“Once Upon a Place”: Compute Your Meeting Location Privately

Technical Report EPFL-REPORT-163503

Igor Bilogrevic*, Murtuza Jadliwala*, Kübra Kalkan†, Jean-Pierre Hubaux* and Imad Aad‡
*Laboratory for Communications and Applications 1
EPFL, Switzerland
firstname.lastname@epfl.ch
†Faculty of Engineering and Natural Sciences
Sabanci University, Istanbul, Turkey
kubrakalkan@su.sabanciuniv.edu
‡Nokia Research Center
Lausanne, Switzerland
imad.aad@nokia.com

***Abstract*—Popular services such as Doodle Mobile and Tymelie are extremely useful planning tools that enable mobile-phone users to determine common meeting time(s) for events. Similar planning tools for determining optimal meeting locations, based on the location preferences of the users, are highly desirable for event planning and management in popular mobile phone applications, such as taxi sharing, route planning and mobile participatory sensing. Yet, they have received very little attention by researchers. An important, and often overlooked, facet of such planning applications is the privacy of the participating users and their preferences; users want to agree on a meeting location without necessarily revealing their location preferences to the service provider or to the other users. In this paper, we address the problem of privacy-preserving optimal meeting-location computation, especially focusing on its applicability to current mobile devices and applications. We first define the notion of privacy in such computations. Second, we model the problem of optimal meeting-location computation as a *privacy-preserving k-center* problem and we design two solutions; both solutions take advantage of the homomorphic properties of well-known cryptosystems by Boneh-Goh-Nissim, ElGamal and Paillier in order to perform oblivious computations. Third, we implement the proposed solutions on a testbed of the latest generation Nokia mobile devices and study their performance. Finally, we assess the utility and expectations, in terms of privacy and usability, of the proposed solutions by means of a targeted survey and user-study of mobile-phone users.**

I. Introduction

Technology used in mobile devices has come a long way since the introduction of GSM, almost two decades ago. Wide-area broadband connectivity and miniaturization enable lightweight smartphones to be used increasingly as personal computers, which makes them a critical tool for improving the efficiency of service-oriented businesses. A recent survey shows that 37% of IT organizations are planning to significantly increase their spendings for mobile devices and their support, and almost 48% of the IT companies recognize that mobile devices and applications are critical for their business processes [1].

One such important application/service for mobile devices is activity management. For example, activity management applications such as Microsoft Outlook [2], Apple iCal [3] and Doodle [4], are becoming increasingly popular [5]. Colleagues can use these applications on their mobile devices to organize business meetings, groups of friends can organize parties on weekends and people unbeknownst to each other can get dating recommendations based on their common time availabilities. Calendar sharing and automatic meeting scheduling, based on common time availabilities, are the two main features of such applications and services.

Although current activity management applications focus primarily on time-related scheduling, little attention has been devoted to the spatial constraints. Determining a suitable *location* of a meeting or activity is as important as identifying the commonly acceptable time slot. Several providers already offer variants of this service (such as finding the location that is in the middle between two user-defined locations) either as on-line web applications ([6], [7], [8]) or as stand-alone applications for mobile devices [9]. Not only is such a feature desirable, but it also increases both the efficiency and the productivity of all the involved parties. For instance, companies could set cost-minimizing travel policies, such that meeting locations are optimized with respect to travel time or distance for each participant, and colleagues would not need to manually search and unanimously agree for a suitable meeting location. Moreover, preferences of companies and employees would be taken into consideration, and the overall meeting costs would be optimized.

One important aspect of such a service is the privacy of users’ meeting location preferences. Let us explain this by means of a practical example. Assume representatives of three competing companies X, Y and Z are exploring new regions for profitable business opportunities. If these representatives of X, Y and Z would need to meet, it is very likely that none of them would like to reveal their strategic interest in a position. In other words, they would like to conceal the association between the meeting participants and their preferred meeting locations. Yet, they would like to find a place that is easy to reach for all of them. In such a scenario, determining the optimal meeting place while preserving the privacy of individuals' preferred locations is of utmost importance. For instance, a global telecom operator recently announced a taxi-sharing application [10], where the users have to reveal their departure and destination points to the server. If such a server is not fully trusted by the users, revealing sensitive locations (such as users home/work addresses) could pave the way for financial and social inference attacks by third-parties.

Popular meeting scheduling services, such as Doodle [4] or Outlook [2], require each participant to indicate his time availabilities in order to find a common time slot. Clearly, in such services, third-party providers and other participants can find out the private schedule information of other participants. If providers offered to find the optimal meeting location in a similar fashion, they would need to know each participant's personal location preferences, which would pose an even greater privacy risk. For instance, if Doodle users were asked to introduce their preferred location in addition to their time availabilities, it would be even easier to abusively track them and profile their habits and mobility patterns. As an example, a popular service offering real street images has been used in a divorce case [11] by a spouse against her husband (after he denied being where he was filmed). Undoubtedly, the privacy of users' personal information is a very sensitive issue, and people are increasingly worried about data disclosure risks and their negative social and financial consequences.

Although there is a strong correlation between time and space dimensions (e.g., a person's location is closely correlated with the time of the day), transposing the mechanisms developed for time scheduling to location determination is very challenging. In the scheduling scenario, for instance, users specify whether they are available or not for each known time slot, and the common availabilities are then computed by aggregating the individual schedules. The common availabilities are then obtained simply by checking whether all users are available in a given time slot. Determining the optimal meeting location, on the contrary, is not as straightforward. For instance, in the space domain users can choose the precision of their coordinates, depending on their level of anonymity, by providing only an approximate region or a highly populated public space. Moreover, a meeting location might not be one of the locations that each user proposes but, on the contrary, it might be a different location that is easily reachable by all participants, and therefore simply aggregating the individual locations might not result in an optimal meeting place. Clearly, as the approach of the time-scheduling problem is fundamentally different from the optimal meeting-location problem, solutions to the former cannot be directly applied to the latter. Additionally, the privacy requirements of hiding the location preferences from the computing party and the other participants make this problem even more challenging.

Our work: In this work, we address the problem of finding the optimal meeting location in a way that preserves the privacy of users' personal location preferences *vis-à-vis* other users and any third-parties that might be involved. To the best of our knowledge, this is the first paper that presents solutions to the privacy-preserving optimal meeting-location problem. Moreover, the design of such solutions is particularly targeted at mobile devices, with limited computational and communication capabilities. The proposed application is independent of any underlying service or network provider, and it can be included in an existing online meeting management services as an add-on feature.

Previous research on related problems, albeit without the privacy considerations, has been carried out mostly in the operations research domain, where the challenge is to find an optimal location (for a warehouse, an ATM machine or a power plant) that minimizes a given cost function (longest distance, shortest time or best connectivity). The purpose of such solutions is purely an optimization goal, without any requirements on the privacy front, mostly because solutions are found in-house, without involving a potentially untrusted third-party or external partners. However, when considering independent users with private information, such solutions based on revealing location preferences to a central entity are no longer applicable or desired. Untrusted infrastructure and potentially competing partners represent two crucial distinguishing factors with respect to the pure optimization techniques studied in operation research. Privacy-preserving schemes, although only for *time* scheduling, have received significant attention from the research community, and several algorithms that preserve the privacy of individual availabilities have been developed. Protocols based on secure multiparty computation [12] and distributed constraint satisfaction problems ([13], [14]) have been designed and evaluated with respect to the privacy guarantees they offer. Nevertheless, as time scheduling and location determination are two fundamentally different problems, no such work seems to exist for the privacy-preserving meeting *location* problem.

Contributions: In this paper, we design the first privacy-preserving, optimal meeting-location service for mobile devices. The main contributions of this work are:

- Two novel algorithms for the privacy-preserving optimal meeting-location (POML) problem
- A formal framework for representing privacy requirements in the POML problem and analysis of the privacy provided by the proposed algorithms
- Implementation of the proposed algorithms on mobile devices, including the first known implementation of the Boneh-Goh-Nissim encryption scheme [15] used in one of the algorithms.
- Extensive performance measurements of the proposed solution on a test-bed of commercial mobile devices, as well as a targeted user-study on mobile-phone users.

The rest of the paper is organized as follows. In Section II we describe the system architecture and threat model, whereas in Section III we define the privacy-preserving meeting-location problem and the privacy requirements. In Section IV we present and analyze our solution, and in Section V we present its implementation on mobile devices and the performance results. In Section VI we discuss the security and optimization extensions to our solutions, and in Section VII we present the results of our user-study. In Section VIII we discuss the related work and we conclude the paper in Section IX.

II. System Architecture

In this section, we outline the network and threat models that are considered in the paper.

A. System Model

Our system is composed of two main entities: (i) a set of users[1] (or mobile devices) $\mathbb{U} = \{u_1, \ldots, u_N\}$ and (ii) a third-party service provider, called *Location Determination Server (LDS)*. The N users want to determine the optimal meeting location that is computed by the LDS.

Each user's mobile device is assumed to be able to establish communication with the LDS either directly or through a fixed infrastructure, such as the Internet. The mobile devices are able to perform public-key cryptographic operations, such as encryption, decryption and digital signature. Moreover, we assume that each user $u_i \in \mathbb{U}$ has means of determining the position $L_i = (x_i, y_i) \in \mathbb{N}^2$ of his preferred meeting location (or his own location) by using a common coordinate system. We consider a two-dimensional position coordinates system, but the proposed schemes are general enough and can be easily extended to three dimensions. These coordinates can be mapped from/to real geographic coordinates, either locally (using a GPS receiver on his mobile device) or through an optional third-party positioning service (PS), such as Google My Location [16]. For instance, such definition of L_i can be made fully compliant with the UTM coordinate system [17], which is a plane coordinate system where points are represented as a 2-tuple of positive values (distances in meters from a given reference point). As UTM divides the globe in grid zones, our solution applies best to such regional scenarios.

We define the set of the preferred meeting locations of all users as $\mathbb{L} = \{L_i\}_{i=1}^{N}$. For the sake of simplicity, we assume a flat-Earth model and we consider line-of-sight Euclidian distances between preferred meeting locations. Even though the actual real-world distance (road, railway, boat, etc.) between two locations is at least as large as their Euclidian distance, the proportion between distances in the real world is assumed to be correlated with the proportion of the respective Euclidian distances.

We assume that each of the N users has his own public/private key pair $(K_P^{u_i}, K_s^{u_i})$, certified by a trusted CA, which is used to digitally sign the messages that are sent to the LDS. Moreover, we assume that the N users share a common secret that is utilized to generate a shared public/private key pair $(K_P^{M_v}, K_s^{M_v})$ in an online fashion for each meeting setup instance v. The private key $K_s^{M_v}$ generated in this way is known only to all meeting participants, whereas the public key $K_P^{M_v}$ is known to everyone including the LDS. This could be achieved through a secure credential establishment protocol such as in ([18], [19], [20]).

The LDS executes the optimal meeting-location algorithm on the inputs it receives by the users in order to compute the optimal meeting location. The LDS is also able to perform public-key cryptographic functions. For instance, a common public-key infrastructure using the RSA cryptosystem [21] could be employed. Let K_P^{LDS} be the public key, certified by a trusted CA, and K_s^{LDS} the corresponding private key of the LDS. K_P^{LDS} is publicly known and users encrypt their input to the meeting-location algorithm using this key; the encrypted input can be decrypted by the LDS using its private key K_s^{LDS}. This ensures message confidentiality and integrity for all the messages exchanged between users and the LDS. For simplicity of exposition, in our protocols we do not explicitly show these cryptographic operations involving LDS's public/private key.

B. Threat Model

a) Location Determination Server: The LDS is the entity that performs the computations in order to determine the optimal meeting location for the users. The LDS is assumed to execute the algorithms correctly, i.e., take all the inputs and produce the output according to the algorithm. However, the LDS may try to learn information about user meeting-location preferences from the received inputs, the intermediate results and the produced outputs of the meeting-location computations. This type of adversarial behavior is usually referred to as *honest-but-curious* adversary (or semi-honest) [22]. In most practical settings, where service providers have a commercial interest in providing a faithful service to their customers, the assumption of a semi-honest LDS is generally sufficient.

b) Users: The participating users also want to learn the private location preferences of other users from the output of the algorithm they receive from the LDS. We refer to such attacks as passive attacks. As user inputs are encrypted with the LDS's public key K_P^{LDS}, there is a confidentiality guarantee against basic eavesdropping by participants and non participants. In addition to these passive attacks, participating users may also attempt to actively attack the protocol by, for instance, colluding with other users or manipulating their own inputs to learn the output.

A complete list of symbols can be found in Table I.

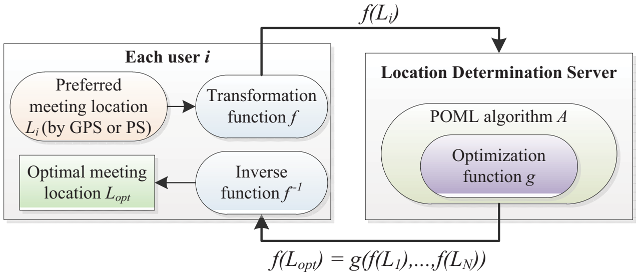
III. Problem and Privacy Definitions

In this section, we define the privacy-preserving optimal meeting-location (*POML*) problem and the necessary privacy requirements that have to be met by any algorithm that solves the POML problem.

[1] Throughout this paper, we use the words *users* and *devices* interchangeably. The meaning is clear from the context, unless stated otherwise.

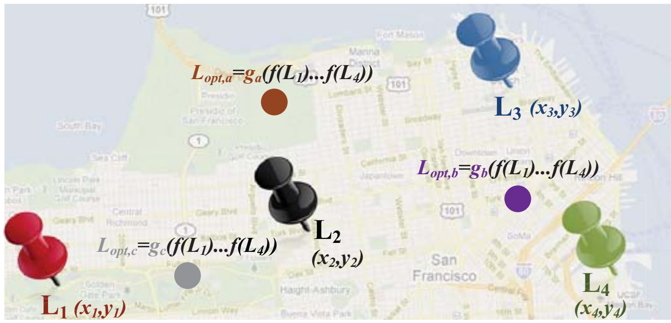
Table I
TABLE OF SYMBOLS.

SYMBOL	DEFINITION
$\mathbf{Adv}^{\mathbf{IDT}}$	Identifiability advantage
$\mathbf{Adv}^{\mathbf{d\text{-}LNK}}$	Distance-linkability advantage
$\mathbf{Adv}^{\mathbf{c\text{-}LNK}}$	Coordinate-linkability advantage
d_{ij}	Euclidian distance between two points in the plane
D_i^M	Maximum Euclidian distance of user i to any other user $j \neq i$
f	Public transformation function based on secret key (for privacy)
g	Optimization function
LDS	Location Determination Server
L_i	Desired meeting location of user i, $L_i = (x_i, y_i)$
L_{opt}	Optimal meeting location
PS	Positioning Service
u_a	Attacker (a user participating in the POML protocol)
$E(.)$	Encryption of (.) (the encryption scheme is clear from the context)
$ElG(.)$/ $Pai(.)$	Encryption of (.) using the ElGamal/Paillier encryption scheme
σ, θ	Element-permutation functions



f(L_i)
Each user i
Preferred meeting location L_i (by GPS or PS)
Transformation function f
Optimal meeting location L_opt
Inverse function f^-1
Location Determination Server
POML algorithm A
Optimization function g
f(L_opt) = g(f(L_1),...,f(L_N))

Figure 1. Functional diagram of the POML protocol, where the POML algorithm is executed by an LDS.



L_opt,a=g_a(f(L_1)...f(L_4))
L_3 (x_3,y_3)
L_opt,b=g_b(f(L_1)...f(L_4))
L_2 (x_2,y_2)
L_opt,c=g_c(f(L_1)...f(L_4))
L_1 (x_1,y_1)
L_4 (x_4,y_4)
San Francisco

Figure 2. General POML scenario, where three distinct optimization functions g_a, g_b, g_c output three different optimal meeting locations $L_{opt,a}, L_{opt,b}, L_{opt,c}$, given the user-preferred meeting locations $L_1, \ldots, L_4$.

A. Privacy-Preserving Optimal Meeting-Location Problem

In this work, we consider the problem of finding, in a privacy-preserving way, the optimal meeting location among several participants, such that (i) each of the users gets to know only the final optimal location and (ii) no user or third-party server knows any other private location information about any user involved in the computations. We refer to an algorithm that solves such problem as *privacy-preserving optimal meeting-location (POML)* algorithm. In general, any POML algorithm A should work as follows. Given a set of N users $\mathbb{U}$, where each user u_i, $i \in \{1, \ldots, N\}$, has a private preferred location L_i, and a transformation function f, the POML algorithm takes $f(L_i)$, $\forall i \in \{1, \ldots, N\}$, as inputs and computes the location function $f(L_{opt})$ as output, where L_{opt} is the optimal location as computed by an optimization function g, given the private inputs $f(L_1), \ldots, f(L_N)$. Moreover, the inputs to the algorithm and the outputs it produces should satisfy the two privacy requirements described earlier. Figure 1 shows a functional diagram of a POML protocol, where the POML algorithm A is executed by an LDS.

Formally, a POML algorithm A works as follows:

- *Input*: a transformation f of private locations L_i

  $$f(L_1)||f(L_2)||\ldots||f(L_N)$$

  where f is a one-way public function (based on secret key) such that it is hard (success with only a negligible probability) to determine the input L_i without knowing the secret key, by just observing $f(L_i)$.
- *Output*: an output $f(L_{opt}) = g(f(L_1), \ldots, f(L_N))$, where g is an optimization function and $L_{opt} = (x_l, y_l) \in \mathbb{N}^2$ is the optimal meeting location that has been selected for this particular set of users, such that it is hard for the LDS to determine L_{opt} by just observing $f(L_{opt})$. Given $f(L_{opt})$, each user is able to compute $L_{opt} = f^{-1}(f(L_{opt}))$ using his local data.

The optimization function g can be defined in several ways, depending on the preferences of the users, their employers or policies. For instance, users might prefer to meet in locations that are close to their offices, and their employers might prefer a place that is closest to their clients. Figure 2 shows three different optimal locations for three distinct optimization functions g. In Section IV-A we describe one such function that we use throughout this paper, and in Section VI-B we discuss other optimization functions that can be used in any POML algorithm.

B. Privacy Requirements and Definitions

The generic POML protocol represented in Figure 1 involves several operations, some of which are executed on the user device and some on a third-party LDS. Moreover, the PS is optionally required by the users in order to obtain the location coordinates of POIs in a given area. In order to guarantee that private information about users is not leaked to other users or third-parties during the execution of the POML algorithm, we need to formally define requirements that any such algorithm has to satisfy. Afterwards, we will evaluate the proposed POML algorithms based on these privacy definitions. Informally, the privacy requirements can be stated as follows. After the execution of the POML algorithm, any user u_i should not be able to infer (i) the preferred location L_j of any other user $u_j \neq u_i$ nor (ii) the relative distances d_{ij} between any two users $u_i \neq u_j$. Likewise, any LDS (and PS) should not be able to infer (iii) the preferred location L_i of any user u_i, (iv) the relative distance d_{ij} between any two users $u_i \neq u_j$ nor (v) the final meeting location L_{opt}. Such privacy requirements can be grouped in two components, called as *user-privacy* and *server-privacy*, defined as follows.

1) **User-Privacy:** The *user-privacy* of any POML algorithm A measures the probabilistic advantage that an attacker a (a user participating in the POML protocol or an external user) gains towards learning the preferred location L_j of at least one other user $j \in \{1, \ldots, N\}$, except the final optimal meeting location L_{opt}, after all users have participated in the execution of the POML protocol. Clearly, an external user does not learn about any preferred locations as it does not receive the output of the algorithm. Therefore, we only consider the non-trivial case of users participating in the POML protocol as attackers, i.e., u_a where $a \in \{1, \ldots, N\}$.

We express the user-privacy in terms of three adversary advantages. First, we measure the *identifiability advantage*, which is the probabilistic advantage of u_a in correctly guessing the preferred location L_i of any user $u_i \neq u_a$. We denote it as $Adv_a^{IDT}(A)$. Second, we measure the *distance-linkability advantage*, which is the probabilistic advantage of u_a in correctly guessing whether the distance d_{ij} between any two users $u_i \neq u_j$, is greater than a given parameter s, without necessarily knowing any users' preferred locations L_i, L_j. We denote this advantage as Adv_a^{d-LNK}. Finally, we measure the *coordinate-linkability advantage*, which is the probabilistic advantage of u_a in correctly guessing whether a given coordinate x_i (or y_i) of a user u_i is greater than the corresponding coordinate(s) of another user $u_j \neq u_i$, i.e., x_j (or y_j), without necessarily knowing any users' preferred locations L_i, L_j. We denote this advantage as Adv_a^{c-LNK}. The next observation follows from the above definitions.

Observation 1: If an adversary has an identifiability advantage over any two distinct users $u_i \neq u_j$, this implies it has distance- and coordinate-linkability advantages over those two users as well. However, the inverse is not necessarily true.

We semantically define the identifiability and linkability advantages by using a challenge-response methodology, which has been widely used to prove the security of cryptographic protocols. We now describe such a challenge-response game for the identifiability advantage $Adv_a^{IDT}(A)$ of any adversary u_a in a POML algorithm A.

1) Initialization: Challenger privately collects $\mathbb{L} = \{L_i\}_{i=1}^{N}$, where $L_i = (x_i, y_i)$ is the preferred meeting location of user u_i, and $f(L_i)$, $\forall i \in \{1, \ldots, N\}$.
2) POML algorithm: Challenger executes the POML algorithm A with the N users and computes $f(L_{opt}) = g(f(L_1), \ldots, f(L_N))$. It then sends $f(L_{opt})$ to each user $u_i, \forall i \in \{1, \ldots, N\}$.
3) Challenger randomly chooses a user u_a, $a \in \{1, \ldots, N\}$, as the adversary.
4) u_a chooses $u_j \neq u_a$ and sends j to the challenger.
5) Challenge: Challenger chooses a random $k \in \{1, \ldots, N\}, k \neq a$ and sends L_k to the adversary. The challenge is to correctly guess whether $L_k = L_j$.
6) The adversary sends L_j^* to the challenger. If the adversary thinks that L_k is the preferred meeting location of user u_j, i.e., if $L_k = L_j$ then the adversary sets $L_j^* = 1$. If the adversary thinks that L_k is not the preferred meeting location of user u_j, then he sets $L_j^* = 0$.

The identifiability advantage $Adv_a^{IDT}(A)$ can be defined as

$$Adv_a^{IDT}(A) = |Pr[(L_j^* = 1 \wedge L_k = L_j) \cup (L_j^* = 0 \wedge L_k \neq L_j)] - 1/(N-1)|$$

where $Pr[L_j^* = L_k]$ is the probability of user u_a winning the game by correctly answering the challenge, computed over the coin tosses of the challenger, L_j^* is u_a's guess about the preferred location of user $u_j \neq u_a$, L_k is the preferred location of user $u_k \neq u_a$ and $1/(N-1)$ is the probability of a random guess.

Similarly, we define the distance-linkability advantage $Adv_a^{d-LNK}(A)$ of any adversary u_a in a POML algorithm A by means of the following challenger-adversary game.

1) Initialization: Challenger privately collects $\mathbb{L} = \{L_i\}_{i=1}^{N}$, where $L_i = (x_i, y_i)$ is the preferred meeting location of user u_i, and $f(L_i)$, $\forall i \in \{1, \ldots, N\}$.
2) POML algorithm: Challenger executes the POML algorithm A with the N users and computes $f(L_{opt}) = g(f(L_1), \ldots, f(L_N))$. It then sends $f(L_{opt})$ to each user $u_i, \forall i \in \{1, \ldots, N\}$.
3) Challenger randomly chooses a user u_a, $a \in \{1, \ldots, N\}$, as the adversary.
4) u_a chooses $u_j, u_k \neq u_a$ and sends (j, k) to the challenger.
5) Challenge: Challenger computes a value s, e.g., the average Euclidian distance $d = \sum_{n=1}^{N-1} \sum_{m=n+1}^{N} d_{nm}/(2N(N-1))$ between any two users $u_n \neq u_m$, and sends (j, k, s) to the adversary. The challenge is to correctly guess whether $d_{jk} < s$.
6) The adversary sends d^* to the challenger. If the adversary thinks that $d_{jk} < s$ then he sets $d^* = 1$, otherwise $d^* = 0$. The adversary wins the game if: (i) $d^* = 1 \wedge d_{jk} < s$ or (ii) $d^* = 0 \wedge d_{jk} \geq s$. Otherwise, the adversary loses.

The distance-linkability advantage $Adv_a^{d-LNK}(A)$ can be defined as

$$Adv_a^{d-LNK}(A) = |Pr[(d^* = 1 \wedge d_{jk} < s) \vee (d^* = 0 \wedge d_{jk} \geq s)] - \frac{1}{2}|$$

where $Pr[.]$ is the probability of the adversary u_a winning the game by correctly answering the challenge, computed over the coin tosses of the challenger, d^* is the guess of the adversary, d_{jk} is the distance between L_j, L_k and s is a parameter chosen by the challenger.

Finally, we define the coordinate-linkability advantage $Adv_a^{c-LNK}(A)$ of any adversary u_a in a POML algorithm A by means of the following challenger-adversary game.

1) Initialization: Challenger privately collects $\mathbb{L} = \{L_i\}_{i=1}^N$, where $L_i = (x_i, y_i)$ is the preferred meeting location of user u_i, and $f(L_i)$, $\forall i \in \{1, \ldots, N\}$.
2) POML algorithm: Challenger executes the POML algorithm A with the N users and computes $f(L_{opt}) = g(f(L_1), \ldots, f(L_N))$. It then sends $f(L_{opt})$ to each user $u_i, \forall i \in \{1, \ldots, N\}$.
3) Challenger randomly chooses a user u_a, $a \in \{1, \ldots, N\}$, as the adversary.
4) u_a chooses $u_j, u_k \neq u_a$ and sends (j, k) to the challenger.
5) Challenge: Challenger chooses a coordinate axis $c \in \{x, y\}$ and sends (j, k, c) to the adversary. The challenge is to correctly guess whether $c_j < c_k$.
6) The adversary sends c^* to the challenger. If the adversary thinks that $c_j < c_k$ then he sets $c^* = 1$, otherwise $c^* = 0$. The adversary wins the game if: (i) $c^* = 1 \wedge c_j < c_k$ or (ii) $c^* = 0 \wedge c_j \geq c_k$. Otherwise, the adversary loses.

The coordinate-linkability advantage $Adv_a^{c-LNK}(A)$ can be defined as

$$Adv_a^{c-LNK}(A) = |Pr[(c^* = 1 \wedge c_j < c_k) \vee (c^* = 0 \wedge c_j \geq c_k)] - \frac{1}{2}|$$

where $Pr[.]$ is the probability of the adversary u_a winning the game by correctly answering the challenge, computed over the coin tosses of the challenger, c^* is the guess of the adversary, c_j is the spatial coordinate (x_j or y_j) randomly chosen by the challenger. u_a sets $c^* = 1$ if he thinks $c_j - c_k < s$, otherwise $c^* = 0$.

We now define the user-privacy of any POML algorithm A on a per-execution basis in the following way.

Definition 1: An execution of the POML algorithm A is *user-private* if the identifiability advantage $Adv_a^{IDT}(A)$, the distance-linkability advantage $Adv_a^{c-LNK}(A)$ and the coordinate-linkability advantage $Adv_a^{c-LNK}(A)$ of each participating user $u_i, i \in \{1, \ldots, N\}$ are negligible in the number of user-preferred meeting locations L_i.

In general, a function $f(x)$ is called negligible if, for any positive polynomial $p(x)$, there is an integer B such that for any integer $x > B, |f(x)| < 1/p(x)$ [22]. According to Definition 1, an execution of the POML algorithm is user-private if and only if any user u_a does not gain any (actually, negligible) additional knowledge about the preferred meeting locations L_j of any other user $u_j \neq u_a$, except the value of the final optimal meeting location L_{opt}.

*2) **Server-Privacy:*** The *server-privacy* of any POML algorithm A measures the probabilistic advantage that the LDS gains in learning the preferred meeting locations L_i of any user u_i, $i \in \{1, \ldots, N\}$. As in the case of user-privacy, we express the server-privacy by means of three advantages. First, we measure the probabilistic advantage of an LDS in correctly guessing the preferred location L_i of any user u_i, called *identifiability advantage* and denoted as $Adv_{LDS}^{IDT}(A)$. Second, we measure the probabilistic advantage of an LDS in correctly guessing whether the distance d_{ij} between any two users $u_i \neq u_j$ is greater than a given parameter s, without necessarily knowing any users' preferred locations L_i, L_j. We call this the *distance-linkability advantage* and we denote it as $Adv_{LDS}^{d-LNK}(A)$. Third, we measure the probabilistic advantage in correctly guessing whether a given coordinate x_i (or y_i) is greater than the same coordinate of another user $j \neq i$, i.e., x_j (or y_j), without necessarily knowing any users' preferred locations L_i, L_j. We call this the *coordinate-linkability advantage* and we denote it as $Adv_{LDS}^{c-LNK}(A)$.

The server identifiability and linkability advantages are defined in a similar fashion as the user advantages, and are presented in Appendix A. The server-privacy of a POML algorithm A on a per-execution basis can then be defined as follows.

Definition 2: An execution of the POML algorithm A is *server-private* if the identifiability advantage $Adv_{LDS}^{IDT}(A)$, the distance-linkability advantage Adv_{LDS}^{c-LNK} and the coordinate-linkability advantage Adv_{LDS}^{c-LNK} of an LDS are negligible.

However, it is reasonable to assume that in practice users will be able to perform multiple executions of the POML protocol, possibly with different sets of participating users at each time. This is particularly true if such a meeting-location service is offered, for instance, by providers to their subscribers. As a consequence, privacy of a POML should be defined over multiple executions.

*3) **POML Privacy:*** We now formally express the privacy conditions that any POML algorithm A has to satisfy, based on the above definitions. First, we define a *private* execution of a POML algorithm as follows.

Definition 3: A *private* execution of any POML algorithm A is an execution which does not reveal more information than what can be derived from the inputs, the intermediate results and its output.

Based on how memory is retained over sequential executions, we define two types of algorithm executions, namely, dependent and independent.

Definition 4: An *independent* (respectively *dependent*) *execution* is a single private execution of the POML algorithm defined in Section III-A in which *no* (respectively *some*) information of an earlier and current execution is retained and passed to future executions.

The information that might be transferred from an earlier execution to the next can include past inputs to the algorithm, intermediate results (on the LDS) and the outputs of the algorithm. Based on the type of execution, the privacy conditions of a privacy-preserving meeting-location algorithm can be defined as follows.

Definition 5: A meeting-location algorithm A is *execution* (respectively *fully*) privacy-preserving if and only if for every *independent* (respectively *all*) execution(s)

1) A is correct; All users are *correctly* able to compute the final optimal meeting location L_{opt};
2) A is *user-private*;
3) A is *server-private*.

A fully privacy-preserving meeting-location (POML) algorithm is a much stronger (and difficult to achieve) privacy requirement. In this first work, we focus on achieving execution privacy. The relationship between a fully POML and execution POML algorithm is given by the following observation.

Observation 2: Any POML algorithm A, as defined in Section III-A, is execution privacy-preserving if it is fully privacy-preserving, but the inverse is not true.

In the next section, we present our solutions to the POML problem and analyze them with respect to their complexity and privacy aspects.

IV. Solutions and Analysis

In the previous section, we have defined the fundamental building blocks that constitute a POML problem, both from a functional perspective (as in Figure 1) and from a privacy context. From a practical point of view, however, the problem is to design specific solutions and protocols that can be implemented on existing commercial mobile devices. In order to achieve the integration between resource-constrained mobile devices and the existing client-server network paradigm, our solutions have to be efficient in terms of computations and communication complexities, while taking advantage of the increasingly available mobile broadband. Such criteria have to be carefully considered when choosing the optimization and transformation functions g and f, respectively.

In this section, we present our solution to the POML problem as follows. First, we discuss the mathematical tools that we use in order to model the optimization function g and the transformation functions f. In the following subsections, we define the optimization function g by taking advantage of the properties of three well-known cryptographic primitives that are used to implement the transformation function f. These primitives, in turn, will guarantee that no private information about the preferred locations of any user is leaked to any other user or third-party involved in the computations. Finally, by merging the f and g components of the POML algorithm, we design our complete POML protocol. We then analytically evaluate its privacy properties and its computation and communication complexities.

In order to separate the optimization part of the POML algorithm A from its implementation using cryptographic primitives, we first discuss the optimization function g and then the transformation function f.

A. Optimization Function g

In order to determine an *optimal* meeting location, there are several factors that need to be considered. First, the optimality criterion needs to consider the spatial constraints present in the problem. For example, a meeting location $L_{opt} = (x_l, y_l)$ among N users $\mathbb{U} = \{u_i\}_{i=1}^N$ might be optimal when all users can reach L_{opt} in a "fair" amount of time. Another criterion might be to minimize the total displacement of all users in order to reach L_{opt}, or simply making sure that no user is "too far" from L_{opt} with respect to another user. Second, computing an optimal meeting location in a privacy-preserving way requires g to perform optimization operations in an oblivious fashion, by taking into account the properties of appropriate cryptographic schemes. Features such as homomorphic encryption and semantic security are of particular interest in this work, as they allow operations on the plain text components to be accomplished by computations on encrypted elements (homomorphic encryption), while ensuring that two identical plain-text components are encrypted in indistinguishable and unlinkable ciphertext elements (semantic security).

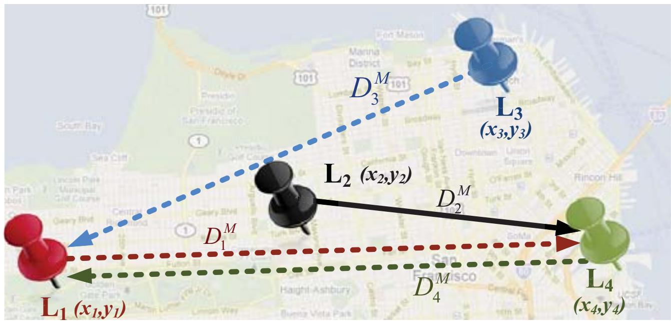


Figure 3. POML scenario, where the optimization function is $g = \text{argmin}(\max_i D_i^M)$. The dashed arrows represent the maximum distance D_i^M from each user u_i to any user $j \neq i$, whereas the solid line is the minimum of all such maximum distances. The optimal meeting location is $L_{opt} = L_2 = (x_2, y_2)$.

In this work, we consider the optimality criterion that has been widely used in operations research to solve the *k-center* problem. In the k-center problem, the goal is to find $L_1, \ldots, L_k$ locations among N given possible places, in order to optimally place k facilities, such that the maximum distance from any place to its closest facility is minimized. For a two dimensional coordinate system, the Euclidian distance metric is usually employed. As the POML problem consists in determining the optimal meeting location from a set of user-desired locations, we focus on the *k-center* formulation of the problem with $k = 1$. This choice is also grounded on the fact that not choosing L_{opt} from one of the location preferences $L_1, \ldots, L_N$ might potentially result in a location L_{opt} that is not suited for the kind of meeting that the participants require. Usually, a business meeting has different logistics requirements than a student party, and therefore the k-center formulation ensures that the final optimal location is a location that has been proposed by one of the participants. Figure 3 shows an example POML scenario modeled as a k-center problem, where four users want to determine the optimal meeting location L_{opt}.

In order to formally define the POML problem using the k-center formulation, we need to introduce the following notation.

- $d_{ij} \geq 0$ is the Euclidian distance between two points $L_i, L_j \in \mathbb{N}^2$
- $D_i^M = \max_{j \neq i} d_{ij}$ is the maximum distance from L_i to any other point L_j $(j \neq i)$.

Based on such notation, we can now formally define the

POML problem as follows.

Definition 6: The *Privacy-preserving Optimal Location* (POML) problem is to determine a location L_{opt} such that

$$L_{opt} \in \mathbb{L} = \{L_1, \ldots, L_N\}, \text{ where } opt = arg \min_i D_i^M$$

In other words, solving the POML problem is equivalent to finding the optimal meeting location, among the set of proposed (and user-desired) locations, such that the distance of the furthest desired location to the optimal one is minimized.

There are two important steps involved in the computation of the optimal location L_{opt}. The first step is to compute the pairwise distances d_{ij} among all users $i, j \in \{1, \ldots, N\}$ participating in the POML algorithm. The second step requires the computations of the maximum and minimum values of such distances. Before proceeding with these computations, in the following subsection we examine the features provided by the cryptographic functions that will ensure the privacy of individual user-desired locations L_i, $\forall i = 1, \ldots, N$.

B. Transformation Functions f

The optimization function g, as defined in the previous subsection, requires the computation of two functions of the private user-desired locations L_i, $\forall i \in \{1, \ldots, N\}$: (i) a function that computes the distance between any two locations $L_i \neq L_j$ and (ii) a function that minimizes the maximum of these distances. In order to achieve the final result and preserve the privacy of the personal information, we need to rely on computationally secure functions that possess the features that are required for such computations.

There are several cryptographic schemes that can be used, but not all of them provide the same features. We are interested in using secure schemes that allow us to compute the Euclidian distance between two points in the plane and the maximization/minimization functions. In our protocol, we consider three such encryption schemes: the *Boneh-Goh-Nissim* (BGN) [15], the *ElGamal* [23] and the *Paillier* [24] public-key encryption schemes.

Specifically, we are interested in the homomorphic properties of these schemes, such as the multiplicative and additive properties, as well as the randomness in the encryption. Given two plaintexts m_1, m_2 with their respective encryptions $E(m_1), E(m_2)$, the multiplicative property states that $E(m_1) \odot E(m_2) = E(m_1 \cdot m_2)$, where $\odot$ is an arithmetic operation in the encrypted domain that is equivalent to the usual multiplication operation in the plaintext domain. The additive homomorphic property states that $E(m_1) \oplus E(m_2) = E(m_1 + m_2)$, where $\oplus$ is an arithmetic operation in the encrypted domain which is equivalent to the usual sum operation in the plaintext domain. The randomness in the encryption ensures that even if $m_1 = m_2$, $E(m_1) \neq E(m_2)$ if two distinct random numbers are used for the encryption. We now briefly describe such homomorphic properties of the three encryption schemes BGN, ElGamal and Paillier. For conciseness, we omit the details of the initialization and operation of these schemes, which can be found in the respective references.

a) **BGN:** Given two plaintexts $m_1, m_2 \in \mathbb{Z}_T^*$ (where $T < q$ and q is a large prime) with their respective encryptions $E(m_1), E(m_2)$, the BGN possesses the following *multiplicative* and *additive* homomorphic properties

$$E(m_1 \cdot m_2) = e(E(m_1), E(m_2)) \cdot h_1^r \mod n$$
$$E(m_1 + m_2) = E(m_1) \cdot E(m_2) \cdot h^r \mod n$$

where $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is an admissible bilinear map, $\mathbb{G}, \mathbb{G}_1$ are two bilinear groups of composite order $n = pq$ (p, q are two large primes), h, g are public, $h_1 = e(g, h)$ and $r \in \mathbb{Z}_n$ is a random integer. BGN is an elliptic curve-based scheme and therefore much shorter keys can be used compared to ElGamal and RSA. A 160-bit key in elliptic curve cryptosystems is generally believed to provide equivalent security as a 1024-bit key in RSA and ElGamal [25]. However, due to the construction of the BGN scheme, only one homomorphic multiplication on each encrypted element is allowed, whereas an infinite number of homomorphic additions can be performed.

b) **Paillier:** Given $m_1, m_2 \in \mathbb{Z}_n^*$, it can be verified that the Paillier encryption scheme satisfies the following multiplicative and additive homomorphic properties

$$E(m_1 \cdot m_2) = E_r(m_1)^{m_2} \mod n^2$$
$$E(m_1 + m_2) = E_{r_1}(m_1) \cdot E_{r_1}(m_2) \mod n^2$$

where $r_i, r \in \mathbb{Z}_n^*$ are random integers and $n = pq$ where p, q are two large primes.

c) **ElGamal:** Given $m_1, m_2 \in \mathbb{Z}_n^*$, where n is a large prime, the ElGamal encryption schemes satisfies the following multiplicative property

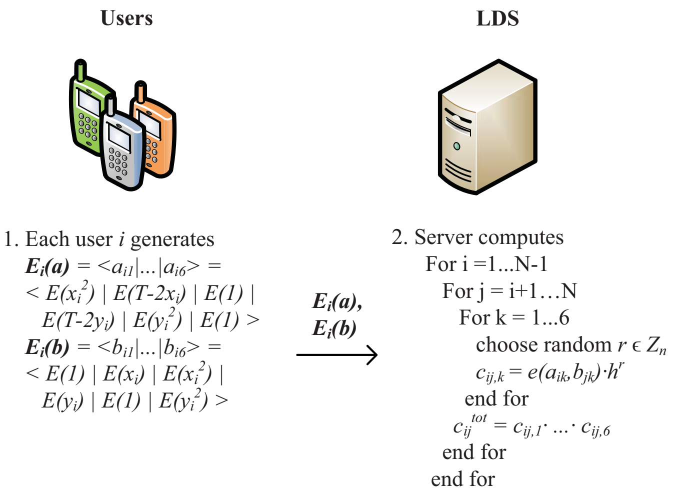
$$E(m_1 \cdot m_2) = E(m_1) \cdot E(m_2) \mod n$$

Based on the three aforementioned encryption schemes, we now describe the distance computation algorithms that are used in our solution.

C. Distance Computations

Computing the pairwise distance d_{ij} between any two user-desired locations L_i, L_j is the main requirement for solving the POML problem. In order to determine the optimal meeting location, we need to find the location L_{opt}, where $opt \in \{1, \ldots, N\}$, that minimizes the maximum distance between any user-desired location and L_{opt}. In our algorithms, we work with the *square* of the distances, as they are much easier to compute in an oblivious fashion using the homomorphic properties of the cryptographic schemes. The problem of finding the argument that minimizes the maximum distance is equivalent to finding the argument that minimizes the maximum distance *squared* (provided that all distances are greater than 1).

Hereafter we propose two distance computation modules that will be used in our POML protocol. Each of these modules computes the square of all pairwise distances between any two user-desired locations, and preserves the privacy of each user's preferred location L_i, $\forall i \in \{1, \ldots, N\}$.



```
1. Each user $i$ generates
  $E_i(a) = <a_{i1}|...|a_{i6}> =$
  $< E(x_i^2) | E(T-2x_i) | E(1) |$
    $E(T-2y_i) | E(y_i^2) | E(1) >$
  $E_i(b) = <b_{i1}|...|b_{i6}> =$
  $< E(1) | E(x_i) | E(x_i^2) |$
    $E(y_i) | E(1) | E(y_i^2) >$

2. Server computes
  For i =1...N-1
    For j = i+1…N
      For k = 1...6
        choose random $r \in Z_n$
        $c_{ij,k} = e(a_{ik},b_{jk})\cdot h^r$
      end for
      $c_{ij}^{tot} = c_{ij,1}\cdot \ldots \cdot c_{ij,6}$
    end for
  end for
```

Figure 4. Distance computation protocol based on the BGN encryption scheme.

1) **BGN-distance:** The first distance computation algorithm is based on the BGN encryption scheme and is shown in Figure 4. This novel protocol requires only one round of communication between each user and the LDS, and it efficiently uses both the multiplicative and additive homomorphic properties of the BGN scheme. The algorithm works as follows. In step 1, each user u_i, $\forall i \in \{1, \ldots, N\}$ creates the vectors

$$\begin{aligned}E_i(a) &=< a_{i1}| \ldots |a_{i6} >\\ &=< E(x_i^2)|E(T-2x_i)|E(1)|E(T-2y_i)|E(y_i^2)|E(1) >\\ E_i(b) &=< b_{i1}| \ldots |b_{i6} >\\ &=< E(1)|E(x_i)|E(x_i^2)|E(y_i)|E(1)|E(y_i^2) >\end{aligned}$$

where $E(.)$ is the encryption of $(.)$ using the BGN scheme and $L_i = (x_i, y_i)$ is the desired meeting location of user u_i. Afterwards, each user sends the two vectors $E_i(a), E_i(b)$ over a secure channel to the LDS. In step 2, the LDS computes the scalar product of the received vectors by first applying the multiplicative and then the additive homomorphic property of the BGN scheme, as shown in Figure 4. For example, in a scenario with two users, one can easily verify that

$$\begin{aligned}E_i(a) \bullet E_j(b) = E(x_i^2 + x_j(T - 2x_i) + x_j^2\\ + y_j(T - 2y_i) + y_i^2 + y_j^2 \mod T)\\ = E(d_{ij}^2 \mod T)\end{aligned}$$

where T is chosen such that $\forall i, j \in \{1, \ldots, N\}, d_{ij}^2 < T$. At this point, the LDS has obliviously computed $E(d_{ij}^2)$, which is the (encrypted) square of the pairwise distances between all pairs L_i, L_j of user-desired locations, where $i \neq j$.

2) **Paillier-ElGamal-distance:** An alternative scheme for the distance computation is based on both the Paillier and ElGamal encryption schemes, as shown in Figure 5. As neither Paillier or ElGamal possess both multiplicative and additive properties, the resulting algorithm requires one extra step in order to achieve the same result as the BGN-based scheme, i.e., obliviously computing the pairwise distances d_{ij}^2. The distances are computed as follows. In step 1, each user u_i, $\forall i \in \{1, \ldots, N\}$ creates the vector

$$\begin{aligned}E_i(a) &=< a_{i1}| \ldots |a_{i4} >\\ &=< Pai(x_i^2)|ElG(x_i)|Pai(y_i^2)|ElG(y_i) >\end{aligned}$$

where $Pai(.)$ and $ElG(.)$ refer to the encryption of $(.)$ using the Paillier or ElGamal encryption schemes, respectively. Afterwards, each user u_i sends the vector $E_i(a)$ to the LDS, encrypted with LDS's public key. In step 2.1, the LDS computes the scalar product of the second and fourth element of the received vectors (as shown in Figure 5). In order to hide this intermediate result from the users, the LDS obliviously randomizes these results with random values r_s, r_t. At the same time, the LDS computes the multiplicative inverse of such values, denoted as r_s^{-1} and r_t^{-1} respectively. These randomized scalar products are denoted as $c_{ij,s}$ and $c_{ij,t}$. In step 2.2, the LDS permutes the order of all $c_{ij,s}$ and $c_{ij,t}$ with its private element-permutation function $\sigma = [\sigma_1, \ldots, \sigma_{N(N-1)}]$, and sends N such distinct elements to each user u_i. In step 3, each user simply decrypts the received elements with the ElGamal private key and re-encrypts them with the Paillier public key. Then each user sends the re-encrypted elements to the LDS in the same order as he received it. In step 4, the LDS reverts the element-permutation function σ, and in step 4.1 it finally computes the d_{ij}^2 for all i, j, after having removed the randomizing factors $r_{ij,s}, r_{ij,t}$ with their inverses $r_{ij,sinv}$ and $r_{ij,tinv}$. At this point, the LDS has securely computed $E(d_{ij}^2)$, the (encrypted) square of the pairwise distances between all pairs of user-desired locations $L_i \neq L_j$.

As the ElGamal-Paillier based distance computation involves decryption/re-encryption operations, it may be possible for participants to maliciously change the masked values. For instance, such an active attack could be performed in order to disrupt the distance computations or to manipulate the result for personal advantage (such as a personally convenient but generally suboptimal meeting location). We discuss such active attacks and defense mechanisms in Section VI.

D. The POML *Protocol*

In the previous subsections, we defined all the necessary operations and cryptographic tools that are required in order to solve the POML problem. We now describe our algorithm that solves the POML problem in an efficient and privacy-preserving way, as shown in Figure 6. The protocol has three main modules: (A) the distance computation module, (B) the MAX module and (C) the ARGMIN MAX module.

Distance computations: The first module (distance computation) uses one of the two protocols defined in the previous subsection (BGN-distance or Paillier-Elgamal distance). We note that modules (B) and (C) use the same encryption scheme as the one used in module (A). In other words, $E(.)$ of Figure 6 refers to the encryption of $(.)$ using either the BGN or the Paillier encryption scheme. Once the distance protocol has been decided, the next modules (B) and (C) are executed as follows.

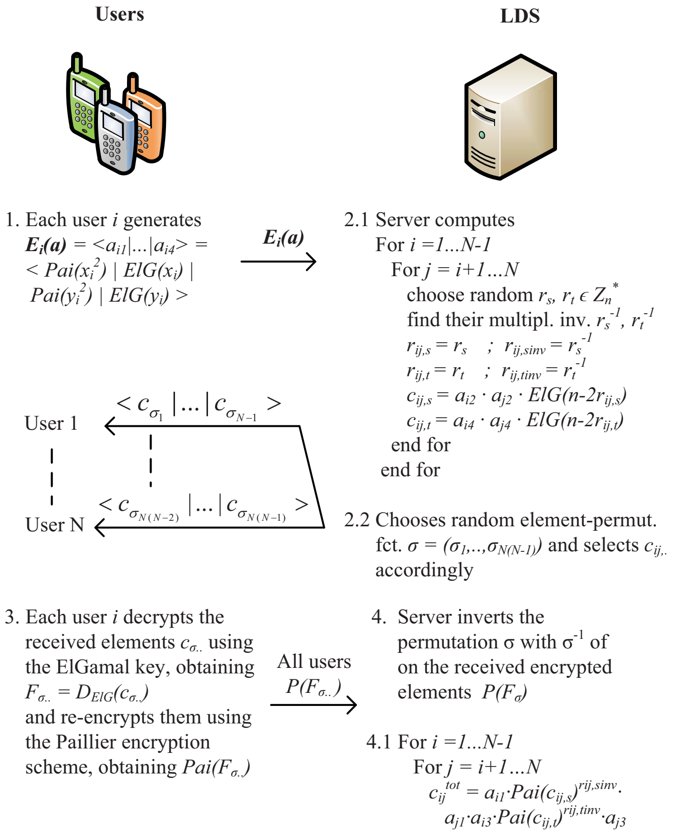

Users
LDS
1. Each user i generates $E_i(a) = <a_{i1}|...|a_{i4}> = < Pai(x_i^2) | ElG(x_i) | Pai(y_i^2) | ElG(y_i) >$
$E_i(a)$
2.1 Server computes
For $i = 1...N-1$
For $j = i+1...N$
choose random $r_s, r_t \in Z_n^*$
find their multipl. inv. r_s^{-1}, r_t^{-1}
$r_{ij,s} = r_s$; $r_{ij,sinv} = r_s^{-1}$
$r_{ij,t} = r_t$; $r_{ij,tinv} = r_t^{-1}$
$c_{ij,s} = a_{i2} \cdot a_{j2} \cdot ElG(n-2r_{ij,s})$
$c_{ij,t} = a_{i4} \cdot a_{j4} \cdot ElG(n-2r_{ij,t})$
end for
end for
User 1
$< c_{\sigma_1} | ... | c_{\sigma_{N-1}} >$
User N
$< c_{\sigma_{N(N-2)}} | ... | c_{\sigma_{N(N-1)}} >$
2.2 Chooses random element-permut. fct. $\sigma = (\sigma_1,..,\sigma_{N(N-1)})$ and selects $c_{ij,.}$ accordingly
3. Each user i decrypts the received elements $c_{\sigma..}$ using the ElGamal key, obtaining $F_{\sigma..} = D_{ElG}(c_{\sigma..})$ and re-encrypts them using the Paillier encryption scheme, obtaining $Pai(F_{\sigma..})$
All users $P(F_{\sigma..})$
4. Server inverts the permutation σ with σ^-1 of on the received encrypted elements $P(F_\sigma)$
4.1 For $i = 1...N-1$
For $j = i+1...N$
$c_{ij}^{tot} = a_{i1} \cdot Pai(c_{ij,s})^{rij,sinv} \cdot a_{j1} \cdot a_{i3} \cdot Pai(c_{ij,t})^{rij,tinv} \cdot a_{j3}$

Figure 5. Distance computation protocol based on the ElGamal and Paillier encryption schemes.

MAX computations: In step B.1, the LDS needs to be obliviously hide the values within the encrypted elements (i.e., the pairwise distances computed earlier), before sending them to the users, in order to avoid leaking any kind of private information such as the pairwise distance or desired locations to any user[2]. In order to obliviously mask such values, for each index i the LDS generates two random values r_i, s_i that are used to scale and shift the c_{ij}^{tot} (the encrypted square distance between L_i, L_j) for all j, obtaining d_{ij}^*. This is done in order to (i) ensure privacy of real pairwise distances, (ii) be resilient in case of collusion among users and (iii) preserve the internal order (the inequalities) among the pairwise distance from each user to all other users. Afterwards, in step B.2 the LDS chooses two private element-permutation functions σ (for i) and θ (for j) and permutes d_{ij}^*, obtaining the permuted values $d_{\sigma_i \theta_j}^*$, where $i, j \in \{1, \ldots, N\}$. The LDS sends N such distinct elements to each user. In step B.3, each user decrypts the received values, determines their maximum and sends the index σ_i^{MAX} of the maximum value to the LDS. In step B.4 of the MAX module (B), the LDS inverts the permutation functions σ, θ and removes the masking from the received indexes corresponding to the maximum distance values.

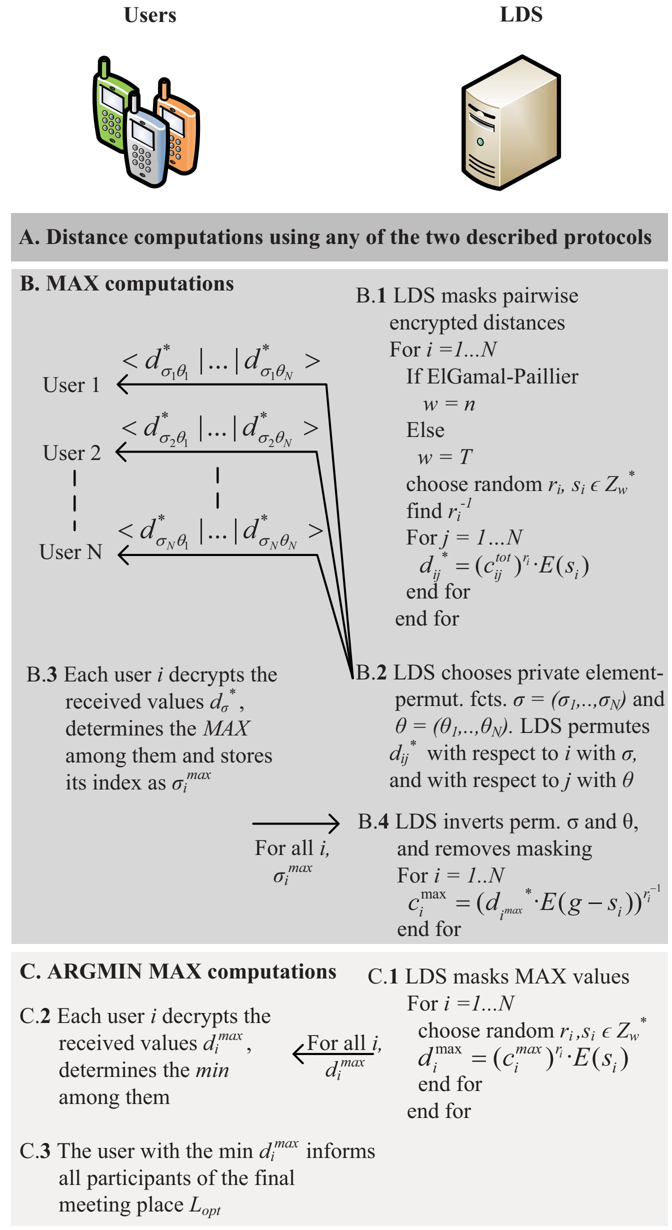

Users
LDS
A. Distance computations using any of the two described protocols
B. MAX computations
B.1 LDS masks pairwise encrypted distances
For $i = 1...N$
If ElGamal-Paillier
$w = n$
Else
$w = T$
choose random $r_i, s_i \in Z_w^*$
find r_i^{-1}
For $j = 1...N$
$d_{ij}^* = (c_{ij}^{tot})^{r_i} \cdot E(s_i)$
end for
end for
User 1
$< d_{\sigma_1\theta_1}^* | ... | d_{\sigma_1\theta_N}^* >$
User 2
$< d_{\sigma_2\theta_1}^* | ... | d_{\sigma_2\theta_N}^* >$
User N
$< d_{\sigma_N\theta_1}^* | ... | d_{\sigma_N\theta_N}^* >$
B.3 Each user i decrypts the received values d_σ^*, determines the MAX among them and stores its index as σ_i^{max}
B.2 LDS chooses private element-permut. fcts. $\sigma = (\sigma_1,..,\sigma_N)$ and $\theta = (\theta_1,..,\theta_N)$. LDS permutes d_{ij}^* with respect to i with σ, and with respect to j with θ
For all i, σ_i^{max}
B.4 LDS inverts perm. σ and θ, and removes masking
For $i = 1..N$
$c_i^{max} = (d_{i^{max}}^* \cdot E(g - s_i))^{r_i^{-1}}$
end for
C. ARGMIN MAX computations
C.1 LDS masks MAX values
For $i = 1...N$
choose random $r_i, s_i \in Z_w^*$
$d_i^{max} = (c_i^{max})^{r_i} \cdot E(s_i)$
end for
end for
C.2 Each user i decrypts the received values d_i^{max}, determines the min among them
For all i, d_i^{max}
C.3 The user with the min d_i^{max} informs all participants of the final meeting place L_{opt}

Figure 6. Privacy-preserving Optimal Meeting-Location protocol (POML).

ARGMIN MAX computations: In step C.1, the LDS masks the true maximum distances by scaling and shifting all of them by the same amount, such that their order (the inequalities among them) is preserved. Then the LDS sends to each user all the masked maximum distances. In step C.2 each user decrypts the received masked maximum values, and determines the minimum value among all maxima. In step C.3, each user knows which identifier corresponds to himself, and the user with the minimum distance sends to all other users his desired meeting location in an anonymous way.

After the last step, each user receives the final optimal

[2] We note that, after the distance computation module (A), the LDS possesses all encrypted pairwise distances. This encryption is made with the public key of the participants and therefore the LDS cannot decrypt the distances without the corresponding private key. The oblivious (and order-preserving) masking performed by the LDS at step B.1 is used in order to hide the pairwise distances from the users themselves, as otherwise they would be able to obtain these distances and violate the privacy of the users.

Table II
ASYMPTOTIC COMPLEXITY OF THE PROPOSED POML PROTOCOLS. THE VALUES IN BOLD CORRESPOND TO THE MOST EFFICIENT PROTOCOL WITH RESPECT TO THE GIVEN OPERATION, AND N IS THE NUMBER OF MEETING PARTICIPANTS.

CLIENT	**PROTOCOL**	**BGN** *(mod n)*	**ELGAMAL-PAILLIER** *(mod n^2)*
Multiplication	Distance	**O(1)**	O(N)
	POML		
Exponentiation	Distance	O(1)	**O(N)**
	POML	*$O(N\sqrt{T})$*	
Memory	Distance	**O(1)**	O(N)
	POML	*O(N)*	
Communication	Distance	**O(1)**	O(N)
	POML	*O(N)*	
LDS			
Multiplication, exponentiation	Distance	$O(N^2)$	$O(N^2)$
	POML		
Bilinear mapping	Distance	$O(N^2)$	-------
	POML		
Memory	Distance	$O(N^2)$	$O(N^2)$
	POML		
Communication	Distance	**O(N)**	$O(N^2)$
	POML	*$O(N^2)$*	

meeting location, but no other information regarding non-optimal locations or distances is leaked. In order to assess the efficiency and to know whether the proposed POML protocol fulfills the privacy requirements defined in Section III, we present the complexity and privacy analysis in the next subsection.

E. Analysis

Our POML protocol, shown in Figure 6, is based on the interaction between users and a third-party LDS. Each of these parties performs operations on both plaintext and encrypted elements, and the resources available on the user devices are usually lower than those of the LDS. Hereafter we discuss the asymptotic complexities of our two distance and POML protocols, by considering both client and LDS computation, communication and memory complexities. We summarize these results in Table II.

1) **Client complexity:** The number of multiplications for the BGN-based distance protocol (Figure 4) is independent on the number of participating clients ($O(1)$), as opposed to the ElGamal-Paillier based protocol ($O(N)$). However, this is not true for the number of exponentiations. As stated in the seminal paper [15], a single decryption in the BGN cryptosystem requires $O(\sqrt{T})$ exponentiations, where T is the order of the plaintext domain. This can be improved by using pre-computation, allowing only integer comparisons (instead of exponentiations) to successfully decrypt an element. On the contrary, the alternative algorithm is much more efficient ($O(N)$) for a reasonable number N of participants (compared to $O(N\sqrt{T})$), because the number of exponentiations is independent on the plaintext domain.

The memory requirements for the BGN-based distance protocol ($O(1)$) are lower than for the alternative one ($O(N)$), mainly because the former does not need to store values (masked pairwise distances) that depend on other users' inputs. Similarly, the BGN-based distance protocol is more efficient ($O(1)$ compared to $O(N)$). The homomorphic properties of BGN allow for the pairwise distance computations to be computed in a single round, avoiding unnecessary message exchanges. However, the complete POML protocol has the same communication complexity ($O(N)$) for both BGN and ElGamal-Paillier primitives.

2) **LDS complexity:** Both BGN- and ElGamal-Piallier-based POML algorithms have quadratic complexity ($O(N^2)$) on the server side. However, the BGN scheme uses $O(N^2)$ additional bilinear mappings in order to perform the homomorphic multiplication, which are expensive. These bilinear mappings are not required in the ElGamal-Paillier scheme. Nevertheless, the exponentiation operations can be pre-computed in BGN (because the h value is public), as opposed to the ElGamal-Paillier-based approach.

The memory and communication requirements are also similar for both the BGN- and ElGamal-Paillier-based protocols. The former is, however, more efficient with respect to the number of message exchanges in the distance computation phase ($O(N)$), compared to the latter ($O(N^2)$).

3) **Privacy:** Informally, a POML algorithm A is private if an adversary (participating user or LDS) does not learn the preferred meeting locations, the mutual distances and coordinate relations of any user, after A's execution. According to the previously outlined privacy definitions, we have the following result.

Proposition 1: The BGN and ElGamal-Paillier based POML algorithms are execution privacy-preserving.
In simple words, Proposition 1 states that both proposed algorithms correctly compute the optimal meeting location, given the received inputs, and that they do not reveal any users' preferred meeting locations to any other user, except the optimal meeting location L_{opt}. Moreover, the LDS does not learn any information about any user-preferred locations. The proof can be found in Appendix B.

V. IMPLEMENTATION PERFORMANCE AND DISCUSSION

In this section, we discuss the results of the performance measurements of our two POML algorithms, conducted on a testbed of Nokia N810 mobile devices (Figure 7). Readers should note that we measured only the computation time on the devices, without the message communication delays. As we use wireless peer-to-peer communication, such delays are negligible. For the elliptic curve BGN-based POML algorithm, we measured the performance of using both a 160-bit and a 256-bit secret key, whereas for the ElGamal-Paillier based algorithm we used the standard 1024-bit secret keys. As mentioned in Section IV, a 160-bit key in elliptic curve

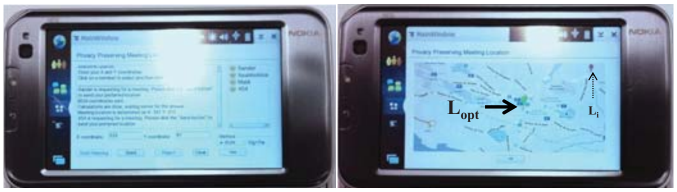


Figure 7. Prototype POML application running on a Nokia N810 mobile device. The image on the left is the main window, where users add the desired meeting participants. The image on the right is the map that shows the optimal meeting location (green pin, thick solid arrow) and the user-desired meeting location (red pin, thin dashed arrow), after the execution of the POML protocol. In this particular example, the optimal meeting location is not this user's preferred location, but the location preference of some other participant.

cryptosystems provides an equivalent resistance to attacks of a 1024-bit RSA key [25].

The client devices (N810s) are based on the ARM 400 MHz CPU and have 256 MB RAM. Their operating system is the Linux-based Maemo OS2008, and we wrote our applications using the Qt programming language, which is optimized for such OS. The LDS has been implemented on a 2 GHz Intel CPU with 3 GB RAM, running the Ubuntu 9.04 Linux. Hereafter we discuss the main performance measurements.

A. LDS performance

Figure 8(a) shows the time required by the LDS in order to perform the pairwise distance computations (module A of Figure 6). We can see that the computation time increases as the number of meeting participants increases, due to the greater quantity of pairwise distances that need to be computed. The ElGamal-Paillier algorithm is the most efficient, requiring 2 seconds to compute the distances among 10 participants. The two BGN-based algorithms are less efficient, but are still practical enough (7 seconds) when using a comparable security level to the ElGamal-Paillier algorithm. The CPU-intensive bilinear mappings in BGN are certainly one important reason for such results. This fact is even more evident when the security parameter in BGN is increased from 160 to 256 bits.

Regarding the subsequent modules B and C of the POML protocol, we observe a that the BGN-based algorithms outperform the ElGamal-Paillier one (Figure 8(b) and 8(c)). The maximum computations on the LDS require 0.5 seconds for the 160-bit BGN algorithm, whereas the ElGamal-Paillier takes almost 2 seconds. A similar result can be observed for the minimum computations. There are two main reasons for this. First, there are no bilinear mappings involved in these modules and second, the BGN-based algorithms use much smaller key sizes. From a practical perspective, both the ElGamal-Paillier and the BGN algorithms have good performance in modules B and C of the POML protocol.

B. Client performance

Figure 8(d) shows the time required to compute the distance (module A) on the Nokia N810 mobile device. As it can be seen, the BGN-based algorithm is the most efficient, requiring only 0.3 seconds, compared to the 1.4 seconds of the ElGamal-Paillier one with 2 users. Thanks to the homomorphic properties of BGN, the pairwise distances can be obliviously computed by the LDS, without involving any decryption/re-encryption operation from the clients (as opposed to the ElGamal-Paillier alternative). Even with a comparatively larger security resistance, the BGN scheme is still faster than the alternative one. It is also important to notice that the design of the proposed BGN distance algorithm allows it to perform well, independently of the number of participants.

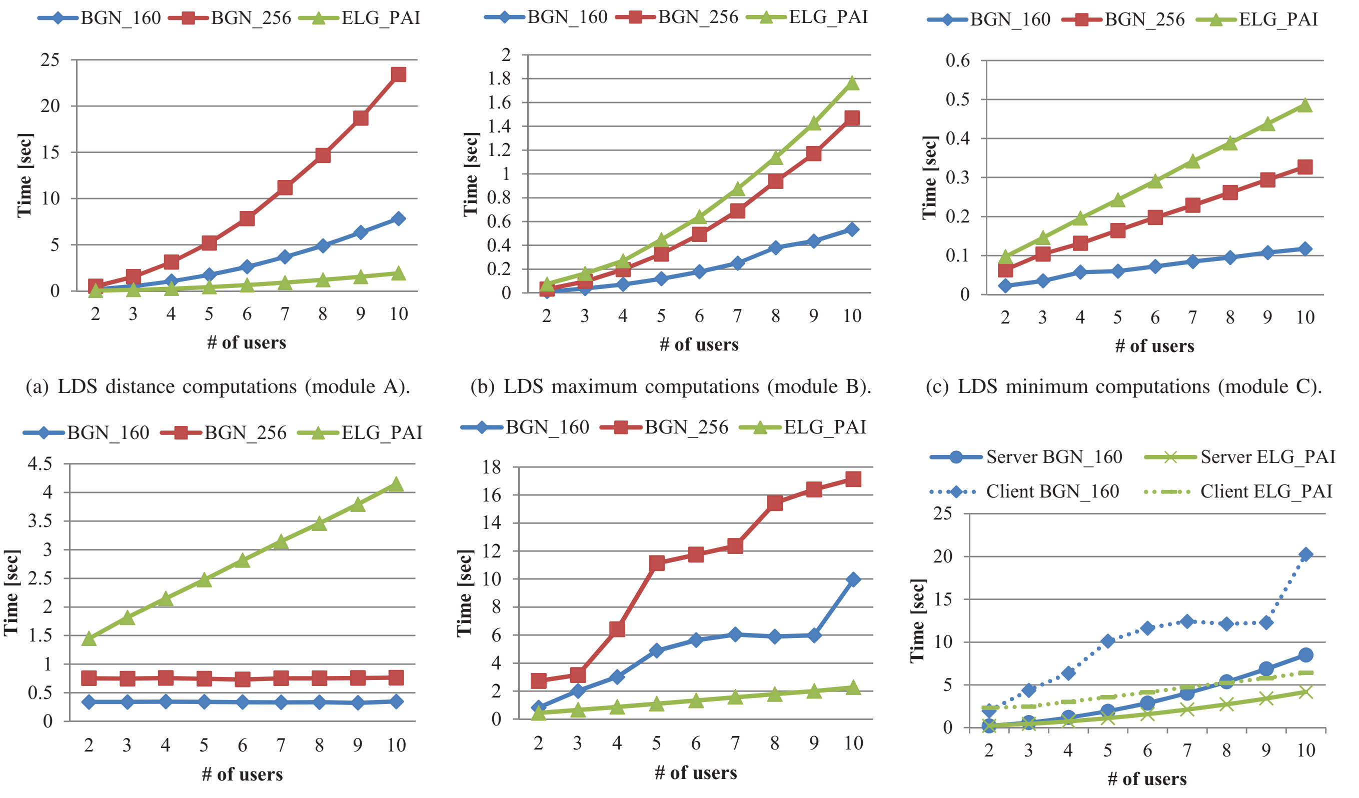
The performance of the maximization/minimization computations are shown in Figure 8(e). As it can be seen, the performance of the BGN-based algorithm in computing such operations (10 seconds for 10 participants) is significantly worse than the comparably secure ElGamal-Paillier based algorithm (2 seconds for 10 participants). The maximization/minimization operations require the client to decrypt the obliviously masked pairwise distances. This is mainly due to the complexity of this operation using the BGN cryptosystem, which depends on the number of participants and the value inside the encrypted message [15]. On the contrary, the ElGamal-Paillier based decryptions are independent of the value inside the encrypted message.

C. Total runtime

Figure 8(f) shows the total runtime for the client and the LDS, computed as the sum of the time required for one complete execution of the POML protocol (Figure 6, time for module A + B + C), excluding the communication delays. As it can be seen, the total runtime for both client and LDS is lower by using the ElGamal-Paillier based POML algorithm, compared to the BGN-based one. With 10 participants, the former algorithm requires 6 seconds to obtain the optimal meeting location on the clients, whereas the LDS needs 4.5 seconds. Using the BGN-based POML algorithm, each client needs more than 20 seconds and the LDS requires approximatively 8 seconds. In other words, the ElGamal-Paillier based POML algorithm is almost two times faster than the BGN-based one.

D. Discussion

The implementation measurement results of our two POML algorithms show that the ElGamal-Paillier based algorithm has a better overall performance than the BGN-based alternative. On a comparable security level and without considering communication delays, both the LDS and the client device require less computation time for the former algorithm compared to the latter. However, aside from the performance, the BGN-based algorithm presents several advantages. First, it involves three less message exchanges between each client and the LDS (Figure 4) compared to the alternative algorithm. Second, much shorter security parameters can be used in order to achieve the same resistance to attacks, and thus lower the memory requirements on the client devices. Third, malicious users cannot change the masked user-preferred meeting-location coordinates once they have been sent to the LDS, as



(a) LDS distance computations (module A).

(b) LDS maximum computations (module B).

(c) LDS minimum computations (module C).

(d) Client distance computations (module A).

(e) Client max/argmin computations (module B/C).

(f) Total client and LDS run times (modules A+B+C).

Figure 8. Implementation performance measurements.

there are no decryption/re-encryption operations in the BGN-based algorithm, as opposed to the ElGamal-Paillier based one.

In the next section, we discuss other security and optimization issues related to the two proposed POML algorithms.

VI. Extensions

In the previous sections we presented our two POML algorithms and evaluated their performance on commercial mobile devices. In this section, we present some enhancements of the proposed algorithms in order to better cope with malicious attacks and to further increase the optimization properties of the final meeting location L_{opt}.

A. Security

According to Proposition 1, the two proposed POML algorithms are execution privacy-preserving. In other words, no (actually negligible) private information about users' preferred meeting locations or pairwise distances is leaked to any other user or third-party LDS. However, active malicious attacks are still possible, and we need to evaluate their feasibility and consequences on the user-privacy. Hereafter we discuss some possible active attacks on our POML protocols, and we present defensive mechanisms in order to thwart them.

There are three main categories of active attacks that could be perpetrated against our POML protocols, namely (i) the collusion among users and/or LDS, (ii) the fake user generation and/or replay attacks and (iii) preferred location modification.

With respect to the collusion attack, there could be scenarios where participants collude among themselves or where the LDS colludes with some participants. In the first case, the colluding participants would be able to know their preferred locations and their respective distances, but they will not be able to infer crucial information about the honest users' locations with certainty. Regardless of the protocol used or the encryption methods, the published optimal result (together with the additional information malicious users may get from colluders) can be used to construct exclusion zones, based on the set of equations and known parameters. An exclusion zone is a region that does not contain any location preferences, and the number of such exclusion zones increases with the number of colluders. We are currently working on quantifying this impact on our optimization and encryption methods. However, in the unlikely case of collusion between the LDS and the participants, the latter will be able to obtain other participants' preferences. In order to mitigate such a threat, the invited participants could agree on establishing a shared secret using techniques from threshold cryptography, such as [26]. The LDS should then collude with at least a predefined number of participants in order to obtain the shared secret and learn the individual preferred meeting locations. Nevertheless, considering both malicious users and malicious LDS at the same time is a rather strong assumption, due to the commercial interests service providers have in guaranteeing a faithful service to their customers.

Generating fake users can be attempted both by the LDS and by any meeting participant, in order to disrupt or manipulate the computations of the optimal meeting location. However, the security of our algorithms prevents such attacks from succeeding. In case the LDS generates fake users, it would not be able to obtain the secret that is shared among the honest users and which is used to derive the secret key $K_s^{M_v}$ for each session v. This attack is more dangerous if a legitimate participant creates a fake, because the legitimate participant knows the shared secret. In this scenario, however, the LDS knows the list of meeting participants (as it is computing the optimal meeting location) and therefore it would accept only messages digitally signed by each one of them. Replay attacks could be thwarted by adding an individually signed *nonce*, derived using the shared secret, in each user's meeting request and reply. These nonces would be forwarded to all other meeting participants, who would be able to verify the signature and check whether all received nonces are equal. A failure in this process would suggest that there has been tampering with the POML protocol.

The last type of active attack could lead to the determination of a suboptimal meeting location. Maliciously modifying or untruthfully reporting the maximum masked values (step B.3 of Figure 6) could deviate the LDS in accepting the false received index as the maximum value, and therefore potentially lead to the determination of a suboptimal meeting location. However, this is rather unlikely to happen in practice. For instance, even if in step B.3 a user falsely reports one of his values to be the maximum when actually it is not, this would cause the algorithm to select a suboptimal meeting location if and only if no other user selected a smaller value as the maximum distance.

B. Optimization

The optimal meeting location L_{opt} is determined based on the k-center problem formulation (Section IV). This fairness criterion chooses L_{opt} in a way to minimize the maximum displacement from any participant's preferred location to L_{opt}. Although appropriate, individual fairness might be complemented with aggregated displacement costs. For instance, minimizing $\sum_{i=1}^{N} |L_i - L_{opt}|^2$ (which is the sum of all distances to L_{opt}) may be taken into consideration together with the aforementioned fairness criterion. This would *de facto* force the system to find a compromise between the two criteria.

Such complements to the original optimality criterion can be easily integrated in the proposed POML algorithms. For instance, the total sum-displacement of all participants to L_{opt} can be easily (and obliviously) performed by the LDS, after the pairwise distances have been computed. Thanks to the additive homomorphic properties of the Paillier and the BGN cryptosystems, the sum of pairwise distances squared is a simple operation for both schemes: it requires the LDS to simply sum up all pairwise distances for each candidate meeting location. Then, the LDS masks and appends these sums to the already computed distances in step B.2 (Figure 6), and in step B.3 each client determines the maximum of such received values, in addition to the maximum of the masked distances squared. As the complexity of this operation is linear, with respect to the number of preferred locations, the proposed POML algorithms would not suffer from an increased computational demand.

In addition to the proposed k-center formulation of the POML problem, a different approach could be envisaged in scenarios where participants could choose more than one preferred meeting location, i.e., $L_i = \{L_{i,1}, \ldots, L_{i,h}\}$, and then apply the optimization to the set of private user-preferred locations. Alternatively, participants could propose a set of locations that are the furthest places where they are willing to meet. Then, given these locations, the POML problem could be solved by constructing, for each user, the minimum convex hull (containing each user's set of constraints) that would represent the region in which each user is willing to meet. The optimal meeting location L_{opt} would then be defined in the intersection (if any) of the private individual convex hulls. Depending on the context and the utilization scenario, this geometric intersection approach to the POML problem might be more suited than the proposed ones (based on the k-center formulation).

VII. User Study Results

Based on guidelines from ([27], [28]), we prepared and conducted a user study on 35 subjects, sampling a population of university students, faculty members and non-scientific staff. The goal of the study was to assess the sensitivity of the subjects to privacy issues in meeting-location applications, as well as to obtain feedback with respect to our prototype application. The questionnaire was based on 32 questions.

The results show that 63% of the respondents use the calendar/agenda on their mobile device to organize meetings. 57% of such meetings involve 4-6 participants, and only 14% have more than 6 participants. When asked about optimal meeting-location applications, 91% claim that they would be at least a little interested in such applications, and among those 52% would be very interested.

With respect to privacy in such applications, 66% agree that it is important to reveal only the necessary information to the system, in order to compute the optimal meeting location (Figure 9). The percentage grows to 89% if we include respondents who tend to agree with such statement as well. After having used our prototype application, 71% of the users tend to or appreciate that their meeting-location preferences were not revealed to others by the algorithm, and only 8% do not care about the privacy of their meeting-location preference.

VIII. Related Work

To the best of our knowledge, this is the first work to address the optimal meeting-location problem with privacy guarantees. Hereafter, we first present recent works that address, without protecting privacy, strategies to determine the optimal meeting location. Then, we discuss contributions in secure multiparty computation (SMC) on point-distance computations.

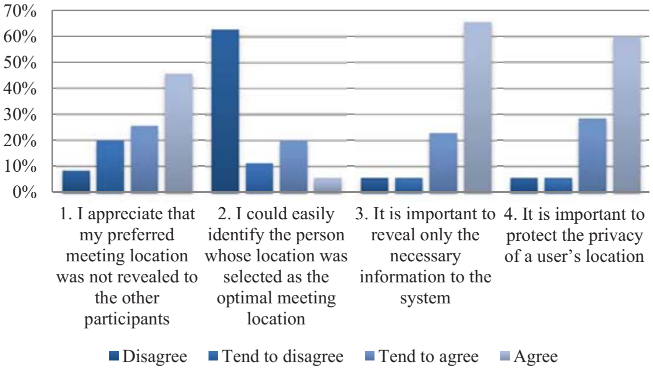


Figure 9. Extract of the user study questionnaire, regarding opinions about privacy issues related to meeting locations, as well as our prototype application. Note that we conducted the experiments in groups of 4-5 participants, hence one user out of 4-5 was the one who recognized his own location as being the optimal location. Thus the 20-25% of users who agree with the second statement.

In their recent work, Santos and Vaughn [29] present a survey of existing literature on meeting-location algorithms, and propose a more comprehensive solution for such a problem. Although considering aspects such as user preferences and constraints, their work (or the surveyed papers) does not address any security or privacy issues. Similarly, Berger et. al [30] propose an efficient meeting-location algorithm that considers the time in-between two consecutive meetings. However, all private information about users is public.

In the domain of Secure Multiparty Computation (SMC), several authors have addressed privacy issues related to the computation of the distance between two routes [31] or points [32], [33]. Frikken and Atallah [31] propose SMC protocols for securely computing the distance between a point and a line segment, the distance between two moving points and the distance between two line segments. Due to the fully distributed nature of their protocols, the computational and communication complexities increase significantly with the size of the participants and inputs. Moreover, all parties that are involved in the computation of the mutual routes' distances need to be online and synchronized. Solanas and Martínez-Ballesté [33] present a distributed protocol for securely computing the centroid of a set of points through a privacy homomorphism, in order to protect users' location privacy from an untrusted service (LBS) provider. After the distributed computation of the centroid location, only the LBS provider is able to decrypt this value and deliver the service to the user.

IX. Conclusion and Future Work

Activity management applications are frequently used by people in order to facilitate the planning of their daily duties. Privately establishing common time availabilities is an important task for all participants, and substantial research effort has already been devoted to such a challenge.

In this work, we addressed the complementary problem of efficiently and privately computing the optimal meeting location, and presented two privacy-preserving protocols that solve such problem. To the best of our knowledge, this is the first work that addresses the privacy concerns in optimal meeting-location determination. By means of analytical evaluation and practical implementation on real mobile devices, we showed that our schemes efficiently compute the optimal meeting location and do not reveal any private information. Moreover, our user-study showed that people are concerned about sharing personal location preferences with untrusted parties, which increases the relevance of our research efforts and reinforces the need for further exploration. As part of our future work, we are considering the inclusion of the extended security and optimality criteria that we discussed in this paper, as well as releasing to the public the source code and our implementation, under the GPL license.

References

[1] Enterprise Strategy Group, "Study on the adoption and use of mobile devices." [Online]. Available: http://www.enterprisestrategygroup.com/2010/08/enterprises-are-embracing-mobile-devices/

[2] Microsoft Outlook. [Online]. Available: http://office.microsoft.com/outlook

[3] Apple iCal. [Online]. Available: http://apple.com/ical

[4] Doodle: easy scheduling. [Online]. Available: http://www.doodle.com/

[5] CHIlabs PDA (Personal Digital Assistants) Use Study. [Online]. Available: http://personal.bgsu.edu/~nberg/chilabs/pda.htm

[6] Rendeznew. [Online]. Available: http://rendeznew.com/

[7] a.placebetween.us. [Online]. Available: http://a.placebetween.us/

[8] MeetWays. [Online]. Available: http://www.meetways.com/

[9] Mezzoman. [Online]. Available: http://www.mezzoman.com/

[10] Orange Taxi sharing App. [Online]. Available: http://event.orange.com/default/EN/all/mondial_auto_en/taxi_partage.htm

[11] "Google cheat view," The Sun Newspaper, 2009. [Online]. Available: http://www.thesun.co.uk/sol/homepage/news/article2350771.ece

[12] T. Herlea, J. Claessens, B. Preneel, G. Neven, F. Piessens, and B. De Decker, "On securely scheduling a meeting," in *IFIP/Sec'01*, 2001.

[13] A. Zunino and M. Campo, "Chronos: A multi-agent system for distributed automatic meeting scheduling," *Expert Systems with Applications*, vol. 36, 2009.

[14] R. Wallace and E. Freuder, "Constraint-based reasoning and privacy/efficiency tradeoffs in multi-agent problem solving," *Artificial Intelligence*, vol. 161, 2005.

[15] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-dnf formulas on ciphertexts," in *Theory of Cryptography*, 2005.

[16] Google My Location. [Online]. Available: http://www.google.com/mobile/maps/

[17] UTM coordinate system. [Online]. Available: https://www.e-education.psu.edu/natureofgeoinfo/c2_p21.html

[18] C. Cachin and R. Strobl, "Asynchronous group key exchange with failures," in *ACM PODC '04*, 2004.

[19] C.-H. O. Chen, C.-W. Chen, C. Kuo, Y.-H. Lai, J. M. McCune, A. Studer, A. Perrig, B.-Y. Yang, and T.-C. Wu, "Gangs: Gather, authenticate 'n group securely," in *ACM MobiCom '08*, 2008.

[20] Y.-H. Lin, A. Studer, H.-C. Hsiao, J. M. McCune, K.-H. Wang, M. Krohn, P.-L. Lin, A. Perrig, H.-M. Sun, and B.-Y. Yang, "Spate: Small-group PKI-less authenticated trust establishment," in *MobiSys '09*, 2009.

[21] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, 1978.

[22] O. Goldreich, *Foundations of cryptography: Basic applications*. Cambridge University Press, 2004.

[23] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, 1985.

[24] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," *EUROCRYPT '99*, 1999.

[25] M. Robshaw and Y. Yin, "Elliptic curve cryptosystems," *An RSA Laboratories Technical Note*, 1997.

[26] B. Schoenmakers, "A simple publicly verifiable secret sharing scheme and its application to electronic voting," in *CRYPTO99*, 1999.

[27] J. Lewis, "IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use," *International Journal of Human-Computer Interaction*, vol. 7, 1995.

[28] M. Chignell, A. Quan-Haase, and J. Gwizdka, "The privacy attitudes questionnaire (paq): initial development and validation," in *Human Factors and Ergonomics Society Annual Meeting Proceedings*, 2003.

[29] F. Berger, R. Klein, D. Nussbaum, J.-R. Sack, and J. Yi, "A meeting scheduling problem respecting time and space," *GeoInformatica*, 2009.

[30] P. Santos and H. Vaughn, "Where shall we meet? Proposing optimal locations for meetings," 2007.

[31] K. B. Frikken and M. J. Atallah, "Privacy preserving route planning," in *WPES '04*, 2004.

[32] S.-D. Li and Y.-Q. Dai, "Secure two-party computational geometry," *Journal of Computer Science and Technology*, vol. 20, 2005.

[33] A. Solanas and A. Martínez-Ballesté, "Privacy protection in location-based services through a public-key privacy homomorphism," in *Public Key Infrastructure*, 2007.

Appendix A
Server Advantages

The server advantages are defined in a similar fashion as the user advantages. For instance, the server identifiability advantage $Adv^{IDT}_{LDS}(A)$ of an LDS (executing the POML algorithm A) is defined as follows.

1) Initialization: Challenger privately collects $\mathbb{L} = \{L_i\}_{i=1}^{N}$, where $L_i = (x_i, y_i)$ is the preferred meeting location of user u_i, and $f(L_i)$, $\forall i \in \{1, \ldots, N\}$.
2) POML algorithm: LDS executes the POML algorithm A and computes $f(L_{opt}) = g(f(L_1), \ldots, f(L_N))$. It then sends $f(L_{opt})$ to each user $u_i, \forall i \in \{1, \ldots, N\}$. The LDS is the adversary.
3) LDS chooses a user u_i, $i \in \{1, \ldots, N\}$, and sends i to the challenger.
4) Challenge: Challenger chooses a random $k \in \{1, \ldots, N\}$ and sends L_k to the LDS. The challenge is to correctly guess whether $L_k = L_i$.
5) The LDS sends L_{i*} to the challenger. If the LDS thinks that L_k is the preferred meeting location of user u_i, i.e., if $L_k = L_i$ then the LDS sets $L_{i*} = 1$. If the LDS thinks that L_k is not the preferred meeting location of user u_i, then he sets $L_{i*} = 0$. If $L_{i*} = L_k$ the LDS wins the game, otherwise it loses.

The distance- and coordinate-linkability advantages of any LDS are defined similarly to the respective user advantages.

Appendix B
Proofs

A. Correctness

Given the encrypted set of user-preferred locations $f(L_1), \ldots, f(L_N)$, the proposed POML algorithms compute the pairwise distance between each pair of users d_{ij}, $\forall i, j \in \{1, \ldots, N\}$, according to the schemes of the respective distance computation algorithms. Following the sequence of steps for such computation, one can easily verify that the ElGamal-Paillier based distance computation algorithm computes

$$\begin{aligned} Pai(d_{ij}^2) = Pai(x_i^2) \cdot Pai(-2x_i x_j) \cdot Pai(y_j^2) \cdot \\ Pai(y_i^2) \cdot Pai(-2y_i y_j) \cdot Pai(y_j^2) \\ = Pai(x_i^2 - 2x_i x_j + x_j^2 + y_i^2 - 2y_i y_j + y_j^2) \end{aligned}$$

which is the same result that is achieved by the BGN-based distance algorithm.

After the pairwise distance computations, the POML algorithm computes the masking of these pairwise distances by scaling and shifting operations. The scaling operation is achieved by exponentiating the encrypted element to the power of r_i, where $r_i \in \mathbb{N}_w^*$ is a random integer and r_i^{-1} is its multiplicative inverse. The shifting operation is done by multiplying the encrypted element with the encryption (using the public key of the users) of another random integer s_i privately chosen by the LDS. These two algebraic operations mask the values d_{ij}^2 (within the encrypted elements), such that the true d_{ij}^2 are hidden from the users. Nevertheless, thanks to the homomorphic properties of the encryption schemes, the LDS is still able to remove the masking (after the users have identified the maximum value) and correctly re-mask all maxima, such that each user is able to correctly find the minimum of all maxima.

In the end, each user is able to determine L_{opt} where $opt = \text{argmin}_i \max_j d_{ij}^2$ from the outputs of the POML algorithm, and therefore the POML algorithms are correct.

B. User-privacy

Hereafter we provide sketches of the proofs of user-privacy, after a private execution of the POML algorithm A.

1) Identifiability Advantage: In our submitted work we define the identifiability advantage of an attacker u_a as

$$\begin{aligned} Adv_a^{IDT}(A) = |Pr[(L_j^* = 1 \wedge L_k = L_j) \cup \\ (L_j^* = 0 \wedge L_k \neq L_j)] - 1/(N-1)| \end{aligned}$$

where $Pr[L_j^* = L_k]$ is the probability of user u_a winning the game by correctly answering the challenge, computed over the coinflips of the challenger, and 1/(N-1) is the probability of a random guess over the N possible user-preferred locations. Now, at the end of the POML protocol, the attacker knows L_{opt} and its own preferred location $L_a = (x_a, y_a) \in \mathbb{N}^2$. Assuming that all users other than u_a have executed the protocol correctly, u_a does not know any preferred location L_i, for $i \neq a$. Therefore, the attacker has not gained any additional knowledge from the execution of the POML algorithm with respect to what he already knows by himself and from the output. Hence, the probability $Pr[L_j^* = L_k]$ of him making a correct guess j^* about the preferred meeting location L_k of user u_k equals the probability of a random guess, which is $1/N - 1$. Thus, the identifiability advantage of the attacker u_a is negligible.

2) Distance-Linkability Advantage: The distance-linkability of an attacker u_a is defined as

$$Adv_a^{d-LNK}(A) = |Pr[(d^* = 1] \wedge d_{jk} < s) \vee \\ (d^* = 0 \wedge d_{jk} \geq s)] - \frac{1}{2}|$$

where $Pr[.]$ is the probability of the adversary u_a winning the game by correctly answering the challenge, computed over the coinflips of the challenger, d^* is the guess of the adversary, d_{jk} is the distance between L_j, L_k and s is a parameter chosen by the challenger. In this case, the attacker has to guess whether the distance d_{jk} between two users j, k is greater than s, and clearly if he at some point in the protocol obtains any pairwise distance d_{jk}, his advantage is non-negligible. However, as explained in the correctness proof, each user gets to know only N masked (and anonymized) values of the squares of pairwise distances. With this information, the attacker wants to solve a system of linear equations of the following form:

$$\begin{cases} C_{\sigma_a,\theta_1} & = r_a \cdot d^2_{\sigma_1,\theta_1} + s_a \\ & \vdots \\ C_{\sigma_a,\theta_N} & = r_a \cdot d^2_{\sigma_1,\theta_N} + s_a \end{cases}$$

where C_{ij} is the received masked value of the pairwise distances and r_a, s_a are random integers privately chosen by the LDS. Hence, possessing only the knowledge of his own preferred location and the optimal meeting location, the attacker cannot uniquely solve this system of equation, because it is still under-determined. Therefore, the distance-linkability advantage of u_a is negligible.

3) Coordinate-Linkability Advantage: In order to have non-negligible coordinate-linkability advantage, an attacker u_a needs to have additional information regarding at least one of the two coordinates of any other user's preferred meeting location. As discussed in the identifiability and distance linkability advantage proofs, after a private execution of the POML algorithm A, the attacker does not gain any additional information about any other user's locations. Therefore, not knowing any other user's coordinate, an attacker does not gain any probabilistic advantage on correctly guessing the relationship between their spatial coordinates. Hence, the coordinate-linkability advantage of an attacker u_a is negligible.

C. Server-privacy

All elements that are received and processed by the LDS have previously been encrypted by the users with their common public key. In order to efficiently decrypt such elements, the LDS would need to have access to the private key that has been generated with the public key used for the encryption. In most practical settings, where service providers have a commercial interest in providing a faithful service to their customers, the LDS would not try to maliciously obtain the secret key. Therefore, all the LDS does in the POML algorithm is to obliviously execute algebraic operation on encrypted elements, without knowing the values within the encrypted elements. Hence, the POML algorithms are server-private.