliceOutput = Boolean;
type BobOutput = Boolean;

type Output = struct {AliceOutput alice, BobOutput bob};
type Input = struct {AliceInput alice, BobInput bob};

function Output output(Input input)
{
    var Data i, k, index, soglia;
    threshold = input.bob[5]; // It gets the threshold
    index=0;

    for (i = 0 to Size-1)
    {
        for (k = 0 to Size-1)
        {
            if (input.bob[i] == input.alice[k])
            {
                index = index+1;
            }
        }
    }

    if (index >= threshold)
    {
        output.alice = 1;
        output.bob = 1;
    }
    else
    {
        output.alice = 0;
        output.bob = 0;
    }
}
}

Listing 1. Secure estimation of place-friendship written in SFDL