Let's Meet! A participatory-based discovery and rendezvous mobile marketing framework

Lampros Ntalkos [a], Georgios Kambourakis [a], Dimitrios Damopoulos [b,*]

[a] Department of Information and Communication Systems Engineering, University of the Aegean Karlovassi, GR-83200 Samos, Greece
[b] Department of Computer Science, Stevens Institute of Technology, Hoboken, NJ 07030, USA

Keywords:
Smartphones
Mobile app
Mobile marketing
Security
Rendezvous system
Privacy

A B S T R A C T

Modern mobile devices are nowadays powerful enough and can be used toward defining a new channel of communication with potential consumers. This channel is commonly known as mobile marketing and there is already a number of mobile marketing apps, whose aim is to increase the sales of some product or service. In this context, the Let's Meet! framework presented in this paper is essentially a mobile marketing app. The app groups two or more persons, who basically do not know each other, having as sole criterion their common interest in an offer about a product or a service. Its main objective is to bring them together, so that they can purchase and enjoy an offer, which otherwise could not afford. One of the highlights of our proposal is that all sensitive user data are transmitted in a secure manner, and thus confidentiality is preserved. Users' privacy is also given great consideration. This means for example that the exact geographic locations of the users are never shared with others. For user authentication, Let's Meet! supports both a complete anonymous mode and OAuth 2.0. The framework's main objective, which is to bring the users together, is guaranteed by means of a one-time coupon, generated by the OCRA algorithm, while the final face-to-face user group meeting is achieved through Wi-Fi Direct technology. Moreover, the app implements a smart queueing system for increasing its efficiency. Every possible effort is made to maximize both the number of products being sold and the number of users that eventually enjoy an offer. Finally, a user rating system has been adopted, which rewards any user attitude that helps towards improving the framework's competence. The above qualities make Let's Meet! a novel proposal when considering similar works in the literature so far.

Published by Elsevier Ltd.

1. Introduction

Marketing is defined as any activity carried out in order to promote products or services and to approach potential consumers. It is not a new concept at all but throughout the years the methodology and the techniques used change constantly. Actually, it is the available technology that defines the means used to reach hopefully a great mass of consumers. Lots of years ago, newspapers and magazines were the only available ways to do so. Radio and TV appeared next, followed by the Internet. Nowadays, perhaps the most advanced means of reaching users are mobile devices, which combine the speed and penetration of the Internet with the portability and personalization features of the so-called smartphones. In this

* Corresponding author at: Stevens Institute of Technology, Castle Point on Hudson, Hoboken, NJ 07030, USA. Tel.: +1 2018447079.
E-mail address: ddamopou@stevens.edu (D. Damopoulos).

context, according to Kaplan (2012), mobile marketing is any marketing activity which is conducted through a ubiquitous network to which consumers are constantly connected using their personal mobile device. These devices appear to be the most suitable for any marketing approach, since they provide high-speed and constant access to a variety of short or long range wireless data networks, either through Wi-Fi, WiMAX or 3G/4G, and they also tend to be described as personal.

The penetration of mobile devices in our lives is constantly increasing. According to eMarketer.com, by the end of 2014, one out of four people in the world will possess a smartphone. This is equal to 1.75 billion people or 1.75 billion potential consumers, when speaking in terms of marketing. On the other hand, large amount of money is invested in mobile marketing. eMarketer.com also estimates that by the end of 2014, $31,45 billion will have been spent worldwide for placing ads in mobile apps. This amount of money is collected by huge advertising companies, Google and Facebook being the first.

Further, the customization and personalization of mobile ads have proved to be of great importance. Targeted advertisements ensure that the right person receives the right message at the right time (Adam, 2002). To achieve this, user-specific information must be retrieved to generate tailored content. User profiles must be built and include demographics, such as the user's age, gender and occupation, as well as preferences, such as what she likes to eat or drink, what kind of entertainment she enjoys or what kind of places she is used to visit. This is actually long-term information and does not change often. On the other side, the context of the user can also be used to generate content that is most suitable to her. The context is described as short-term and it may be the user's location, the current time or the activity she is engaged into.

Users may identify great value in receiving customized messages, but privacy concerns may also arise when disclosing their personal information. It is therefore up to each user to decide how much information about herself she is willing to disclose and how much valuable the return will be. Privacy is defined as the ability of the individual to control the terms under which personal information is acquired and used (Fraenkel and Westin, 1967; Kambourakis, 2014). When a mobile app user is given the ability to control in detail which information about her is disclosed, then her privacy is being respected and this person feels comfortable to use this app (Loukas et al., 2012). On the contrary, improper handling of user data could result in the discovery of consumer identity and behavior, which may be used for unsolicited marketing or price discrimination (Damopoulos et al., 2013; Kobsa, 2007). Moreover, it is the individuals' own experiences and characteristics that affect their attitude regarding privacy. Any previous privacy invasion experiences, personal innovativeness and coupon proneness play an important role (Xu et al., 2011).

Contribution of this work: We propose a mobile marketing framework, coined Let's Meet!, that puts emphasis on user's privacy. Let's Meet! groups two or more people, who basically do not know each other, having as sole criterion their common interest in an offer about a product or a service. Its main objective is to bring them together, so that they can purchase and enjoy the offer, which otherwise could not afford. So, Let's Meet! collects and publishes offers by different vendors, while the end-users can access them through a mobile app. It thus can be seen as a medium of promoting products or services with the end-users being (almost) constantly connected to this network. The system relies on a client/server architecture, where the server (broker) keeps information about offers and interacts with the users, while the client (a mobile app) enables a user to review the available offers, express her interest in, say, one of them and receive feedback about which other users are also interested in the same offer and which is their approximate location. Eventually, the system makes user groups of appropriate size and guides the group members to a face-to-face contact, so that they can complete the purchase.

A real case scenario would include a published offer, where for every two tickets to a cinema movie, one is given for free. This means that two users are required to purchase one packet of this offer and essentially every user can enjoy a 50% discount. If the movies company decided to give away 10 packets of this offer, then 20 users in total can benefit from this. So, as long as the offer has not expired, a mobile user can find out how many users have claimed it so far and whether the maximum limit of 20 participants has been reached or exceeded. She can then claim the offer herself, and only then she is given access to the approximate geographical locations of the rest of the participants.

The highlights of the proposed framework are:

- User's privacy is well preserved. Among others, the geographical location of a user as seen by peers is relative, no personal data are kept in log files or transmitted elsewhere, all communications are performed over a secure channel. The user can be authenticated by a third party, but she can also stay totally anonymous.
- The system's efficiency is strengthened by implementing a queueing system. A user is encouraged to express her interest in an offer, even when no vacancies exist, and the system will try to include her in a group, if a user of higher priority withdraws her own interest or fails to appear in time at the offer location.
- The system adopts a rating process, based on the users' reliability. Punctual users are rewarded, while misbehaving ones may be penalized in some of their future transactions.
- For an offer to be sold, the group peers need to communicate via Wi-Fi Direct technology (Wi-Fi Alliance, 2013) and reassemble a one-time coupon originally generated by the broker.

Every company that wishes to reach more customers and finally increase its profits could potentially decide to use this framework. In this work, the architecture suggested includes a third-party broker, where offers from all possible producers are collected. So, movies companies, as in the example above, taxi companies, restaurants and cafés, all publish their offers on a single location that a mobile user can browse. But this infrastructure can as well be established in a local area, e.g., within the boundaries of a large shopping mall. In that case, a user visiting this mall could use a mobile app operated by the mall, in order to learn about active offers. This way, customers that do not know each other, can meet for instance over the detergents counter and purchase two pieces by paying only one.

The rest of this paper is structured as follows: The next section outlines the Let's Meet! framework and its workflow. A typical usage scenario is discussed in Section 3. In Section 4, a more technical analysis of the protocol is done. Section 5 addresses security and privacy issues, while Section 6 reports on the memory and energy requirements related to the client app. Section 7 reviews the related literature and makes a comparison with our framework. Finally, in Section 8, we conclude and give some directions for future work.

2. Framework architecture

Let's Meet! comprises of specific entities that interact with each other. First, we describe these entities and the main actions they take part in. Next, we detail on the framework and its workflow, starting from an offer being published until the time where all people interested in that offer meet each other in person so as to complete the purchase. Recall that the main objective of the framework is to get people together, while the actual purchase is out of scope of this work.

2.1. Terminology

Offer	The product or service that is being promoted
Packets	The quantity of the products or services being offered. Packets are always $\geqslant 1$
Producer P	Sells the above product or service and publishes the corresponding offer
User U	Is interested in the offer and claims it
Broker B	An intermediate entity (server), that matches users based on common claimed offers. This entity comprises the server-side in the Let's Meet! architecture and thus in the following the terms Broker and server are used indistinctly
Fellows	A set of users that are currently interested in a common offer. The more offers a user claims, the bigger this set becomes. Also, this set is time-dependent since users' interest may change over time
Peers	The users that are qualified by the system to purchase one packet of a given offer. Note that peers are always a subset of fellows
Group	The peers comprise one (1) group. It is a fixed team of people that can move on and make the purchase. The number of groups cannot be greater than the number of packets
Group master	The first user that becomes member of a given group
(one-time) Coupon	An 8-digit numeric code, which is used as a guarantee that the group peers have met each other in person
Coupon issuer	The broker B issues the coupon, splits it to as many parts as the peers and sends each of them a corresponding coupon part
Coupon redeemer	The broker B validates the coupon, which is reassembled and sent back to the broker by the group master

2.2. Let's Meet! framework

The most important module of the framework is the Broker B. This entity plays multiple roles. It interacts with both the producers and the users. Also, it communicates with a database to manage offers, users and users' interest in offers. It finally includes a groups agent, a smart module that is being executed at suitable times and decides which users are qualified to join a group. An overview of the framework is depicted in Fig. 1.

More specifically, the Broker comprises four modules. The Profiles Manager (A1) is responsible for managing the users' profiles. The Offers Manager (A2) manages the offers and the users' interest in offers. The User Serving Module (A3) serves each user that is currently connected to the system. The Groups Agent (A4) is responsible for forming the groups. The data being managed by the system can be categorized to Users Profile Data (DB1) for data related to the users and to Offers Data (DB2) for offers-related data.

2.2.1. Let's Meet! workflow

Now let's see the main logical steps carried out throughout an offer's lifecycle.

Producer publishes Offer. At first, a producer decides to promote one of his products or services and publishes a suitable offer. The Offers Manager (A2) accepts the offer details, i.e., title, description, number of packets, starting and expiration date and offer location, and updates the offers data (DB2). From now on, the offer is online and any user can review it through her mobile app as long as the offer is active, i.e., the offer's expiration date has not passed by. An offer that starts some time in the future is also considered to be online and can still be claimed by interested users.

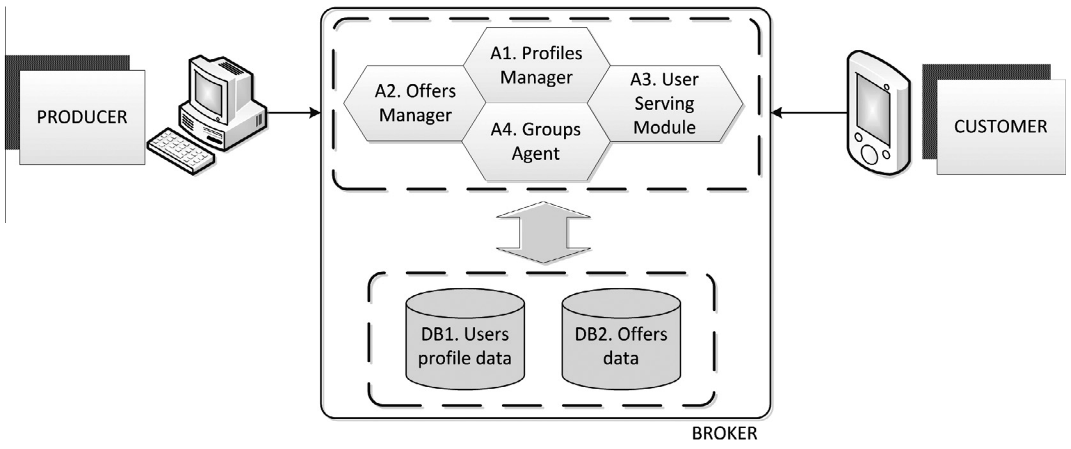


Fig. 1. An overview of Let's Meet! framework.

Customer claims Offer. Suppose a user is interested in the offer above, so she claims it. Consequently, the module that serves the current user takes two actions. If this user is a newcomer for the system, a new entry is created in DB1 by the Profiles Manager (A1). Then, the Offers Manager (A2) keeps in DB2 the user's interest in this offer.

Customer pings Broker. Naturally, the user will have to appear physically at the offer location, if the offer's starting time is about to come or if it has already passed by. In either case, the user's device starts pinging the broker periodically, to report itself, as depicted in Fig. 2. This procedure starts as soon as the user claims the offer, no matter if the offer has started or not. With every ping, the device's exact location is being sent. The broker finds the user's fellows, i.e., other users interested in common offers, and sends them an approximate value of the location of the initial user. This is done by randomly obfuscating the location in a radius of 50 m. This way every user knows approximately where her fellows are located, but not their exact position. The periodic report of the device helps also the system keep track of online users. This procedure may last for a while or for quite a long, depending on the offer's details. If, for example, the offer expires in a long time from the moment the user claimed it, then the above procedure will repeat itself many times, as long as the user remains online.

Broker fixes Group. At the time a user finds herself in the vicinity of an offer, then the Groups Agent (A4) is triggered. This agent tries to include this user in some group and to finalize it, if the requirements are met. As depicted in Fig. 3, the Offers Manager retrieves from DB2 the priority and the Time-of-Appearance of all fellows. Based on these two parameters, the peers are chosen and returned to the Groups Agent. Now, the agent knows how many peers comprise the group, so it computes a one-time coupon, splits it to respective parts and sends them to the peers. The first peer included in the group is considered the *master* and has a specific role: she contacts all other peers with a direct communication channel, using Wi-Fi Direct technology, and collects all coupon parts. Then, she reassembles the coupon and sends it back to the Broker. The latter checks that the coupon received corresponds to the group for which it was issued, that it has not yet been redeemed, and that it matches the one originally created. This agent (A4) is also triggered when the offer expires and not

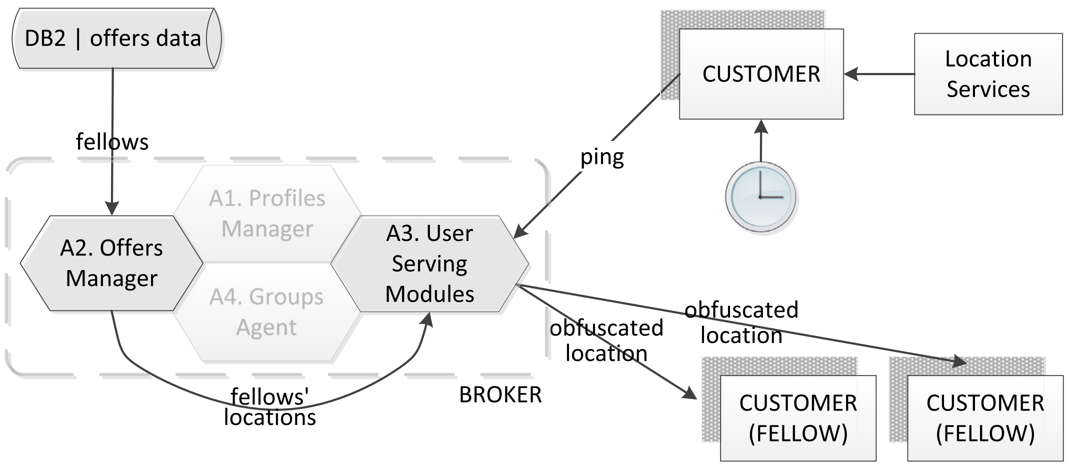


Fig. 2. Customer pings Broker.

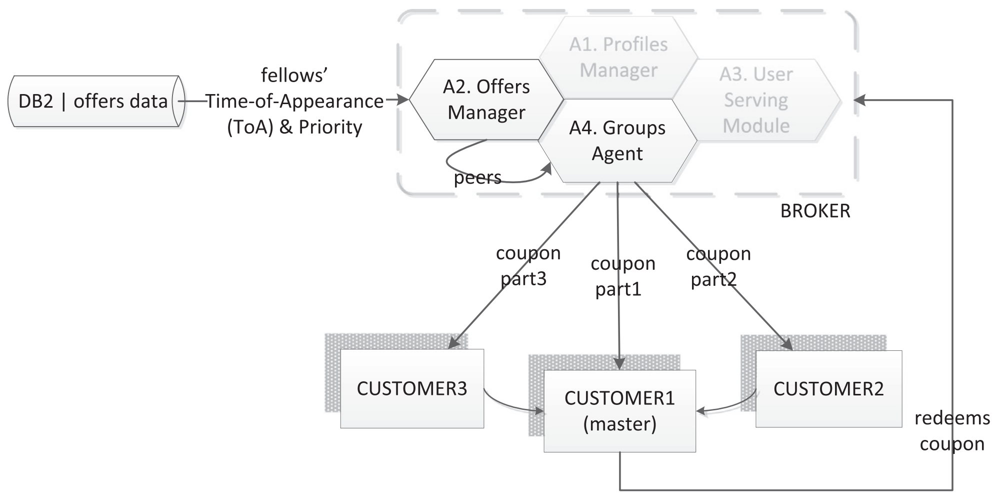


Fig. 3. Broker fixes Group.

all packets are given away. It then looks for users in queue or waits for "delayed" users, in an effort to maximize the packets sold.

3. Usage scenario

To get a clearer understanding of the framework capabilities, one needs to study the Let's Meet! platform from both end-user's and server's viewpoint, having the offer's lifecycle in mind.

In the following sections it will get clearer how a user can:

- preview the available offers and claim an interesting offer or "un-claim" it;
- have knowledge about who her fellows currently are and which is their approximate location;
- have knowledge about offers' status, i.e., whether an offer is full or not, whether the user is in queue or not, how many vacancies still exist, and so forth;
- move towards the offer location.

The finalizing steps are carried out at the Broker-side. The latter:

- keeps track of online users;
- records users' claim priorities and time of appearance;
- decides which users join a group using the above knowledge as well as information about the offers (packets available and users per packet);
- assures that group members meet each other (by means of a one-time coupon) and rates users in terms of their behavior after claiming an offer.

For a better explanation of the functionalities of the framework, indicative screenshots are included and a discussion will be based upon each of them.

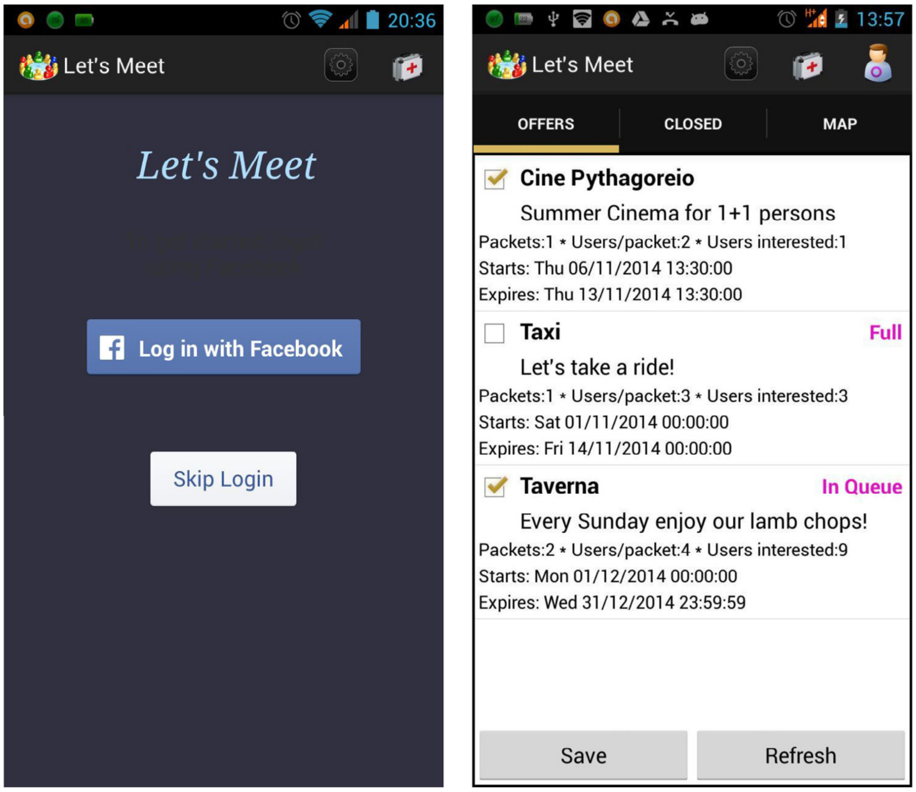
3.1. User interface

3.1.1. Login screen

On the client side, the user is presented with the app logo and the capability of authenticating herself via Facebook. The authentication mechanism will be explained in detail in Section 5. The user has also the option of skipping login, in case she wishes to retain her anonymity and revealing no personal information about herself to the system.

3.1.2. Offers tab

As shown in Fig. 4, after logging in (or skipping login) the user is presented with the app's main screen. Three tabs are available. In the first one, all active offers are visible, ordered by expiration date. The ones that expire sooner appear at the top, so as to attract the user's attention. Keep in mind that only if at least one packet of the offer is still available, will the offer appear on this screen. For each offer, one can preview the title, description, starting and expiration date, total



Cine Pythagoreio
Summer Cinema for 1+1 persons
Packets:1 * Users/packet:2 * Users interested:1
Starts: Thu 06/11/2014 13:30:00
Expires: Thu 13/11/2014 13:30:00

Taxi Full
Let's take a ride!
Packets:1 * Users/packet:3 * Users interested:3
Starts: Sat 01/11/2014 00:00:00
Expires: Fri 14/11/2014 00:00:00

Taverna In Queue
Every Sunday enjoy our lamb chops!
Packets:2 * Users/packet:4 * Users interested:9
Starts: Mon 01/12/2014 00:00:00
Expires: Wed 31/12/2014 23:59:59

(a) Let's Meet! Login Screen (b) Let's Meet! Offers Screen

Fig. 4. Let's Meet! welcome and offers screens.

number of packets and total number of users that currently claim it. Also, a flag appears to indicate that all available packets are reserved, so this offer is full. Finally, in the case of a claimed offer, another flag indicates whether the user is a priority one or she has been placed in the queue. In the latter case, it's up to her to decide whether she will move toward the offer location, hoping that some users of higher priority will regret and withdraw their interest in the same offer or at least fail to appear at the offer location.

The check box at the left side is used to claim or "un-claim" the offer. It is considered a good practice for a user to "un-claim" a no-more interesting offer, since this way she avoids receiving a negative rating (see Section 3.2.4) and also encourages other users to solicit this new vacancy. However, by doing so, she loses her priority and if she regrets and re-claims the offer, she will be given a new one. The save button must be hit for the selections to be kept to the DB, but any time the app goes to background an automatic saving happens too.

In the example of Fig. 4b, the "Taxi" offer requires 3 users and this is the number of users already interested. This is why the *Full* flag is on. The user of the app has claimed "Cine Pythagoreio" and "Taverna". However, Taverna was already full, before the user claimed it. One can see that for 2 available packets, with each packet requiring 4 users, only 8 users totally could receive priority. All the others would be put in queue (as in this example). That is why the user of the app sees the *Queue* flag on, while the rest 8 users do not. This user could choose to, say, walk or drive to the offer location, hoping that someone would withdraw, but in the following examples, she chooses to un-claim the offer.

The user has also the capability of refreshing the data being viewed. Let's recall that each offer view provides detailed information, that can be either offer-specific (e.g., title, description, expiration date) or global and can be affected by other users' activity. This global information is the total number of users that currently claim the offer and the flags we just talked about. If the app were to display live data about the offers, then this would create a noticeable burden on the server, which would scale in case the system acquired high traffic and served a large number of concurrent users. Even worse, at the client side, high mobile data charges would apply and the battery of the mobile device would drain faster. A simple solution to this situation includes a button that when clicked refreshes the offers' list. The app can also auto-refresh the offers' list, based on a preference that the user sets. By doing so, it is always probable that the data the user sees are out-of-date. It is also probable, that a user claims an offer that seems to have vacancies, but actually it has not (because someone else claimed it in the meanwhile). As a result this user is placed in queue.

So it is the user's decision as to what is best for her. As already pointed out, frequent updates keep the offers' list fresh and provides the user with a better picture of the overall system, but have a negative effect on battery consumption and mobile data usage.

3.1.3. Closed offers tab

As characteristically shown in Fig. 5a, this tab enables the user to preview all closed offers. Closed offers are all those for which the user successfully managed to join a group. The title, description and redemption timestamp are visible. The redemption timestamp is the time at which the master of the group successfully validated the coupon to the server, as already described in Fig. 3.

3.1.4. Map tab

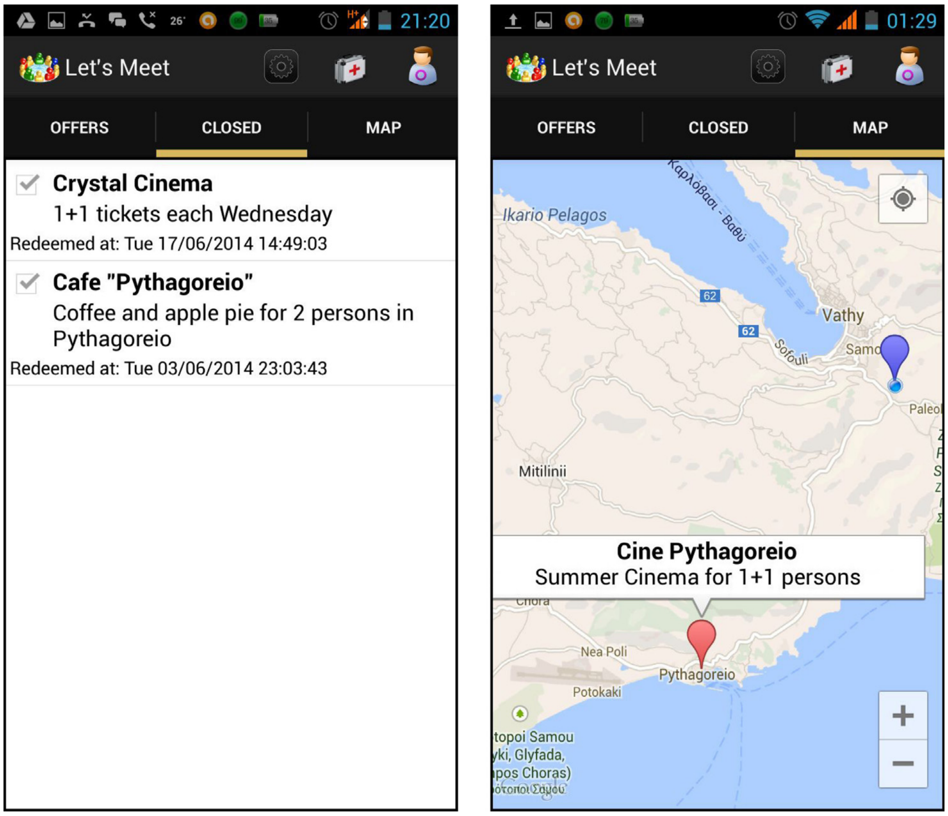
The map tab on the other hand includes a Google map of the area where all claimed offers and all fellows are visible. As we observe in Fig. 5b,blue markers are used for users and red markers for offers. The contents and the zoom level of the map change dynamically when the user alters her claims, so as all necessary feedback is given to the user. When a user marker is tapped, the user rating appears and when an offer marker is tapped, a short description and the offer's expiration date appear.

3.2. Server-side actions

3.2.1. Location updates

This process is supported by the periodic report of each online device to the server. In Section 2 we discussed that an online device pings the broker in specific time intervals. This way the server:

- always is aware of who is online;
- obfuscates any location information it receives before propagating it to the user's fellows. Thus, fellows never know during this phase the exact location of each other;
- tracks which user is close enough to a claimed offer, so as to initiate the appropriate actions to form a group.



Crystal Cinema
1+1 tickets each Wednesday
Redeemed at: Tue 17/06/2014 14:49:03

Cafe "Pythagoreio"
Coffee and apple pie for 2 persons in Pythagoreio
Redeemed at: Tue 03/06/2014 23:03:43

Cine Pythagoreio
Summer Cinema for 1+1 persons

(a) Let's Meet! Closed Offers Screen (b) Let's Meet! Map Screen

Fig. 5. Let's Meet! closed offers and map screens.

As it is discussed in detail in Section 5, the server receives an accurate location from a device, makes decisions over it, obfuscates it and propagates (multicasts) it to fellows. In our opinion, this is the best approach regarding the user's location privacy, because the server needs the exact locations in order to function properly, but the users must not know with accuracy where the others are located.

3.2.2. Joining groups

As users arrive at the offer location, the server constantly examines several parameters for deciding whether they should be included in a so-called pending group or not or they should wait for a while. A pending group is a temporary group for which the necessary number of users have not yet gathered. Actually, it is unknown if this will ever happen and depends on the claims as well as on the users reaching the offer location. Supposing that an offer defines as n the size of its groups, then either $1,2$ or $n-1$ users comprise a pending group and all of them are not yet qualified to enjoy the offer until the nth user appears. Then the pending group is fixed and converted to a "real" group. Only one pending group can exist at a time.

In our example, exemplified in Fig. 6, the server watches "Cine-Pythagoreio". The pieces of information it uses to make decisions are:

- the offer's packets and the necessary number of users per packet;
- the arriving user's claim priority and time of appearance;
- the offer's expiration date.

The offer's packets and necessary number of users per packet will determine the maximum number of groups that can be formed. The arriving user's claim priority and time of appearance are used to categorize the user (with respect to this offer). Six categories exist based on the values of these two parameters, but for the current phase the system uses only the first four. So, either a user is a priority one or not. This value is set at the time the user claims the offer and with respect to other fellows that may have solicited the same offer before or after her. Also, the user either has appeared at the offer location in time or delayed or never. This value is set when the user's device reports a location that is very close to the offer's one and so the user is considered to have reached a rendezvous point. The six possible outcomes are depicted in Table 1.

So the Broker, and more specifically the Groups Agent, takes the following steps:

1. Initially, it looks for category A users and tries to fill as more groups as possible with this kind of users. Keep in mind that it is possible that all groups can get filled with A users. This is true by definition of category A: if the necessary number of users for all packets claim the offer and then appear at the offer location, then all groups are gracefully fixed. This step runs before the offer expires and for every newcomer.
2. If step 1 fails to fill all groups and the offer expires, then category B users are examined. Remember that these are users in the queue, but who have appeared in time, just waiting and hoping to acquire the offer. So, now, B users are used to assemble the remaining groups. The B users are examined ordered by priority: the earlier a user claimed, the highest priority she has received.

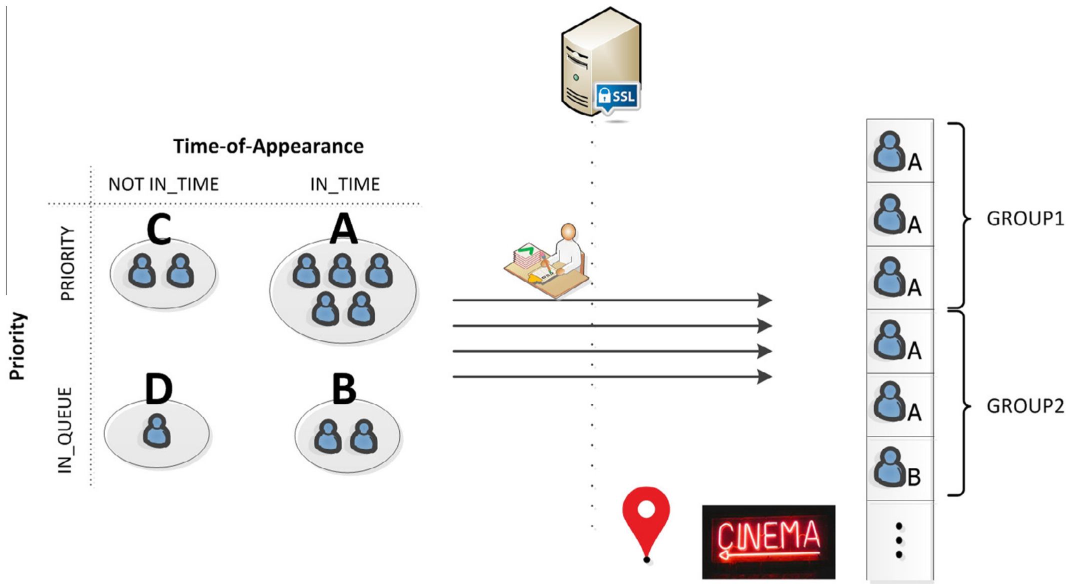


Fig. 6. Joining groups.

Table 1
Users categorization according to priority and time of appearance.

	Time of Appearance		
	Never	Not in time	In time
Priority	E	C	A
In queue	F	D	B

3. If step 2 fails to populate all groups, then the agent waits for a specified amount of time before examining category C and D users, i.e., delayed ones. The waiting time is offer-specific and set by the producer when publishing the offer. When this time amount elapses, C and D users, ordered by time of appearance, are used to populate any remaining groups.

If the offer is still not full, then it is explicitly closed by the agent. As observed, the agent is triggered (a) when a user arrives, (b) when the offer (if not yet full) expires, and lastly (c) when the extension time elapses.

3.2.3. Generation of a coupon

For each fixed group, the broker issues a coupon, which is an 8-digit numeric string. The security and the robustness of the algorithm used for the coupon generation will be discussed in Section 5. The broker counts the group members and splits the coupon to as many parts as necessary. Each part is sent to a member of the group. The first user that became member of the group with the process described in the previous paragraph is considered the group master. Since this user is the first to have appeared at the offer location, every next member should come in direct contact with her and ***not*** through the server. The technology used for this contact is Wi-Fi Direct and assures that only when the two users are physically close enough, will this contact be feasible. So, the new user "says hi" to the master and passes her coupon part, acquired previously by the server. In the same way, all other members give their coupon parts to the master and at the end the master is able to reassemble the coupon and redeem it to the server. The server checks it against the one that was initially generated and if they match, then it is assumed that the users have met each other. The above procedure is depicted in Fig. 7.

3.2.4. Rating of users

This is the final event in an offer's lifecycle. The largest possible number of groups was assembled, depending on how many users claimed the offer and how many actually appeared at the offer's location (in time or not). Some users managed to be part of a group, while some others did not, either from their own fault or because other users were faster. In either case the broker rates all of them and may take according actions in the future, based on each user's personal rate. In the rating

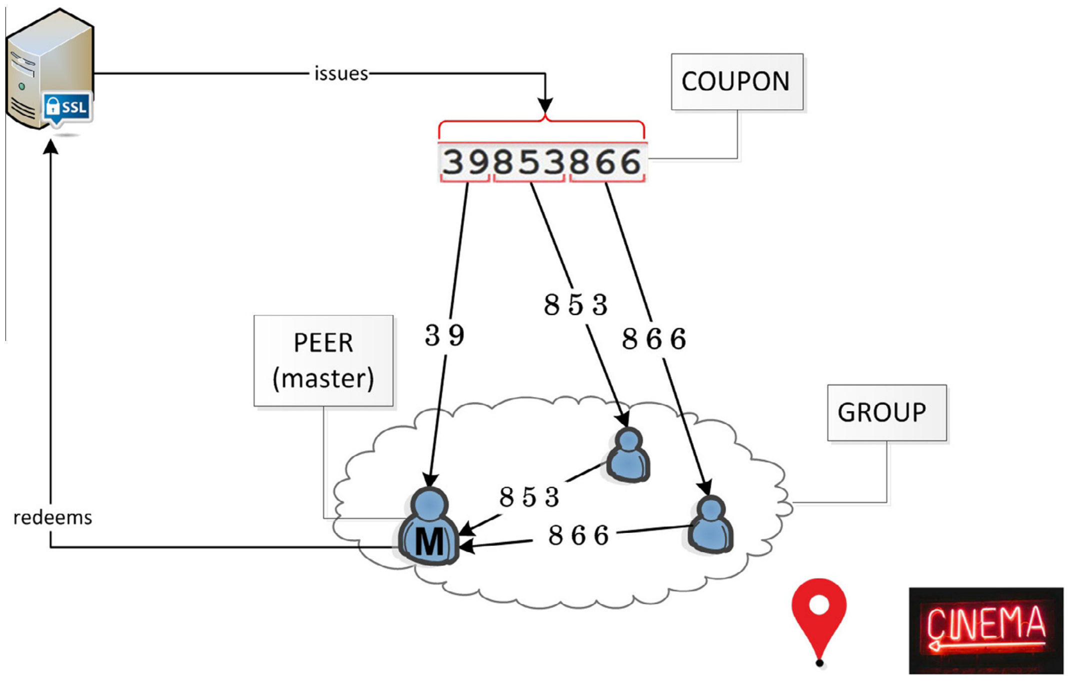


Fig. 7. Coupon generation and redemption.

Table 2
Users rating.

	User category					
	A	B	C	D	E	F
In Group	+	+	+	+	+	+
No Group		0	–	0	–	0

process, the rest two categories that a user can fall into must be taken into consideration. These are categories E and F, as shown in Table 1. Category E is consisted of priority users that never appeared, while category F is consisted of users in queue that decided not to appear. The rating algorithm is summarized in Table 2.

As observed from the table, a positive point is given to every user that was finally part of a group, no matter which category she falls into. On the other hand, when a user doesn't manage to be part of a group, then:

- if it is about a user of category B, D or F, nothing happens. These are all users in queue and had no obligation to appear at the offer location. So, the system does not "punish" them for actually not getting there.
- if it pertains to a user of category C or E, then this user receives a negative point. This is a user that reserved an available place and was actually inconsistent with her rendezvous. Even worse, this behavior deprived other users from enjoying the offer or possibly discouraged them from trying to get at the offer location, and therefore should be "punished".

As we observe, it is impossible for a user of category A not to be part of a group. This stands true by definition and that is why the table has no value for this case.

4. Protocol

In this section, a more technical discussion of the Let's Meet! communication client/server protocol is done, which is the heart of this framework. All the business logic of the framework is incorporated in the protocol transactions. Of great importance is the protocol implementation especially at the server-side, since the server interacts with multiple devices at the same time and these devices claim to access common resources. The server methods that grant access to these resources must be thread-safe.

4.1. Entities

The entities that appear in the procedures of the protocol are:

- the Let's Meet! Transport Layer Security (TLS)-enabled server;
- the database server;
- a smartphone, representing any location-aware mobile device (e.g., by means of a GPS receiver);
- a peer smartphone and;
- a Location Service Provider.

4.2. Notations

Specific conventions have been made regarding the notations used throughout the diagrams.

- Messages sent between the Let's Meet! server and the mobile devices are shown in UPPERCASE. Any parameters passed with these messages are included in parentheses.
- Actions taken by the server or the mobile device are shown in *UPPERCASE ITALICS*, surrounded by a rectangle. Some parameters in parentheses are sometimes included, to better clarify the action taken.
- Decisions made by the server are depicted with the corresponding decision symbol with the two possible outcomes.
- Interactions between the server and the database are shown with bi-directional arrows connecting these two entities. Sometimes, the method called is written in mixed case above, following standard Java naming conventions.
- A serial number of the main steps of the whole protocol procedure are included in small circles at the most-left side of the diagram, for a better understanding of the steps followed and when referenced in the description text.

4.3. Procedures

4.3.1. Connection

This procedure is executed each time a mobile device wishes to connect to the server. The mobile device requests a connection to the server by sending a **REQUEST_CONNECTION** message. The International Mobile Station Equipment Identity

(IMEI) number of the device acts as a userID and identifies the device at the server and the DB. For security reasons, though, it is not the IMEI number that is included in this message, but the outcome of a cryptographic hash function of it. This topic is discussed in more detail in Section 5.

The server is always aware of the devices connected. It maintains a data structure with all userIDs and other user-specific information. So, the server is in position to decide whether this specific device is already connected. If true, a **CONNECTION_REFUSED** message is sent back. Otherwise, a **CONNECTION_SETUP** message is sent, along with initialization parameters. These can be either global or user-specific parameters that are saved in the DB, so that they can be globally updated, and accessed by the server when this procedure runs.

With respect to our implemented prototype, every message exchanged between the server and the mobile device is encapsulated in a *LetsMeetData* Java object. This is true for any of the procedures described in the following as well.

4.3.2. Retrieve offers and user's selections

In this procedure, the mobile device retrieves a fresh list of all available offers, i.e., offers that are currently active. The server remembers the user's previous selections and pre-checks these specific offers, if any. So, all user preferences are persisted across app or device restarts or after any Internet connectivity failures. The user is able from this screen either to claim an offer or to "un-claim" it, in case she changes her mind.

The mobile device sends a **GET_OFFER** message, with the hashed IMEI included. The server calls the *getSelectableOffers()* method of the DB module and retrieves a detailed list of all active offers. The userID is needed to retrieve the user's previous selections. The results are wrapped in a **DETAILED_OFFERS** message and sent to the mobile device. The device renders this offers' updated list both on the offers and on the map screens.

As discussed in Section 3, there is a specific refresh policy applied. Bear in mind that the offers list is not "live" but is auto-refreshed every x minutes, according to the user's settings, or can be refreshed with the click of a button. In this way, an excessive battery drain and any unexpected mobile data charges, say, in case of a 3G data connection, are avoided, while the user is always aware that she is not in front of an "always-updated" list.

4.3.3. Save offers

In this procedure, the user's interest in offers is saved to the DB. This happens each time the user hits the save button, but as already mentioned, it is also automatically triggered every time the app goes to the background, since it is generally unknown whether the app will return to the foreground before the device is switched off or before Android OS decides to kill some apps for saving resources.

The mobile device sends a **SAVE_CLAIMED_OFFERS** message to the server. This message carries the device hashed IMEI and a list of all offerIDs along with their check status. The check status field indicates the offer's status in respect with its previous value. The possible values are: (i) *checked*, if the user claimed a previously unclaimed offer, (ii) *unchecked*, if the user unclaimed a previously claimed offer, and (iii) *the_same*, in case the user made no change about this offer. The server uses this information to decide what changes must be carried out in the DB and calls the *saveOffers()* method of the DB module. The user is inserted in the database if she is a new one and the user's selections about the offers are properly updated. Any previously checked offers that are now unchecked are removed from the DB, while any newly checked offers are inserted, with respect to the priorities of users also interested in the same offers. Unclaiming an offer results in the user's priority getting lost. If the user does this by mistake, she may be given a much lower priority when she re-claims.

The DB module responds back to the server and the server responds back to the device, depending on the success or failure of the saving process. If an error occurred, a **CLAIMED_OFFERS_NOT_SAVED** message is sent back. Otherwise a **CLAIMED_OFFERS_SAVED** message is sent and the offers that were updated are included. The mobile device uses this last information to efficiently update only the interface's part that has actually changed.

4.3.4. Location updates

The mobile device receives periodically its location from a geolocalization service (Google Location Services in our case). Within specified time intervals the device reports its location to the server by sending a LOCATION message. The first step that the server takes is to inform other users about the new location received. It queries the DB to retrieve all users that have claimed common offers with the current user. This set of users is called "fellows", as already mentioned, and can change any time. The server obfuscates the user's location by changing it randomly in a 50 m radius circle and wraps it in another LOCATION message which is pushed to all fellows, if of course they are online. Upon receipt, all fellows update the location of the user on the map screen of their app. The previously discussed steps are illustrated in Fig. 8.

The next step for the server is to find out how close to claimed offers the user is and to act accordingly. So, for every offer o that the user has claimed, the following procedure is done: the distance d of the user from offer o is calculated. If the distance $d(o)$ is less than 50 m, which is the proximity limit, then the user is considered to have appeared. The DB saves the appearance time and also categorizes the user based on the time of appearance and the priority she has got when claiming. These categories were discussed in detail in Section 3 and are summarized in Table 1.

For the procedure discussed here and described in Fig. 9, only category A users are being processed. All others will be managed by a server thread that will be described later in this section. So, if the user is of category A, she immediately joins a group. The device then receives a **USER_WAITS_FOR_PEERS** message, indicating that the user should wait for more peers

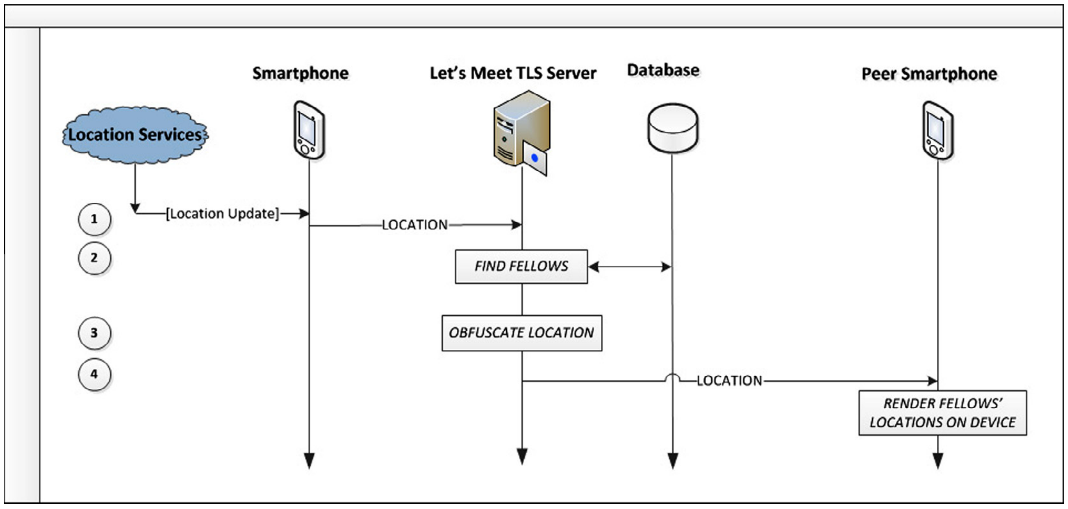


Fig. 8. Location updates.

to appear for the group to be fixed. The USER JOINS PENDING GROUP action is further analyzed in the next procedure, where the users of the other categories are also handled.

4.3.5. Fill expired offers thread

The server periodically gets informed about the offers that have just expired and are not filled yet by category A users. This is done by calling the *getExpiredOffers()* method of the DB module. This is the time where users of categories B, C and D can be used to fix as many groups as possible. If the server manages to fix all groups, then the offer will be considered full. Fig. 10 is used to exemplify this procedure.

Initially, the server queries the DB to find category B users, i.e., those that appeared in time but had claimed the offer after the maximum number of clients was reached, so they were put in queue. These users are ordered by priority and one-by-one join a pending group, as long as the offer does not get full in the meanwhile. Each user that joins a pending group receives a **USER_WAITS_FOR_PEERS** message, as described in the previous procedure.

If the offer is still not full, the server waits for some extra time in order to give one more chance to delayed users. When this extra time passes by, the server queries the DB to find category C and D users, i.e., those that appeared after the offer had expired. The priority of those users, if any, is not taken into consideration any more. The users are ordered by the time of appearance and once again are used one-by-one to "feed" the groups. Each of them receives the **USER_WAITS_FOR_PEERS** message.

If the offer is still not full, then the server finds the users that are part of a pending group and sends them a **USER_IN_NO_GROUP** message. These particular users were unlucky since not enough members were gathered to fix the group. The offer is then explicitly closed. The last step is to rate all users that have not yet been rated. More details about this step will be found in the next paragraph, since this action is run again in the USER JOINS PENDING GROUP action.

Now let's describe the USER JOINS PENDING GROUP action, which we met several times in the previous diagrams. Remember that this action is run when a category A user joins a pending group, but also when other users join a pending group due to the offer not being full yet.

First off, the user is assigned a groupID. This is either the groupID of an already existing pending group or a brand new groupID created by the DB. The user is then rated and actually gets a positive point, since she is qualified to enter a group. Even if this group is never converted to a real one, this will not be the user's fault. The number of members of this pending group are now compared to the number of users required by a group. If they are equal, then the group is fixed. All these "pending" users are now considered members of the fixed group and a One-Time-Coupon (OTC) is generated. The latter is then split into as many pieces as the number of its members. The current user and all other group members receive a **USER_IN_GROUP** message, each one containing a coupon part. The offer's available packets are decreased by one. If there is none left, the offer is explicitly closed.

When the offer is closed, the system rates all those users that have not yet received any rating with regard to this offer. Since the USER JOINS PENDING GROUP action rates all users that became part of a group, either pending or a fixed one, then this step must find out and handle all the remaining users. The rating algorithm described in Table 2 is used for this purpose.

4.3.6. Coupon exchange and validation

The users that are part of a fixed group have just acquired a coupon part, as described above. Everyone of them communicates via a WiFi-Direct channel with the group master and sends a **PEER_SENDS_COUPON** message including the

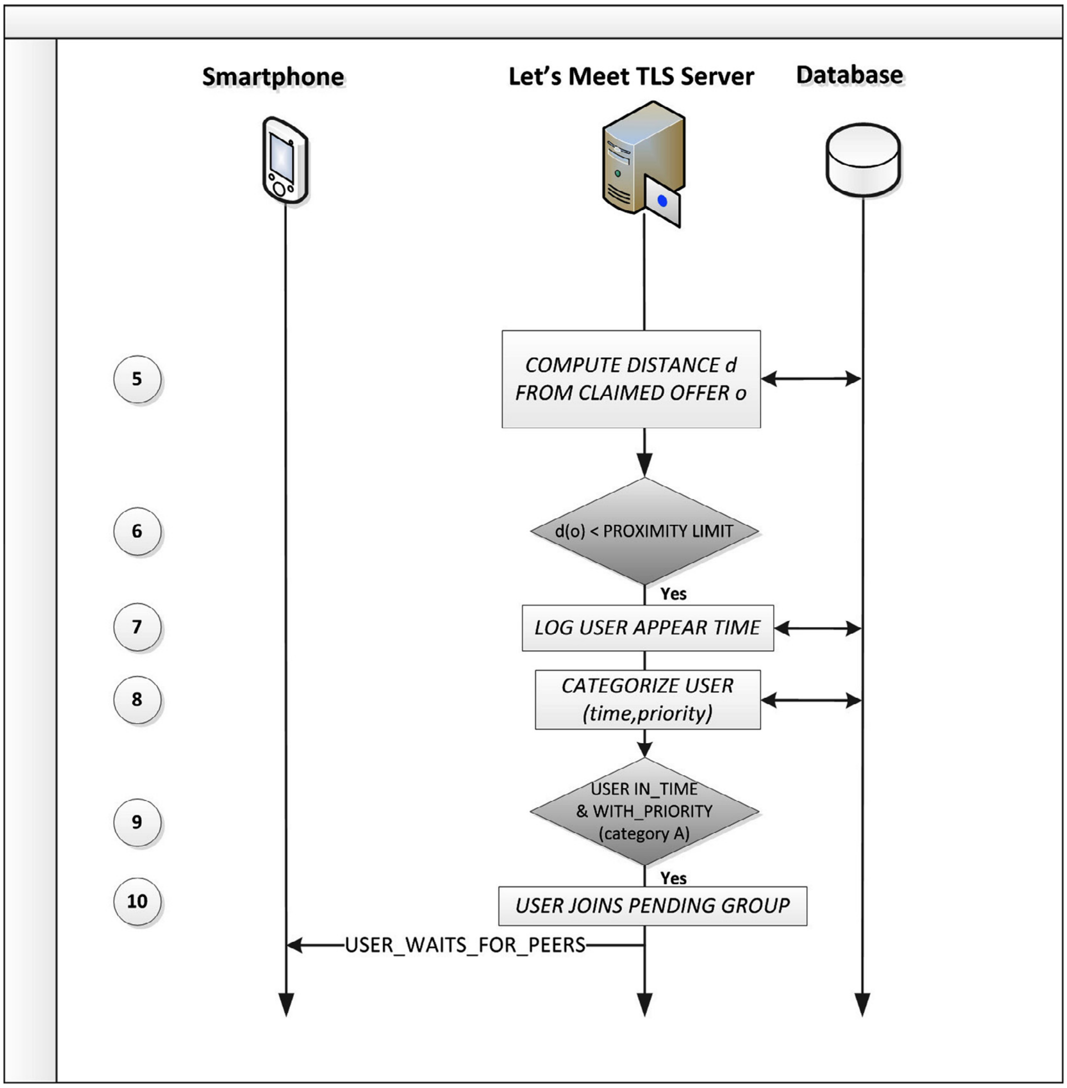


Fig. 9. Location updates - compute users' proximity from a claimed offer.

corresponding coupon part. The master, after she has collected all parts, reassembles the coupon and encapsulates it in a **MASTER_SUBMITS_COUPON** message. The server validates it. More specifically, it checks whether the coupon received equals to the one originally created, whether it corresponds to the group for which it was issued and whether it has not yet been redeemed. If the coupon passes all these tests, it is considered valid and a **COUPON_REDEMPTION_OK** message is sent back to the master and all the other group peers. The group members update their user interface. Otherwise, a **COUPON_REDEMPTION_FAILED** message is returned. This situation is depicted in Fig. 11.

Bear in mind that the actual purchase is out-of-scope of this work. An indicative way, though, for the purchase to be completed would be the Broker informing the Producer about the ID of the offer being sold and sending him the OTC number. The group master could present herself in person to the Producer and show him the OTC. The Producer would then match the two numbers and proceed with the purchase.

4.4. Device-to-device communication

Throughout this text, the term Wi-Fi Direct (Wi-Fi Alliance, 2013) appears to denote the technology used for the mobile devices to communicate to each other during the last procedure of the protocol, as described in Section 4.3.6. Wi-Fi Direct enables mobile devices to connect directly, thus making it simple to, say, exchange data without the need of a router or any kind of a hotspot. A one-to-one connection is feasible, as well as a group of several devices. In our case of course, successive one-to-one connections happen, each time a group member reaches the offer location and contacts with the group master.

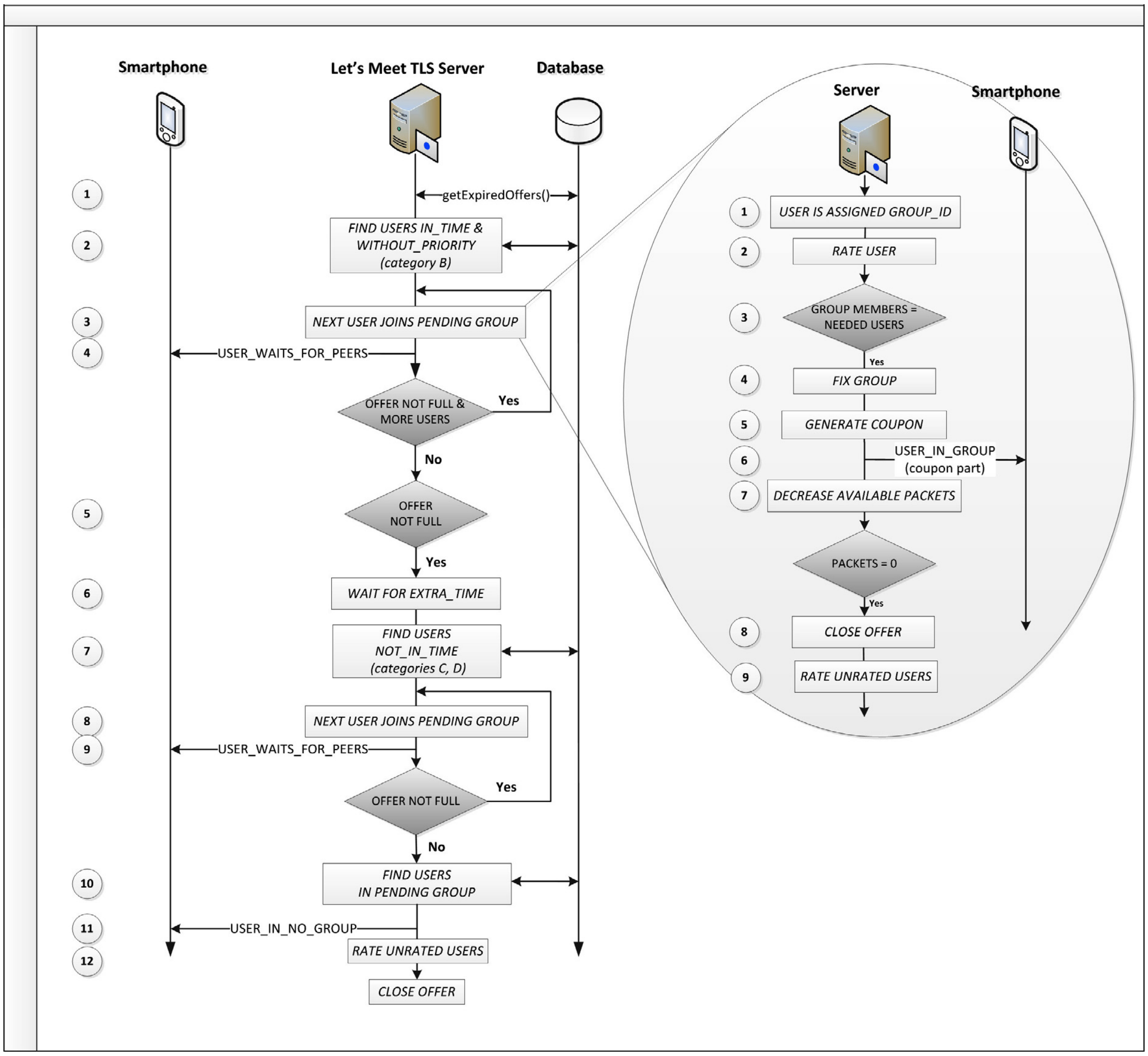

Smartphone
Let's Meet TLS Server
Database
getExpiredOffers()
FIND USERS IN_TIME & WITHOUT_PRIORITY (category B)
NEXT USER JOINS PENDING GROUP
USER_WAITS_FOR_PEERS
OFFER NOT FULL & MORE USERS
Yes
No
OFFER NOT FULL
WAIT FOR EXTRA_TIME
FIND USERS NOT_IN_TIME (categories C, D)
OFFER NOT FULL
FIND USERS IN PENDING GROUP
USER_IN_NO_GROUP
RATE UNRATED USERS
CLOSE OFFER
Server
USER IS ASSIGNED GROUP_ID
RATE USER
GROUP MEMBERS = NEEDED USERS
FIX GROUP
GENERATE COUPON
USER_IN_GROUP (coupon part)
DECREASE AVAILABLE PACKETS
PACKETS = 0
CLOSE OFFER
RATE UNRATED USERS

Fig. 10. Fill expired offers.

Therefore, this technology is used as a means of in-door localization to actually guarantee that the two persons are physically close one another and may even have eye contact.

One could argue, though, that other technologies still exist that offer this kind of functionality, the most “tough” competitor being Bluetooth. Hence, a brief comparison of these two communication protocols is provided in the following to better explain why the one was selected over the other:

Range: The maximum distance of a Wi-Fi Direct communication is almost 200 m, i.e., double than the range of Bluetooth. In indoor environments, these numbers diminish, since the power of the electromagnetic signal loses part of its power when traveling through walls, glasses and other obstacles. So the advantage of Wi-Fi Direct is obvious here. Our framework needs the maximum range available, in order to guarantee that two devices can “see” each other when they are close one another.

Popularity: Wi-Fi Direct is becoming more and more popular in mobile devices. As far as Android-powered devices are concerned, Wi-Fi Direct is a feature available at Android 4+ versions. As of September 2014 (Android Developers, 2014), almost 88% of Android-powered devices use Android version 4 or above, meaning that their users can access to Wi-Fi Direct functionality, as long as the integrated network circuits (wireless chip) of the device are Wi-Fi P2P CERTIFIED (Wi-Fi Alliance, 2013). Bluetooth, on the other hand, is a protocol existing for years and it is unlikely for some users not to have been acquainted with it or for a device not to have been equipped with the appropriate hardware. Due to the popularity of the protocol, on December 2014 the new Bluetooth Core Specification (version 4.2) was released aiming to secure and improve the Bluetooth technology for the Internet of Things (IoT) (org/en-us/specification/adopted specifications, 2014).

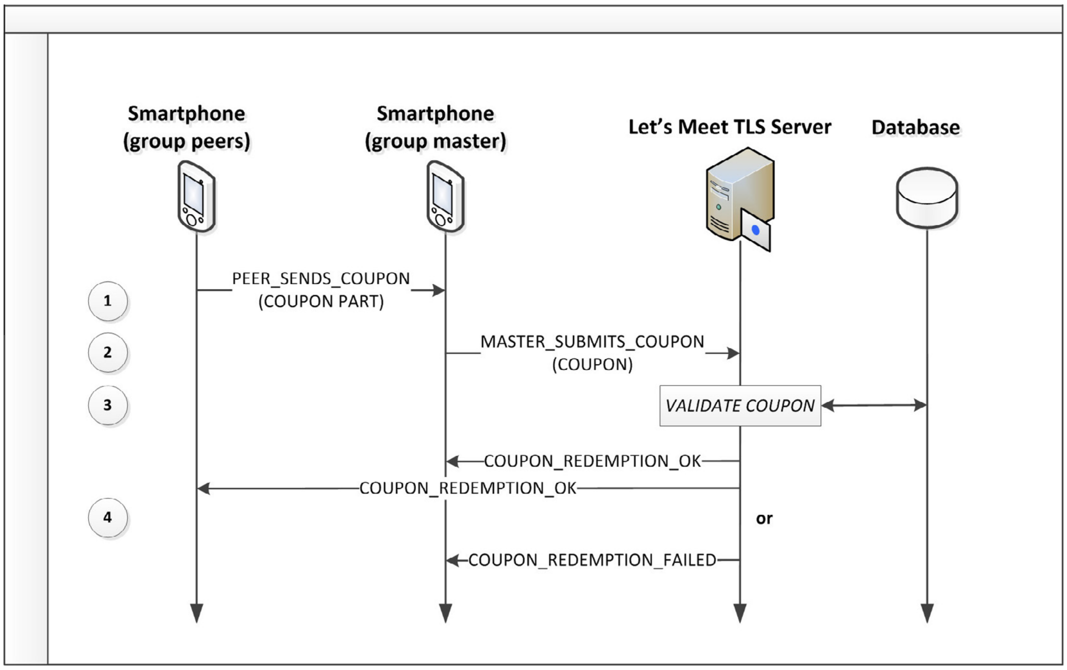


Fig. 11. Coupon exchange and validation.

Speed: Bluetooth 4.0 can transfer data with speed up to 25 Mbps, while Wi-Fi Direct can reach ten times that speed, i.e., 250 Mbps. Of course, the amount of data transmitted from one device to the other is in the case of Let's Meet! very small, just a few bytes. Thus, it seems that this high bit rate is for now of low importance. Keep in mind, though, that this high speed makes Wi-Fi Direct a very popular and must-have feature among smartphone users, who wish to exchange directly large amount of data. Therefore, high speed actually adds even more to the popularity of Wi-Fi Direct.

Security: Both protocols have AES encryption in their security arsenal, but Bluetooth 4.0 uses a 128-bits key, while Wi-Fi Direct uses a 256-bits one, since it implements the WPA2 security standard.

Power consumption: Bluetooth, as of version 4.0., consumes significantly less power while maintaining its theoretical range of almost 100 m. Wi-Fi Direct, on the other hand, can consume as much as 40 times more power.

Reliability: One must admit that due to Wi-Fi Direct's small age, it hasn't been fully tested in the real world about its reliability. But, again, the authors believe that its increasing popularity and its well-known base, i.e. Wi-Fi, will support every effort towards this direction.

As proved above, Wi-Fi Direct prevails over Bluetooth with respect to range, speed and security. Wi-Fi Direct tends to be as popular as Bluetooth and this is very important, since Let's Meet!'s proper function relies on this feature. The most important advantage of Bluetooth over Wi-Fi Direct seems to be the level of power consumed. But in the case of Let's Meet!, the rest of the parameters examined here were considered to be of greater importance. Moreover, the duration of each device-to-device communication is not supposed to last long, so it was decided that it was worth "spending" a bit more power in order to support the framework's functionality. After the recent release of Bluetooth 4.2, that focuses on the IoT, we plan as future work to support Let's Meet!'s device-to-device communication over both Wi-Fi Direct and Bluetooth.

5. Security

When developing a mobile app, the developer faces great challenges, as far as the protection of the data is concerned. All the communications between the mobile device and the server are wireless, thus subject to interception or modification by a malicious user. On the other hand, the mobile device is not under the control of the developer but under the control of the end-user. One cannot know what the intentions of a user are or what will happen if the device gets stolen or comes into the possession of an ill-motivated person. Hence, careful choices must be made as to which side the data reside (server or device), how it is protected there and how it is communicated from one party to the other.

Of equal importance is also the fact that the algorithms that implement the business logic of the app are as secure as possible. In this direction, it is generally a good practice to use solutions thoroughly tested and suggested in the literature than implementing new ones.

Another concern that must also be given great attention is the user's privacy. As discussed in the introductory section of this work, users are asked to give some of their private information away in exchange for some value added service. Nevertheless, the users need to know exactly what kind of information they disclose and how it is used in order for a trusted relationship to be built. This is why the Let's Meet! framework is designed from the ground up in a way so as to retrieve the minimum user information it needs to function properly and nothing more.

In this section, we discuss the approaches made during the development of the proposed framework so as to secure the merits mentioned above, namely:

- TLS is used for server authentication, and session confidentiality and integrity.
- OAuth 2.0 is used for user authentication in an opt-in basis.
- Only approximate locations are propagated to other users. The server never stores location information in its DB.
- No personal information or user preferences are requested.
- The OATH Challenge-Response Algorithm (OCRA) (M'Raihi et al., 2011) is used for generating the one-time-coupon per group.
- A user is identified through her device IMEI, which is hashed by means of a keyed-Hash Message Authentication Code (HMAC) at the device before being transmitted to the server. As discussed further down, the key used for the hash function is unique for each user and must be typed in each time the app launches.

5.1. Threat model

In a client–server app like this one, there may be malicious entities trying to impersonate either the server or some user. The mechanisms used to ensure authentication are different for these two parties and so are discussed separately. If some entity manages to impersonate both participants, this is what we call man-in-the-middle attack. In the following we'll see that the server authentication is strongly implemented, thus securing the system from any server impersonation or even man-in-the-middle attack. The user authentication, although secure, is not mandatory and gives opportunities for client impersonation attacks. In any case, this feature is offered with respect to the user's privacy and it is up to her to take that risk. We make the following assumptions:

Broker is a Trusted Entity: It is obvious that the broker is considered a very sensitive entity, since private information about the users reside there, either stored permanently in the DB or for a very limited amount of time in the server's RAM. To be more precise, the users' exact location exists in the server's RAM for a while. Moreover, the OTC for all the offers are generated there. Last, the device IMEI of all the enjoyers are used by the server and also stored in the server's DB. In either case, this information is considered private. This means that if the Broker is curious or compromised by an external entity the protection of this information is at stake. Therefore, it was assumed while developing this framework that the broker is a trusted entity and as least vulnerable to attacks as possible. Nevertheless, for the case of IMEI further precautions are taken as discussed in Sections 5.4.2 and 5.7.1.

Despite the fact that a user willing to remain anonymous can choose to skip authentication, there still exist mechanisms that can track her down (Linn, 2005). So our server needs to build a higher level of trust with current and potential users and this can be achieved through a privacy policy. It is in there guaranteed that the users' personal information is used in no other way but solely for the proper function of the service. In case the server turns out to be semi-honest, there are solutions for securing the web communications of this framework, but are out of scope of this work. However, the interested reader can refer to Ruiz-Martinez (2012) for further details on this topic.

Producer-Broker Communication: The way producers publish their offers has not been implemented in our prototype. We assume, though, that the most practical and secure way would be a web or mobile interface that would be also protected with TLS. As an extra level of security, password-based encryption (PBE) could be applied. Since the number of the producers will not be that large, every new producer could register and receive a PIN. An algorithm would then derive the encryption key, using the PIN that the producer would type when the application starts up.

5.2. Server authentication

5.2.1. Certificates

The server authentication is guaranteed by implementing the TLS protocol (Dierks, 2008) in our framework. For this scheme to function, a Trusted Third Party (TTP) is needed and in our case this role is played by the University of the Aegean. The server is actually hosted within the University premises. So, using the *openssl* library, a self-signed X.509 certificate was created for our Certificate Authority (CA), which from now on will be called AegeanCA. Keep in mind that AegeanCA is a CA and as such can sign her own certificate. It possesses a key-pair, whose length is 2048 bits long.

The server authenticates itself by presenting a X.509 digital certificate, signed by AegeanCA. The certificate's common name (CN) is letsmeet.aegean.gr and it matches the Fully Qualified Domain Name (FQDN) of the machine that hosts the Let's Meet! server. The reason will get clear below, when the certificate's validation by the client will be discussed.

5.2.2. Certificate validation

TLS is generally regarded as a strong authentication mechanism. In order to take advantage of its power, though, any client using it must be correctly implemented. If the client fails to make some important checks, then the power of TLS is nullified and the system is susceptible to attacks. This problem becomes more common nowadays in mobile apps that use TLS, in contrast with traditional web clients (i.e., browsers) where there has been a continuous testing for many years now and they are considered secure. Mobile apps grow rapidly and many new developers come into this field. But failing to carry out some checks that TLS unfortunately does not mandate is as if no TLS is used at all.

Before getting into the details of TLS implementation in our client, let's review first the steps that any decent TLS client should perform, when presented with a server certificate:

1. check certificate validity period;
2. make sure that the issuer is a trusted entity;
3. validate hostname;
4. make sure that the certificate has not been revoked.

Regarding Let's Meet! framework, the prototype is implemented in Android 4.0+, so some details about this phase will be more Android-specific. The principles though apply to any OS a mobile device may be running. So, the step 1 above is simple to implement. As far as step 2 is concerned, when the client app is downloaded to the user's device in the form of a.apk file, the AegeanCA certificate is included as a raw resource. When the app is launched by the user, this certificate is used as the sole trusted certificate of the app, i.e., the truststore's size is equal to one. This means that any server this app tries to connect to must present a certificate issued by AegeanCA. Certificates issued by any other CA or self-signed certificates are rejected. So, this app bypasses Android's built-in list of trusted certificates and this way it makes sure that a rogue CA that may be injected somehow in the OS of the device cannot affect the app. This technique is known as pinning.

The other important step is hostname validation and failing to do it is a common mistake. The certificate the server presents includes a common name (CN) and the value in this field is the name of the entity this certificate was initially issued for. This value should be identical with the URL that the client app connects to. If the app fails to run this check (and step 2 is also missing), then some rogue server with an otherwise valid certificate but issued for a completely different entity can be erroneously considered valid.

Finally, the server certificate must not have been revoked. To make sure about this, the client app retrieves the Certificate Revocation List (CRL) of the issuing entity and checks whether this server certificate is in there. This check is necessary for cases where the server's private key has been compromised and a new keypair had to be generated. A malicious entity trying to impersonate the server would still use the old certificate and any client app would therefore be able to detect and reject it.

5.3. Confidentiality and integrity

Keep in mind that all the above steps are necessary to guarantee proper server authentication. An app that fails to implement correctly any of them is susceptible to Man-in-the-middle attacks. But even then, confidentiality is ensured by means of data encryption. The TLS handshake procedure ends successfully producing a session key hierarchy, which is then used throughout the session by both parties. All the data traveling back and forth are now encrypted and decrypted with this key, thus making eavesdropping very hard to achieve. Moreover, TLS offers integrity checks, using hash functions for computing Message Authentication Codes (MACs). Thus, the data are very difficult to be tampered with.

5.4. User authentication

The users on the other hand have no certificate. A certificate is costly and requiring to purchase one in order to use the app would discourage the majority of potential users from using it, thus leading to a less flexible framework. Instead, the OAuth 2.0 framework is used.

5.4.1. OAuth 2.0

Some well known and widely acceptable web services, now also available in mobile versions, manage among others their users' DB. So every user that wishes to use these services should first login and authenticate herself. This would be simple if that specific user was always using the service by herself. Soon appeared cases where the user would like to delegate the usage of this service to a third entity, be it a person or some software. It would be inappropriate to share her credentials with this entity, because this would grant full access to her account. Also, such a scheme would be inefficient in case the user decided to change her credentials.

This problem was faced by requiring that the user gives explicit authorization to any third entity that wishes to use the service on the user's behalf. This authorization is accompanied with a list of specific rights the entity acquires and are embedded in a token that is sent back to this entity. From now on, the entity may call the service APIs by appending the token, thus proving that is has the authorizations to access the corresponding resources. This scheme gives fine-grained control over the rights that are being granted to any third entity, unlike the "all or nothing" approach of disclosing one's credentials. The token may also be revoked at any time, in case the user decides that she no more trusts this entity.

Almost all well-known social networks have developed authentication APIs and provide this feature to third-party applications, either web or mobile. The implementation specifications can be found in RFC6749 (Hardt, 2012). Facebook, Google+, Twitter, Foursquare, Dropbox, etc., which have one way or another their own user DB, offer this functionality. In exchange they require that the user grants some minimum rights to the app. In our case, the third entity is the Let's Meet! app, which gives the user the option of authenticating herself with Facebook. If the user moves on, she is forwarded to Facebook and supplies her credentials. Upon successful authentication and while still in the Facebook domain, the user must consent to give the app access to her public profile in Facebook, which is the username and the profile photo, and to her list of friends. Only when she agrees, will the token be generated and the Let's Meet! app proceed to the next screen.

Naturally, there is a privacy issue here, regarding the user's personal information from Facebook being disclosed to the app. The app **does not use** any of them, respecting the user's privacy, but the users are often suspicious when asked to grant access to their personal information. This holds true for any app that uses this feature and the problem may be overcome by building a good reputation over time, so that the users eventually feel they can trust the app. Generally speaking, every app should request the least data needed to function properly and always inform the user about this.

5.4.2. Skipping login

Let's Meet! offers a complete anonymous mode. There is an option of skipping login for users that do not want to login with Facebook and preserve their anonymity (Kambourakis, 2014). This mode, though, has the side effect of a user being more vulnerable to impersonation attacks. In this scenario, the only data that identifies the user to the broker is the IMEI of the device (as discussed in Section 4). The IMEI seems to be a private information and not straightforwardly known to others, but it is not impossible to leak. Moreover, as already pointed out in Section 5.1 the way it is generated makes it not so random (i.e., has low entropy). It is a 15-digit numeric code, some digits of which represent the device origin and model and some others are manufacturer-defined and express the device serial number. There is also a check digit.

In order to face this weakness, the client app could produce, say, a SHA-256 digest of the IMEI before transmitting it to the server. However, even this anonymization scheme is considered weak, and in combination with IMEI's structure, makes this number prone to brute-force de-anonymization attacks. So, we choose to apply a stronger scheme which requires a passphrase typed in by the user when launching the app, and used subsequently as a key to produce an HMAC-SHA256 of the IMEI. The outcome is then transmitted to the server and this data is what the server knows as the userID of the specific user. The robustness of this approach is discussed in detail in Section 5.7.1.

5.5. One-time coupon

5.5.1. OTC generation

Remember that the main objective of the Let's Meet! framework is to get people together, to make them meet in person, so that they can enjoy some offer that requires two or more members. So, the procedures that guarantee that all members have indeed met each other should be also securely implemented. The users qualified to compose a group and only them must become a team and move on to the purchase.

To achieve the above goals, the broker uses the algorithm described in RFC4226 (M'Raihi et al., 2005) to generate an 8-digit numeric string (OTC), which is actually a keyed-Hash Message Authentication Code (HMAC)-based one-time-password (HOTP). This algorithm is capable of generating numeric strings comprising of 4–8 decimal digits. In order to get the best possible outcome, we selected the maximum number of digits, which is 8. Having in mind that this coupon will then be split to 2, 3 or 4 parts, the number that each member receives is also of practical size, although for now this number is only utilized by the app and not by the user herself.

The inputs of this algorithm are an incrementing counter and a secret key. Keeping the key constant and incrementing the counter by one, the resulting coupons will be uniformly and independently distributed. According to the RFC4226 (M'Raihi et al., 2005), the best possible way for a malicious user to guess the next value of the coupon is to perform a brute force attack. Any previous values that this user may have intercepted can be of no help to her. To be more precise, the implementation used in our case is OCRA, which is a generalization of HOTP with inputs not solely based on an incremented counter and secret key value. OCRA is documented in RFC6287 (Hardt, 2012). According to this documentation though, OCRA does not guarantee confidentiality and must be implemented within a secure environment such as TLS or IPSec. In our case, TLS has been selected.

The broker generates a coupon for each new group. Since this group acquires a unique ID from the DB, this groupID is the most suitable value to feed the algorithm. Regarding the secret key, this is never stored at the server's disk but always computed every time the server launches and kept in RAM. To generate this key a key derivation function is used. Multiple operations of hashing, performed on a password combined with a salt return the desired key. In our case, we use the *PBKDF2WithHmacSHA1* suite, with 1000 computing iterations, while the password and the salt remain constant since the key generated must always be the same for the OCRA algorithm to function properly. The key derived is 256 bits long.

5.5.2. Coupon exchange between peers

The step required to guarantee that the peers have actually met each other and are in close proximity to each other is the coupon exchange phase. Bear in mind that the broker sends some coupon parts to the members of a group. The first user that becomes part of the group is called the group master. She is supposed to be standing at the precise offer location. For instance, in the case of cinema tickets, she could be standing in front of the cashier. Or in the case of a dinner offer, the group master could be waiting in the restaurant lounge. As a new peer arrives, she is expected to approach the offer location. Then, her device starts looking for the master using WiFi-Direct technology. When she finds her, the coupon part is transmitted to her. Since this Wi-Fi Direct communication works within a range of a few meters, then the successful exchange of the coupon can only mean that these two users are physically close one another.

Similarly, all other members of the group must send their coupon to the master in the same way. When the master collects all parts, she reassembles the coupon and sends it back to the broker for validation. The fact that the broker sent parts to many members and receives the whole from only one is a guarantee that all members have come in face-to-face contact.

5.6. User privacy

As discussed in Section 1, the mobile marketing apps try to promote products or services to people, but they strive to make the promotions more targeted without invading to people's personal lives. Only when a promotion is tailored to the user's personal characteristics and needs, is it possible to be acceptable and perhaps to result in a purchase. The user on the other hand recognizes the value of targeted promotions and wishes to receive them in order to get suggestions about what she could buy.

But a user profile must be built for the above scheme to work. The user has to give away some of her personal information: either demographics such as age, gender, education level, occupation or personal preference data. There are apps that explicitly ask for these data, for example through a registration form. The app could also collect preference data by monitoring the user's actions while interacting with the system.

Therefore, one important question arises: how much about herself is a user willing to disclose in order to receive personalized promotions? The answer is different for each user, as is the feeling about one's privacy. Privacy is anyway defined as the right of the person to choose what is meant to be shared with others and what is not.

Let's see how Let's Meet! protects the users' privacy.

5.6.1. Location privacy

Each user's device retrieves its current location from a geolocalization service (Google Location Services in our case) with the best possible accuracy. This location is reported to the broker for some decisions to be made. However, before being propagated to the user's fellows, this location is obfuscated in a radius of 50 m. This process produces always a different outcome. So even if a user stands still somewhere, her fellows could see her marker on the map moving in a radius of 50 m (the marker will move about every minute, which is the location update frequency).

What is achieved here is protecting the user's location. One must always keep in mind that some fellows in Let's Meet! are most possibly complete strangers in real life. At least this must be true for the first time some users join a common group. So, it is very likely that they do not wish to share their location with a stranger and Let's Meet! respects that. On the other hand, any location information reaching the server is never saved to the DB. Upon receiving a new location, the server decides instantly (online) which are the fellows of this user and informs them about this (obfuscated) location. This information exists only for a very small period of time in the server's RAM and nowhere else. As a result, even if the DB server was compromised, no location information would leak.

5.6.2. No user profiles

The framework is designed to function solely with the current interest of a user in some offers and her location. There is no personal information that is being collected in any phase of the app. Even in the case that the user authenticates herself through Facebook, the information that the app receives access to is not used at all. So, the users can rest assured that their personal information is not used or saved anywhere by the platform and of course cannot leak to third parties.

The system also does not collect any information from the user's interactions with the app. For example, when someone claimed frequently offers of a certain category or type, this would mean that she is most interested in them. A minor user profile could be built and later used in different ways. In Let's Meet! this is not done, although it would improve the user experience and make more targeted suggestions.

5.7. Other security implementations

5.7.1. Hashed user IDs

Each user is identified by the system through the IMEI number of her device, no matter if she has chosen to authenticate herself or to skip the login process. As discussed earlier in this section, the IMEI is a low entropy number, hence it is not a good idea to use it for the above purpose as it is. More specifically, the user must type a passphrase every time she launches the app. This passphrase is used as key to a HMAC-SHA256 hash function, which returns the digest of the IMEI. As long as the user always types the same passphrase, the same digest will always be generated. This value is then used in every subsequent message the client sends to the broker. At the server side, the server deals with hash values in order to identify the users, and not with the real IMEIs. These hash values also identify the users in the server's DB.

In case a malicious app somehow finds an IMEI, it cannot use it to impersonate the user that owns the corresponding device, since the passphrase would be also necessary. Bear in mind that in case a wrong passphrase is inserted, the app connects to the server successfully, but the user "sees" an empty profile. This holds true even in the case of a legitimate user, who makes a mistake when typing. So, in the case of the malicious app, it could try repeatedly all possible passphrases, but there would be no profound way for it to know that it finally made a correct guess. So, there is practically no chance that the user's profile/rating can be harmed this way.

From the server's viewpoint, the real IMEIs do not reside in the DB, but their keyed digests do. Even worse for the attacker, each digest was computed with a different key, that the server or any other adversary do not know. So, if the DB is compromised, the real IMEIs cannot be inferred, since the adversary must take a guess from a precomputed table of IMEIs, combining each IMEI with all values within a list of possible passphrases. This would be a very long lasting and computationally demanding procedure. The same arguments are true, in case sometime the server turns out to be not so trusted and misuses the data it has access to.

The only viable attack here stems from the user side in the form of a Denial of Service (DoS) one. Specifically, a malicious user (or a group of them) could continuously launch a great number of connections using different randomly generated hashes (or the hashings of same IMEI but using diverse passphrases) toward the server aiming to paralyse it. The use of spoofed IP-s makes this attack harder to stop, say, by means of a firewall or Intrusion Detection System (IDS). On the downside, this requires the aggressor to repeatedly enter different passphrases, and thus proved extremely annoying to him (considering that there exist more relaxed ways of achieving DoS). In any case, few will argue that this kind of attack is practically impossible to avoid in all of its versions, and therefore should be always seen in conjunction with all other deployed countermeasures at the Broker's side.

5.7.2. Prevent double login

The server always knows who is online. So, when a user tries to connect to the Let's Meet! server a check is performed to find out if this user is already connected. If true, then the current session fails to connect. This is a minor defence against impersonation attacks.

5.7.3. Automatic log-off

When a user shuts down the app, a log-off procedure is initiated by the server. The online-users list is updated and all corresponding resources are released, mostly threads serving this client. Moreover, the server has an extra protection mechanism that automatically logs off users that have not been alive for a while. A data structure is maintained where the most recent ping from every online user and the time it arrived are recorded. A user is considered inactive when a certain amount of time has elapsed since her last ping was received. This amount is for now static and is equal to double the time interval between location updates. For instance, if each device is supposed to ping every minute, then the system would wait 2 min and then disconnect an inactive user.

Following our discussion in 5.7.1, this mechanism protects the server from a malicious app that could just initiate lots of connections with the server and then do nothing. This would result in resources being reserved and the possible slowdown of the service other users would receive. The auto log-off feature is also useful when the app running on the user's device shuts down unexpectedly for some reason. If this happens in a large scale it would cause problems to the server.

6. Memory and energy consumption

The PowerTutor (2014) app was installed on our mobile device to measure the energy consumption of Let's Meet! The device was connected to the Internet through the mobile data network. The signal quality was not so good and sometimes switched from 3G to 2G and back. The app was used with the data profile as discussed above, i.e., with three active offers and two offers redeemed in the past.

As it will be seen below, the most energy-consuming hardware parts are the display and the CPU. When the app launched, almost 2.5 J were consumed. During the connection procedure, the device generates a session key hierarchy, which is the result of the TLS handshake. These are CPU-intensive commands and justify this amount of needed energy. Later, the app user requested an offers refresh for several times. This procedure consumed in average 0.85 J. When saving the user's claimed offers, another 0.2 J of energy were consumed in average. The location update procedure consumed a trivial amount of energy, since our monitoring tool could not measure it and did not show any change in the total energy consumed.

After a continuous usage of the app for almost a quarter of an hour, we concluded that the LCD display is responsible for consuming almost the 73% of the total energy, with the CPU requesting the rest. Moreover, since during our measurements we had to constantly switch between Let's Meet! and PowerTutor, we suppose that the display should be typically responsible for a bigger percentage of the total energy consumption. Finally, during the above time interval, the average power consumption was 38 mW, 10 mW of which were consumed be the CPU and 28 mW by the display. Let's Meet! was also evaluated with respect to the amount of memory it occupies on the device. A couple of mobile apps were installed to take this measure. Our app takes almost 57 MB of memory space, when talking in terms of Resident Set Size (RSS). The Proportional Set Size (PSS) though is quite lower, i.e., almost 28 MB. This second metric takes the shared libraries into account and considers that each process that uses such a library occupies a corresponding percentage of space in the RAM.

7. Related work

In this section we attempt a review of the most important similar to ours frameworks that have been proposed in the literature so far. Some of them are oriented towards security issues, while others, which are the most recent ones, focus on mobile marketing principles and make little or no reference to security or privacy implementations. For each included framework, we present its most important services, its infrastructure and comment on security and privacy aspects it offers.

7.1. Related frameworks

7.1.1. Secure e-coupons

This work (Blundo et al., 2005) concentrates on the secure distribution and redemption of electronic coupons, via the internet or any wireless medium. More specifically, the objective of this framework is to guarantee that electronic coupons cannot be counterfeited by third-parties or double redeemed by the consumers. According to the authors, the users' privacy is also preserved.

The entities that take part in this scheme are:

- The merchants who sell products or services.
- The advertisers who make a contract with the merchants so as to produce e-coupons and publish them on the web.
- The users who can preview and download coupons from the advertisers' website and;
- The retailers, where users can redeem their coupons with the aim to obtain the product.

In order to provide e-coupon data authentication, message authentication codes (MACs) are generated. Actually, three kind of MACs are used across the framework, which require three different keys:

1. The manufacturer M generates a MAC using a secret key, which is known only to her. The details of the promotion, i.e., discount offered, description, expiration date etc., are protected by the manufacturer.
2. The manufacturer M and the advertiser A share a symmetric key. The advertiser adds a Serial Number (SN) to the promotion data and protects the resulting coupon with a new MAC. The serial number protects from double-spending, since the manufacturer keeps a list of redeemed SNs per advertiser.
3. The manufacturer M and the retailer R share another symmetric key. The retailer, which is the redemption entity, realizes the payment and produces an Invoice Number (IN) with the participation of a banking institute. This number is attached to the coupon data and the resulting data is protected with the third MAC. The IN ensures that a retailer will not ask to be reimbursed for a coupon that was never spent or even use an IN twice, since the manufacturer keeps a list of redeemed INs and can contact the bank to validate a newly-arrived IN.

The framework does not require user registration. Therefore, personal information is not transmitted to the system, thus guaranteeing users' privacy.

7.1.2. e-coupons

e-coupons (Patil and Shyamasundar, 2005) was designed on the top of PayWord (Rivest and Shamir, 1997) and according to its authors is a secure, efficient and lightweight scheme suitable for micro-payments, i.e., payments for small-scale transactions, like internet streaming for a limited time space, VoIP calls, news feed, etc. This system offers security regarding the financial transactions, but at the same time is fast and includes the feature of delegating the spending capability of the user. The scheme is credit-based, so the vendors need to be offered guarantee that they will be paid after redemption.

The main entities are the vendor V, who sells services, the user U who consumes services and the broker B, a trusted third party who certifies the user and guarantees the vendor that she will receive payment. The user makes an initial handshake with the broker and gets a digital certificate. She has then the right to mint chains of "paywords", which are actually digital coins. Paywords are hash-chains and are created in the reverse order than the one they are spent. The root of the chain, known as commitment, is sent only once to the vendor and proves that the user possesses all other paywords. Each payment by the user is validated by the vendor by just applying a hash function on the next payword received, thus is very fast. The redemption is done by the vendor against the broker and can happen at a later time by submitting only the last payword. In this respect, the system does not actually rely on online banking systems and is characterized as off-line.

The messages sent between the entities are protected using the TESLA (Perrig et al., 2002) protocol. The sender produces keys (using hash chains) and uses them, each one for a specific interval of time, to generate a MAC of the message. This key is initially unknown to the receiver and is disclosed only when the interval passes by. So, the receiver must buffer messages for a short period of time, before he can validate them. When he gets the key, he first checks that it falls in the correct time interval and then authenticates the message. A loose synchronization must also be done for the mechanism to function properly.

Since the user has an initial physical contact with the broker, anonymity cannot be guaranteed.

7.1.3. Smart shopping system using social vectors and RFID

Within this framework (Chu et al., 2013), users are able to receive suggestions about products they may like, when they enter a retail store or walk nearby. The suggestions are made based on the users' preferences and on purchases made in the past by similar users. The similarity of users is quantified and computed by means of social vectors.

The users need to run an initial registration phase, where they explicitly give their preferences. Thus, a social vector is being built, which expresses the profile of the current user against this specific retail store. Each time a user makes a purchase, these data are kept in a cloud server, so the system gains knowledge about what kind of users buy what kind of products.

When another user passes by or happens to be inside the store, the system becomes aware of that by using Radio Frequency Identification (RFID) technology and tries to make a match between this newcomer and other past users or groups of users. This match is based on calculus carried out on users' social vectors. Then the "most similar" user's or group's purchase history is recovered and suggested to the new user, since these are the products that will most likely fit to her taste.

Moreover, the framework offers personalized promotions, but the users have to expose a great deal of data about themselves in order to receive these valuable promotions. Their preferences must be given to the system, while their purchase history is also saved in a cloud server. This system is a case where privacy is given away in order to obtain value from personalized promotions.

7.1.4. SMMART: System for Mobile Marketing: Adaptive, Personalized and Targeted

SMMART's (Kurkovsky and Harihar, 2006) main objective is to deliver personalized and targeted promotions to a user visiting a retail store. This framework takes user's interests and preferences into consideration for filtering and ranking the products that a specific retail store is currently offering. At the same time it respects user's privacy and does not transmit any personal information to the server.

The user's preferences are expressed by weighed keywords. The user can initially declare specific keywords and tag a weight on each of them. Then, while the user uses the app and navigates through the products, the app updates her preferences. A product that is being viewed in detail gives a bit more weight to all attached keywords. Every time a weight is altered, a corresponding timestamp is updated. Keywords that are not frequently updated have their weight diminished and finally zeroed.

The above mechanism updates user's preferences efficiently but also respects user's privacy, since all preferences are stored in the device. Each time the app makes a query against the store's server, the preferences are attached and the server decides which ads are best suited to the user. No other personal information that can identify the user is transmitted and no logical connection can be made between ads being sent to the user and purchases that may follow.

The framework uses the "push" technique for promoting products. Also, the app offers a secondary function of searching for promotions, which the user can use while in shop. This is a "pull" case.

7.1.5. eNcentive

The framework of eNcentive (Ratsimor et al., 2003) distributes coupons, tailored to the users' preferences and their current context. To achieve this, each user has an associated profile, which is stored in her device. The infrastructure mandated by the framework is a wireless transmitter in the area of the retail store and a corresponding server. The user can use the received coupons for herself. She can also advertise and distribute them to users with common preferences, even when she walks out of range. In case the new users spend their coupons, the first one gets rewarded for that, e.g., with some extra discount.

User's privacy is preserved by storing profiles locally in the mobile device. So, when a retail store advertises its coupons, these are filtered at the device level and only a user who is interested in such coupons will be presented with this offer. Similarly, when a device advertises its coupons to others, the user's profile stored at the target device will filter incoming promotions. This is how ads personalization is achieved and user's privacy is preserved at the same time.

The system is fair with every user that has participated in the distribution of a coupon. The ID of each device is attached to the coupon and all these devices will receive a reward if the initial coupon is finally redeemed.

No mechanism had been implemented though to guard against double-spending, as of the time this work was published. This scheme can lead to multiples of coupons than the number originally created, a situation that the retail stores should control.

7.1.6. TMAS: Targeted Mobile Advertising System

TMAS (Li and Du, 2012) framework gathers ads published by advertisers and makes suggestions to mobile users based on their profile and context. User profiles are created and kept at the server side, which is a drawback as long as privacy is concerned.

The framework uses the "pull" technique. The mobile device of the user issues queries against the system using keywords. The system finds all related ads and filters them according to the user location and her profile data. The filtered result is sent back to the user. There is also a sophisticated ranking mechanism, so the user is presented with the most related ads at the top.

When the user interacts with the results, the app collects information about her interests and sends feedback, which are used to update her profile. Feedback is also sent to the advertisers, so they can measure their campaign's effectiveness and the users' interests.

7.1.7. Foursquare

Foursquare (Kaplan, 2012) is a well-known framework, available both to desktop computers and mobile devices. The user can share her location and retrieve relevant Points of Interest in the area. She can also check-in in specific places, like restaurants, cafs, theaters, venues, etc. When someone has checked many times in a place, then she collects various kinds of badges. The system then rewards this user for this bagde she's got. Often, this reward is done with co-operation with other companies that are advertising some of their products in this venue. Foursquare is the entity that guarantees that the end-user has indeed visited this place enough times and most likely has used the above products or has received a corresponding advertisement.

On the downside, Foursquare requires a great deal of personal information in order to be of real value to the user. One must share her exact location, but also usually shares the activity she is engaged into. For instance, a user says that she's at the restaurant, she's at the movies, she's playing tennis or at a meeting. She even shares her being at home. So, the platform knows where the user is and what she is doing. Also, due to your Foursquare check-ins, completely strangers or people you are less acquainted with are able to anticipate your location. They know for example that you go to the supermarket every Saturday at 10am. It is again up to her to decide whether it is worth disclosing all these personal data for what she earns back.

7.2. Comparison

Table 3 summarizes the services and the main characteristics of each framework presented here and compare it with Let's Meet! The points we focus on is the framework's security, if the user's privacy is respected, and whether the framework is collaborative and in which way.

As observed from the table, for the first four frameworks, there was no reference found in the respective works regarding any security features implemented. In fact, these works where mostly oriented towards the business model of the corresponding platform. There was an effort to imagine and design more sophisticated ways to reach a consumer, probably taking some default security measures for granted. On the other hand, *Secure e-coupons* (Blundo et al., 2005) and *e-Coupons* (Patil and Shyamasundar, 2005) focus mostly on the security mechanisms and discuss more traditional coupon systems.

The *Collaborative* column shows which projects combine information from multiple users with the aim to return value added services to all the participants. In these cases, a user does not interact only with the system itself but also with other users. This interaction happens mostly indirectly (with the server acting as an intermediary), but users can also come in direct contact to exchange information or coupons. In *eNcentive* (Ratsimor et al., 2003) a user's device advertises a coupon to others and when one of them makes a purchase, then the initial one is also rewarded. In *Smart Shopping System* (Chan et al., 2013), the profiles of users are compared by the server which seeks to find similar users in order to make a suitable suggestion to a new one. In *Let Meet!*, a user's personal piece of information, i.e., her location, is propagated to others by the broker. But the users also come into direct contact when exchanging their coupon parts during the final phase of the whole procedure.

The business model of the *Let's Meet!* framework is not alike any of the other frameworks discussed here. From a technical point of view though, *eNcentive* is quite similar because the users contact each other with a direct communication channel, without any server acting as an intermediary. This contact happens so as coupons are exchanged and more users can benefit from a promotion being offered in a retail store. Such a co-operation exists in *Let's Meet!* too, but is has been implemented as part of the business process and not for further spreading of a promotion.

Regarding the user's privacy, *Let's Meet!* is quite different from all the other solutions here. They all create and manage user profiles, while *Let's Meet!* is designed to function without them. Hence, no user is running a risk of her personal information being leaked. On the downside, this approach makes the app a little less user-friendly, meaning that it cannot

Table 3
Related work comparison.

Solution	Services	Security	Collabo-rative	User privacy
TMAS	Matches Users with Advertisers to deliver tailored Ads			User profiles on server
SMMART	Delivers Targeted Promos in a retail shop. Search for ads			Preferences stored locally. Personal identity not transmitted
eNcentive	User receives personalized promotions. User is rewarded for distributing promotions.		✓	Local user profiles
Smart Shopping System	Customized messages based on purchases of similar customers		✓	Preferences and purchase records stored in the cloud
Secure e-coupons	E-coupon integrity No fake coupons No double-spending	MACs protect messages		Users don't register Personal info not revealed
e-Coupons	Efficient & leightweight for micro-payments Users can delegate their spending capability	User cert. Hash-chains (paywords). TESLA for integrity		Initial registration with Broker/Bank needed
Foursquare	Users are informed for near-by POIs Users are rewarded for check-ins	SSL		Location and activity explicitly exposed
Let's Meet	Groups users by common interest in offers	TLS OAuth2.0 One-Time-Coupon	✓	

remember what the user prefers and make quick suggestions about offers. From this point of view, *SMMART* and *eNcentive*, which manage a user profile stored locally on the device, could be considered a bit more fascinating to the user, assuming that measures have been taken to protect the data stored in the mobile device.

As far as security is concerned, our solution adopts all modern security mechanisms with the aim to provide confidentiality and server authentication. In our days, the mobile devices are computationally strong to support this anyway. As discussed in Section 5, the user authentication process can be seen as a weak point in our framework, but in our opinion there should be a trade-off between building an app that always authenticates users and an app that gives the user the option to remain completely anonymous.

8. Conclusions and future work

The majority of modern mobile marketing apps are trying to offer a better user experience by applying personalization techniques in their portfolio. In this respect, tailored ads and promotional messages can be of great value to the end-users. On the other hand, personalization can only be achieved by retrieving some of the users' personal information, which poses privacy issues. As seen in Section 7, most of the frameworks found in the literature, that are similar to Let's Meet!, collect user personal information and build a profile, which is later used to tailor a message to the user's profile and current context. These profiles are stored either in a cloud server or locally, in the user's mobile device. In the first case, even if all possible security measures have been exercised, there is always a risk that sensitive data could leak. The latter case is a much better scenario, although the data are still exposed to dangers, such as when security is improperly implemented. One would argue that storing no profile data at all would be the best solution for the end-user, but such an app would cease being useful.

The framework proposed in this paper groups people, based on their common interest in an offer about a product or a service and tries to bring them together. For this purpose, one's location is shared among the users of the group, but with respect to the user's privacy. When the location of the user is transmitted to the system, this information resides at the server side for a limited amount of time, while, at the same time, it is deliberately obfuscated before being propagated to the others. As far as security is concerned, the server authenticates itself by using a X.509 digital certificate. This way, confidentiality and message integrity are also guaranteed. The users have the option to be authenticated by a third-party authentication provider (Facebook), but a complete anonymous mode is also supported. As a conclusion, our solution is a mobile marketing tool that does not lack any of the security features every mobile app should have. At the same time, an approach of disclosing as little personal information as possible is attempted, with respect to the user's privacy. As a result, this mobile tool is valuable to both producers and consumers. Producers are able to sell more of their products or services, while users have access to them in lower prices.

Taking the above into account, one can argue that the strict policy of the Let's Meet! framework against disclosing user personal information results in an app, that is not so exciting, as far as usability is concerned. More specifically, a repeating user interacts with the app as if she was always a newcomer. The system keeps no information about her, so it cannot identify her in any of the next times and cannot be more helpful. On the other hand, it has been proved (Beatrix Cleff, 2007; Toch et al., 2010) that users have a more positive attitude against apps that ask for some personal information, as long as they are given detailed control over what is shared and for which reason. Having these in mind, it would be considered a good approach for a future version of Let's Meet! framework to maintain a minimal user profile and, if the user consents, to use it for elevating the user experience. Initially, offers should be categorized. Then, a user's interactions with an offer, i.e., claiming it and appearing at the offer location, would be used to build a user preferences model. These data could be later used to make more suitable future suggestions. Again, the user's ability to control this system behaviour and to cancel it anytime she wishes is mandatory. This is the only way a trusted bilateral relationship can be built.

As far as the rating system is concerned, the system could deal with the users in more detail when rating them. The user's appearance time, for instance, would influence her rate and so all users that managed to be included in a group would not be treated equally. In this manner, faster users would be rewarded for their speed. In general, a more sophisticated rating algorithm can be adopted and each user's rate can affect her future transactions with the system. If, for example, a misbehaving user is a priority one, with respect to an offer, then the system could adjust her priority to a lower value, in favor of another with a better reputation, and thus give better chances both for the offer to be sold and for the users to enjoy it.

Overall, this work can be used as a reference towards building marketing tools that take the users' privacy under serious consideration. Such an app needs to grant end-users full control over what they share about themselves. Only in this context is it possible for an app to preserve a large user base and keep every one of them happy.

References

Adam, S., 2002. A model of web use in direct and online marketing strategy. Electron. Markets 12 (4), 262–269. http://dx.doi.org/10.1080/101967802762553521.

Android Developers, Android Platform Versions 2014. URL https://developer.android.com/about/dashboards/index.html.

Blundo, C., Cimato, S., De Bonis, A., 2005. Secure E-coupons. Electron. Commerce Res. 5 (1), 117–139. http://dx.doi.org/10.1023/b:elec.0000045976.24984.48.

Chu, Terry H.S., Hui, Felix C.P., Chan, Henry C.B., 2013. Smart shopping system using social vectors and RFID. In: International MultiConference of Engineers and Computer Scientists. vol. I. March 13–15, 2013, pp. 239–244.

Damopoulos, D., Kambourakis, G., Anagnostopoulos, M., Gritzalis, S., Park, J., 2013. User privacy and modern mobile services: are they on the same path? Pers. Ubiquit. Comput. 17 (7), 1437–1448. http://dx.doi.org/10.1007/s00779-012-0579-1.

Dierks, T., Rescorla, E., 2008. The Transport Layer Security (TLS) Protocol, Version 1.2, 2008. URL <http://tools.ietf.org/html/rfc5246>.

E, Beatrix Cleff, 2007. Privacy issues in mobile advertising, international review of law. Comput. Technol. 21 (3), 225–236. http://dx.doi.org/10.1080/13600860701701421.

Fraenkel, O., Westin, Alan F., 1967. Privacy and Freedom. Atheneum, New York, p. xvi. Ann. Am. Acad. Political Social Sci. 377(1), pp. 196–197 (1968). URL http://dx.doi.org/10.1177/000271626837700157.

Hardt, D., 2012. The OAuth 2.0 Authorization Framework. URL <http://tools.ietf.org/html/rfc6749>.

Kambourakis, G., 2014. Anonymity and closely related terms in the cyberspace: an analysis by example. J. Inf. Secur. Appl. 19 (1), 2–17. http://dx.doi.org/10.1016/j.jisa.2014.04.001.

Kaplan, A.M., 2012. If you love something, let it go mobile: mobile marketing and mobile social media 4 ×4. Bus. Horiz. 55 (2), 129–139. http://dx.doi.org/10.1016/j.bushor.2011.10.009.

Kobsa, A., 2007. Privacy-enhanced personalization. Commun. ACM 50 (8), 24–33. http://dx.doi.org/10.1145/1278201.1278202.

Kurkovsky, S., Harihar, K., 2006. Using ubiquitous computing in interactive mobile marketing. Pers. Ubiquit. Comput. 10 (4), 227–240. http://dx.doi.org/10.1007/s00779-005-0044-5.

Li, K., Du, T.C., 2012. Building a targeted mobile advertising system for location-based services. Decis. Support Syst. 54 (1), 1–8. http://dx.doi.org/10.1016/j.dss.2012.02.002.

Linn, J., 2005. Technology and Web user data privacy: a survey of risks and countermeasures. IEEE Secur. Privacy, 52–58.

Loukas, A., Damopoulos, D., Menesidou, S., Skarkala, M., Kambourakis, G., Gritzalis, S., 2012. MILC: a secure and privacy-preserving mobile instant locator with chatting. Inf. Syst. Front. 14 (3), 481–497. http://dx.doi.org/10.1007/s10796-010-9254-0.

M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., Ranen, O., 2005. HOTP: An HMAC-Based One-Time Password Algorithm. URL <http://www.ietf.org/rfc/rfc4226.txt>.

M'Raihi, D., Rydell, J., Bajaj, S., Machani, S., Naccache, D., 2011. OCRA: OATH Challenge-Response Algorithm. URL <http://tools.ietf.org/html/rfc6287>.

https://www.bluetooth.org/en-us/specification/adopted specifications 2014. Bluetooth core specification 4.2. URL https://www.bluetooth.org/en-us/specification/adopted-specifications.

Patil, V., Shyamasundar, R.K., 2005. E-coupons: an Efficient. Secure Delegable Micro-Payment Syst. Inf Syst. Front. 7 (4–5), 371–389. http://dx.doi.org/10.1007/s10796-005-4809-1.

Perrig, A., Canetti, R., Tygar, D., Song, D., 2002. The TESLA broadcast authentication protocol. Cryptobytes 5.

PowerTutor 2014. A Power Monitor for Android-Based Mobile Platforms. URL <http://powertutor.org/>.

Ratsimor, O., Finin, T., Joshi, A., Yesha, Y., 2003. eNcentive: a framework for intelligent marketing in mobile peer-to-peer environments. In: The 5th International Conference on Electronic Commerce (ICEC 2003). October 03, 2003.

Rivest, R.L., Shamir, A., 1997. PayWord and MicroMint: two simple micropayment schemes. Secur. Protoc., 69–87 http://dx.doi.org/10.1007/3-540-62494-56.

Ruiz-Martinez, A., 2012. A survey on solutions and main free tools for privacy enhancing Web communications. J. Network Comput. Appl. 35 (5), 1473–1492. http://dx.doi.org/10.1016/j.jnca.2012.02.011.

Toch, E., Cranshaw, J., Drielsma, P.H., Tsai, J.Y., Kelley, P.G., Springfield, J., Cranor, L., Hong, J., Sadeh, N., 2010. Empirical models of privacy in location sharing. Proceedings of the 12th ACM international conference on Ubiquitous computing - Ubicomp '10 URL http://dx.doi.org/10.1145/1864349.1864364.

Wi-Fi Alliance, Wi-Fi Direc, 2013. URL <http://www.wi-fi.org/discover-wi-fi/wi-fi-direct>.

Xu, H., Luo, X.R., Carroll, J.M., Rosson, M.B., 2011. The personalization privacy paradox: an exploratory study of decision making process for location-aware marketing. Decis. Support Syst. 51 (1), 42–52. http://dx.doi.org/10.1016/j.dss.2010.11.017.