The current issue and full text archive of this journal is available at
www.emeraldinsight.com/1742-7371.htm

REGULAR PAPER

Air-Writing: a platform for scalable, privacy-preserving, spatial group messaging

Rene Mayrhofer
Department of Mobile Computing, University of Applied Sciences Upper Austria, Hagenberg, Austria
Alexander Sommer
Evolaris Next Level GmbH, Vienna, Austria, and
Sinan Saral
Technical University of Vienna, Vienna, Austria

Received 13 January 2011
Revised 17 May 2011
Accepted 21 June 2011

Abstract

Purpose – Spatial messaging is a direct extension to text and other multi-media messaging services that have become highly popular with the current pervasiveness of mobile communication. It offers benefits especially to mobile computing, providing localized and therefore potentially more appropriate delivery of nearly arbitrary content. Location is one of the most interesting attributes that can be added to messages in current applications, including gaming, social networking, or advertising services. However, location is also highly critical in terms of privacy. If a spatial messaging platform could collect the location traces of all its users, detailed profiling would be possible – and, considering commercial value of such profiles, likely. The purpose of this paper is to present Air-Writing, an approach to spatial messaging that fully preserves user privacy while offering global scalability, different client interface options, and flexibility in terms of application areas.

Design/methodology/approach – The authors contribute both an architecture and a specific implementation of an attribute-based messaging platform with special support for spatial messaging and rich clients for J2ME, Google Android, and Apple iPhone. The centralized client/server approach utilizes groups for anonymous message retrieval and client caching and filtering, as well as randomized queries for obscuring traces.

Findings – Two user studies with 26 users show that the overall concept is easily understandable and that it seems useful to end-users. An analysis of real-world and simulated location traces shows that user privacy can be ensured, but with a trade-off between privacy protection and consumed network resources.

Practical implications – Air-Writing, both as an architectural concept and as a specific implementation, are immediately applicable to practical, globally scalable, private group messaging systems. A publicly available messaging platform is already online as beta version at http://airwriting.com

Originality/value – Air-Writing addresses three concerns: flexibility concerning arbitrary messaging applications, user privacy, and global scalability of the associated web service. To the best of the authors' knowledge, previous approaches focus on at most two of these issues, while the authors' approach allows all three requirements to be fulfilled.

Keywords Mobile communication systems, Systems software, Privacy, Mobile computing, Location privacy, Location-based services, Group messaging

Paper type Research paper

International Journal of Pervasive Computing and Communications
Vol. 8 No. 1, 2012
pp. 53-78

1742-7371
DOI 10.1108/17427371211221081

1. Introduction

Textual messaging is one of the most popular applications on mobile phones[1] and still more widely used than the "mobile web"[2]. Potential reasons are a lack of cheap data flat rates in cellular networks until very recently and – maybe more importantly – limited user interface capabilities on current mobile phones. Small screens and limited text input are sufficient for the short messaging service, and both users and services have become accustomed to the surrounding limitations (limited text size, non-real-time, no guaranteed delivery, partial service outages due to network overload, etc.). Thus, textual messaging proves effective and popular in many use cases.

We suggest that textual messaging can be significantly improved by adding various mixture-sets of attributes like the spatial or group (Counts, 2007) component. Previous prototypes have already shown encouraging results in gaming (Ballagas *et al.*, 2007) meeting scenarios (Hazas *et al.*, 2005) and grassroots group communication (Rahemtulla *et al.*, 2008). In this article, we contribute a specific platform design and implementation for attribute-based messaging using mobile devices. Our design explicitly aims at global scale deployment, guaranteeing user privacy even though messages can be localised with full accuracy, and dealing with the inherently limited user interfaces on mobile phones. Potential application areas are manifold; we initially aim at mobile gaming, infotainment, building local communities and localised, personalised advertisement.

Our approach is to use centralised storage for actual messages and rich clients capable of safeguarding their users' privacy in addition to handling the complete user interface. Web clients are supported for interaction with "remote" places, but may not provide the same privacy guarantees. There are several advantages of such a centralised approach – the most important is the pragmatic reason that web server based architectures are well understood and supported by a wide range of potential clients and hosting platforms. Although peer-to-peer based approaches seem more compelling in many scenarios (e.g. infrastructure-less interaction) and may potentially offer better scalability, practical experience shows that client/server architectures (currently) often provide better performance and user experience. Global scalability can be achieved using already established content distribution networks (CDNs), and we therefore design our architecture to be applicable to such static content distribution. Our suggested client/server protocol is based on standard HTTP connections (e.g. over UMTS) and has been explicitly designed to withstand attacks against user privacy on the level of telecommunication and internet providers. We reach this objective with three design principles:

(1) Clients never transmit their exact location to the server. Instead, they "query and cache group messages" for a larger area and filter locally.

(2) Queries do not contain any unique identifiers (besides the source IP address, which may optionally be obscured). Instead, the underlying messaging concept uses groups comparable to virtual blackboards. Users are by default anonymous and may only optionally identify themselves for posting messages or convenience features.

(3) Queries are subject to randomisation in time and space to impede statistical attacks.

The present article makes two contributions: to present this architecture, and to study its usability and privacy on a specific implementation and a first mobile game.

In total, 26 users participated in two competitive scavenger hunts in the inner districts of Vienna and Graz. Their exact location traces and interactions with their mobile clients were recorded and used to suggest improvements for the prototypical user interface and to simulate how effective attacks on the communication link would be in violating user privacy.

In Sections 2 and 3, we start with a brief discussion of related work and how we use the two terms "attribute-based messaging" and "spatial messaging" in the scope of this article. Section 4 then defines the overall architecture and protocol between a central server and distributed, mobile clients and therefore the main scientific contribution. Both the protocol and filtering and caching methods implemented by mobile clients are responsible for safeguarding user privacy and message security, which are discussed in more detail in sections "safeguarding user privacy" and "secure messaging". A specific implementation of this architecture, Air-Writing, is briefly presented in Section 5 and is online at http://airwriting.com for global public use. In Section 6, we present an initial analysis of its usability (section "usability analysis") and the effectiveness of our privacy safeguards against some statistical attacks (section "privacy analysis").

2. Related work

In an earlier system, interest in spatial messaging was low in practical experiments (Burrell and Gay, 2002) but increasing[3], [4]. Most of the location-based platforms like GeoNotes (Persson *et al.*, 2002) Plazes[5] and JotYou[6] have similar functionalities. One can read and write location-based messages and/or see (meet) friends who are nearby. Air-Writing attempts to go beyond these basic applications and provide a sustainable (Froehlich *et al.*, 2007) and privacy aware system (as explained in Section 3).

Privacy

Privacy for location-based systems has recently become a highly active research area, e.g. Toch *et al.* (2010), Boesen *et al.* (2010), Brush *et al.* (2010) and Scipioni and Langheinrich (2010). For the more specific use case of spatial, messaging systems, only few approaches have so far been presented. Kido *et al.* (2005) describe a solution using dummies for hiding the current location, but their approach leads to additional network traffic. In Air-Writing, we intend to provide privacy protection without narrowing the possible use cases and with lower overhead.

Gedik and Liu (2008) have presented "k-anonymity", which tries to improve the solution proposed by Grunwald and Grueteser (Gruteser and Grunwald, 2003). K-anonymity relies on indistinguishability of k persons in a region. Adversaries shall be unable to distinguish who is who and thus the privacy of each individual is protected. In k-anonymity, the level of privacy is defined by the number of (for the adversary) indistinguishable subjects in a given region – the higher the better. More recently, Toch *et al.* (2010) applied a variant of this concept to location sharing systems by evaluating the entropy of places. In a study with 28 participants, they observed that the histogram of location entropy shows a heavily long-tail distribution with over 82 percent of all locations having been visited by only one participant and therefore uniquely identifying the subject. With its global scope, Air-Writing cannot guarantee a minimum number of users for any given region and time. Therefore, although k-anonymity is a general way of measuring privacy, we cannot directly apply it to Air-Writing in general. However, we can derive from Toch *et al.*'s findings that it is highly important not

to reveal the user's specific location. Our architecture avoids this privacy violation by never pushing detailed client locations to the server but only filtering locally.

Bowen *et al.* (2008) described a way of cloaking for hiding the actual location of individuals. Although similar in aim to the privacy concept used in Air-Writing, our approach consumes less resources by using client caching and filtering and therefore requires only one instead of many server queries. Krumm (2007) compares the effectiveness and applicability of such spatial cloaking with different privacy algorithms based on inference attacks, showing that cloaking, among others, is in principle worthwhile for improving privacy. Our selection of privacy measures (client caching and filtering as an extension to cloaking and randomisation in time and space) were inspired by this work.

Scalability

Another major problem of location-based services is the global scope and its impacts on system performance. Narzt *et al.* (2007) suggest the use of groups for distributing the computational workload over different computers and a "direct addressing" strategy.

Air-Writing also supports groups; they are primarily used for building communities, but are also central to preserving privacy. A variation of direct addressing can be achieved by using CDNs (Peng, 2004) and thus providing global scalability.

In terms of implementing actual server backend, e.g. Banerjee *et al.* (2002) (among others) point out the disadvantages of processing the client requests by using threads. They argue that the thread context switching overhead may reduce the overall system performance, if every client request is handled by a separate thread. This is true for most of the cases but there are some trade-offs. One of them is that such event looping servers require non-blocking or asynchronous I/O operations which introduces a level of code complexity. Air-writing tries to keep the amount of client requests on each server manageable by ensuring a low amount of thread context switching (cf. details in section "scalable server").

3. Attribute-based messaging with mobile devices

Attributes are the main concept for composing messages in Air-Writing. In all messaging systems, the most important attribute is the text itself. Also, the group membership can be seen as an attribute (for group-based systems like newsgroups). Another and currently popular attribute is the location (for location-based messages).

One of the initial aims of Air-Writing is to explore and provide as many different attributes as imaginable regardless of their significance. This is a basis for combinations of attributes, flexible application design, and general exploratory research on messaging applications.

Some of the message attributes currently implemented in Air-Writing are:

- Group-id (mandatory, user-defined, set on client): ID of the group where messages are sent to or received from. E.g. Judas sends to the "love" group.
- Text (mandatory, user-defined, set on client): the text of the message. E.g. Judas sends Maria the text "I want to kiss you!".
- Longitude, latitude (mandatory, sensor-derived, set on client): a message is linked to a specific place with the help of GPS, and therefore is only visible (ready to receive) for another user where the message was sent from. E.g. user

Judas writes a message "I want to kiss you" at "Stephansplatz". If his girlfriend Maria is coming to Stephansplatz, she will receive it. Location is determined by GPS and cannot be directly defined in the user interface (note that Judas cannot "cheat" his current position using normal clients, but could fake it with a modified one).

- Radius (optional, user-defined, set on client): the scope (radius in meters) of a message the client is sending. E.g. Judas wants his message to be visible not only at Stephansplatz Church but really at Stephansplatz location. Therefore, he sets the location radius to 30 meters (instead of the default value of 10 meters).
- Start-date, End-date (optional, user-defined, set on server): messages will be available a specific time. E.g. Judas wants to send Maria a virtual kiss message on valentine's day. Therefore, he sets the time scope of the message to the 14 February.
- Amount (optional, user-defined, set on server): messages can be grabbed from a place *n* times. After the *n*th "pick" action (i.e. reading a message), the message will disappear. E.g. Judas only wants to send one virtual kiss. He sets the pick value to 1 of his message. When Maria receives the kiss message, it will be the only one.
- Encrypted (optional, user-defined, set on client): messages can be locked by textual passwords. E.g. Judas adds a special password to the virtual kiss message that only Maria knows. Nobody besides Maria will be able to read the kiss message.
- Emotion (optional, user-defined, set on client): messages can be added with emotional states. Only users with the same emotional state are able to read the message. E.g. Judas is happy and wants to receive only happy messages.
- Relationship (optional, user-defined, set on client): relationship messages have three different values: single, inlove and family. The single value is for singles, the inlove for users who are newly in love and the family value is for families. E.g. Maria and Judas are singles. Maria is writing flirt messages for singles and Judas, who is setting his profile to single, will receive Maria's invitation for a first blind date.
- Wealth (optional, user-defined, set on client): wealth messages have three different values (poor, normal, rich). E.g. Judas is going to marry Maria and therefore he is setting his wealth state to rich. He is now able to receive adds which are targeted for premium occasions.
- Age (optional, user-defined, set on client): the age attribute allows targeting different age groups. Judas is an adult and he just wants messages from adults, too.
- Gender (optional, user-defined, set on client): the gender attribute helps in targeting messages for males or females only.

Additional message attributes which are planned to be implemented in Air-Writing are:

- *NFC*. A message appears, if a specific or unspecified near field communication (NFC) tag (i.e. message received via NFC) is in range of the client.

- *Move*. Messages will be able to move virtually around the world, causing location-based messages to no longer be linked to a static place but change their location depending on a defined rule (e.g. linear movement). A use case for this attribute is a bottle post based gaming applications.
- *Keep*. Messages appear if a user stays at a place for (at least) a specific time period. This attribute could be part of a sales campaign.
- *Speed*. Messages appear if a specific speed is reached. This attribute is useful for anti radar trap services.
- *Drop*. Picked messages disappear from the mobile client and reappear for the community on the current location of the message, e.g. for a game based on transporting virtual objects.
- *Borrow*. Picked messages disappear from the mobile client and reappear for the community on the original location of the message.
- *Transfer*. Picked messages move directly to another client. This attribute enables client based transfers of messages. A use case for this attribute could be hide and seek based games.
- *Global delete*. A picked message may be deleted from the server and is therefore not available anymore for others. Secure applications may demand such a feature.
- *Local hide*. If a picked message is locally hidden, it is only visible at the server (online platform).
- *Cluster*. The messages of a location become visible for n users, if those n users are at that location together. This attribute could be part of a sales campaign or group-based competition games.
- *Multi place*. The same message is visible at different places. This attribute could be part of a sales campaign.
- *Composition*. Messages can be split and merged together.
- *Change*. Messages can be changed completely. This attribute is useful for wiki based applications.
- *Appendix*. Messages can be appended to other messages, e.g. for a poet's contest.
- *Sequence*. A message becomes visible to the client only if the predecessor message was received. This attribute is important for games and virtual tour guides.
- *Exclusive*. A message X looses its visibility for a client if the client received message Y or vice versa. This attribute is important for dynamic games where decisions can be made within the game.
- *Puzzlecluster*. A message becomes visible if n users with n different messages meet at a specific place. This attribute is useful for investigative games or sales campaigns.
- *Multi language* A message is available in various languages.

Any non-mandatory attributes can be freely combined as shown in the examples below as long as they are not mutually exclusive. If the client assigns the value,

cheating is possible; if the server enforces certain attribute values, it is not. E.g. the location attribute could be faked using a modified client. Possible strategies to avoid or defuse cheating are mentioned in section "protecting against cheating". The question whether attribute values should be assigned and managed from the client or server side does not only affect privacy issues, but also the expandability and flexibility of an attribute-based system architecture.

Server vs client managed attributes

Managing (defining, implementing) the attributes on the server or client side has different implications, advantages, and disadvantages (Table I). As long as all attributes are defined on the server side (we call these server managed), a universal attribute compatibility is guaranteed; all clients know what they have do with an attribute. Defining an attribute includes its name and allowed values. For example, the server managed attribute "relationship" allows the values single, inlove, and family. Management of such an attribute happens both on the server and client sides. That is, if the user sets the relationship value to single, the client will request messages for single messages and the server will only respond with appropriate messages by filtering all possible messages for allowed attribute values (in this case, single).

However, if an attribute is defined by the client itself, all of its management is almost exclusively on the client side, too. The server only needs store the attribute data and meta data in the form of abstract key/value combinations. If a client requests messages that include client managed attributes, the server will respond immediately with all messages, even if some of them are inappropriate or irrelevant to the specific client. From a privacy point of view, this is a desirable property, as the client-side filtering reveals fewer details to a potential eavesdropper. On the other hand, from a performance point of view, client managed attributes lead to higher overhead because the server is no longer able to filter messages for specific clients. This impacts both network messages (causing larger transmissions over the typically limited UMTS networks) and client performance (causing larger local message caches and higher CPU utilization for the client-side filtering).

Another potential issue of client managed attributes is that all clients must implement the attribute protocol and management logic to handle such attributes (and messages) correctly. All clients with the same (protocol) version of the server are capable of managing server managed attributes. If two different client implementations have their own client managed attributes, both of them have to implement the protocol

	Server managed	Client managed
Attribute definition	Server	Client
Universal attribute compatibility	Yes	No
Attribute management	Server and client	Client
Attribute implementation effort	High	Low
Attribute complexity	Very high	High
Protocol complexity	Low	High
Protocol overhead	Yes	No
Attribute request logging and eavesdropping	Yes	Not identifiable
Complexity for implementing context aware aspects	High	Low
Message owner	Server	Client

Table I.
Comparison of server and client managed attributes

and management logic of the other client. The obvious advantage is that clients can be extended without requiring changes on the server side, which allows easier experimentation and prototyping without impacting and in-production (and therefore slowly updated) server infrastructure. One such example is an attribute implementing NFC functionality. The disadvantage is protocol complexity and potential incompatibility between different client versions.

Many of our planned attributes are candidates for client managed attributes, e.g. nfc, keep, speed, transfer, sequence, exclusive. With only minor functional additions at server side, client managed attributes become more effective. If they are able to access, change and share some form of global variables, client managed attributes may imitate basic features of their server counterparts. Table II explains three possible types of global variables for client managed attributes. We propose that global variables for client managed attributes should be defined as simple JSON value lists.

Exploring and evaluating attributes

All attributes are candidates for analytical and empirical studies. Setting the "correct" value of the radius attribute, for example, is not trivial. If the radius of a message is too small, this message may remain "hidden" for most of the users (section "usability analysis"), but if the message radius is too big, the added location context information might be too unspecific. Varying this value for analytical studies is therefore mandatory and will also affect privacy issues (section "secure messaging").

Groups

Because single attributes are often not enough or even not useful enough for creating a meaningful message, we also explore potentially useful combinations of such attributes. Groups are an intuitive means to define usable attributes and attribute combinations. Different groups can re-use such combinations in different ways and therefore increase the value of a single combination.

Air-Writing follows a top-down approach for realising an attribute-based messaging service as shown on Figure 1: a combination of attributes forms a group, and such groups hold many messages. The purpose is first an attempt for flexibility. If attributes or attribute combinations (and therefore groups which are using them) turn out to be useless, they can simply be ignored without weakening the architecture or implementation. Second, it is a call for comparative studies to determine why some combinations fail or are successful. And finally, it provides the capability for managing a vast amount of new message types and messages themselves.

Table II.
Global variable levels for client managed attributes

Level	Definition	Example
1	Level 1 global variables are added to the user profile	A game with high-scores. The high-score logic is fully implemented at client side. Points are stored at the user profile. The user is able to compete with his own high-scores
2	Level 2 global variables are added to the group	A game with high-scores people can share within a group. "Who is the best player of the scavenger hunting game?"
3	Level 3 global variables are added to server	A game with high-scores people can share and compare between different games (groups) "who is the best player of all scavenger hunting games?"

For efficiency, flexibility and user-experience reasons, the visibility of messages is regulated by their membership to groups. A group consists of the group name (freely definable by the creator, which can be any registered user), the creation date, the attribute mixture set (and if all of these attributes are mandatory for messages to this group), a list of members, messages from users, and membership settings (private or public). Examples of groups are:

- *Aidtips* (attributes: group-id, text, longitude, latitude): provide additional information for people with disabilities – a handicapped person visits a tourist attraction and suddenly needs a wheelchair accessible toilette.
- *Amor* (attributes: group-id, user-name, text, longitude, latitude, radius, start-date, end-date): a location-based contact service – a single is looking for a partner within a radius of 50 km on valentine's day.
- *Scavenger hunt* (attributes: group-id, text, longitude, latitude, encrypted): a location-based game – a group of students is playing "scavenger hunt" – to win, they need to unravel some mysteries.

The group "scavenger hunt" has been implemented for a first game and is described in more detail in section "scavenger hunt game 1". This particular group does not require to set all attributes, e.g. the "encrypted" attribute in this group is not mandatory and therefore, messages to this group can consist of locked (encrypted) and unlocked (plain text) messages.

As a final aim, an attribute-based architecture should also be able to communicate with other attribute-based architectures, whereby those architectures can have different sets of attributes and communication protocols. Therefore, they should be able to handle incomplete sets of attributes, also for further extensions. Third parties may use subsets of groups on their own clients or online platforms from our architecture, instead of building their own.

4. Messaging architecture and protocol

The implementation of Air-Writing consists of an internet platform and mobile clients. A message in Air-Writing consists, at the minimum, of the following properties: text (content), longitude, latitude, radius and group-id. Optional attributes which each message may currently include (but are not limited to): subject, user-name, start-date, end-date, encrypted, amount. Both the protocol and the reference implementations for server and client parts are extensible and can easily support arbitrary new attributes.

There are several types of request and response messages in our current Air-Writing protocol. The most important ones are for polling the server Get Messages and for writing a new message Send Message. Figure 2(a) shows a poll request in which

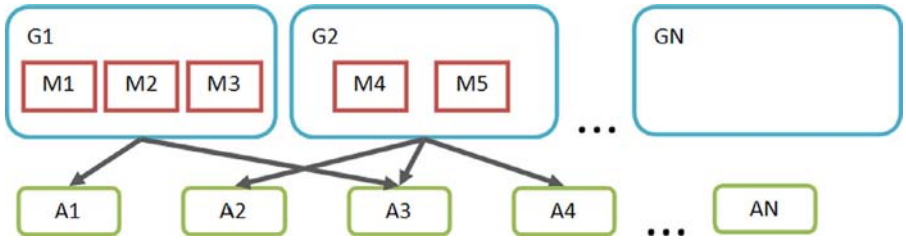

Figure 1.
Service model from a user's point of view

the rich mobile client queries the server for messages in a specific group and a given location (defined by longitude and latitude). The server responds with all messages from the specified group for this area. Note that there is no user name required for any client to receive messages.

Figure 2(b) shows a send message session. The user constructs a message on their mobile device and sends it to the server via a Send Message request. In turn, the server responds with an acknowledgement or failure message; the operation can fail if the user does not have sufficient access rights to post to the specified group (if it is private and restricted to its members) or if the user did not provide a user name to a non-anonymous group (the group creator can specify if anonymous postings are allowed). Consequently, the Send Message request consists of group-id, text, longitude, latitude, an optional user name and all further optional attributes like pick amount, encryption status, and others.

To summarise, our architecture is defined by:

- Messages are organised in "groups", which are arbitrary strings.
- Attributes are managed on the server and/or client side.
- Groups also act as pseudonyms for querying messages.
- Clients request messages based on group name and their location. For each group there is a distinct query (with randomised delays) to prevent tracing based on the specific set of groups a client is interested in.
- Posting new spatial messages can be done anonymously or identified by user nickname.
- Users do not have to login; by logging in, they optionally get a "profile" with their nick and the set of groups they are interested in and can also use a web portal instead of their local device; also, when starting to use a new mobile device, the profile can be used to configure it with the groups it should use.

Safeguarding user privacy

User privacy is, as mentioned before, a critical issue for all spatial messaging services. Air-Writing depends on strong privacy protection as a design principle if users should be expected to use it on a daily basis. The following principles are applied to all parts of the architecture to provide a reasonable compromise between strong privacy guarantees, flexibility, and scalability:

- Queries do not contain any form of identification besides the querying IP address, which can optionally be obscured using standard methods such as tunnelling through the Tor network (Dingledine *et al.*, 2004).

Figure 2.
Basic air-writing protocol for receiving and sending spatial group messages

Notes: (a) Get messages; (b) send message

- User-configurable query intervals are randomised by up to 50 percent to make correlation attacks based on exact query times harder.
- Mobile devices do not transmit their exact location but only an area (as defined in a hierarchical lattice for user-configurable query "width") and receive messages for this scope; clients then locally filter messages to only show those for the exact location; this not only minimises the number of transmitted messages but also prevents recording exact location traces on the server side; additionally, clients can also randomise these queries by also (randomly) querying neighbouring areas in addition to the one they are currently in.

Secure messaging

Besides privacy, Air-Writing also provides secure messaging capabilities. Users may encrypt their messages before sending them to the server when the client platforms support the required encryption algorithms. Any recipient can only see the real content if they know the required password for decryption. Because the messages are encrypted and decrypted on the client platforms, neither potential eavesdroppers (even on the level of internet service providers) nor Air-Writing server administrators can possibly read the content of secured messages.

The decryption key can be generated locally at the client using different means, e.g. from pictures takes with the mobile phone camera (Buhan *et al.*, 2007) or from 2D barcodes attached to the location – keys can of course also be privately agreed using any other out-of-band scheme (a number of these are provided, e.g. by OpenUAT (Mayrhofer, 2007)).

Protecting against cheating

Users or third-party clients may manipulate the value of attributes before communicating with the server. If an attribute value is set by and consequently checked solely on the client, the value can easily be faked because the server has no control over its integrity. In order to avoid cheating, two generic strategies exist: the first is to make cheating useless. If the scope of a message can be set freely, it does not make sense to cheat. The second strategy is to restrict the possibilities to cheat. In order to ensure that messages are accessed only by users on a specific location, users have to physically place passwords or barcodes at that location and encrypt all messages with those passwords or barcodes before virtually placing them in Air-Writing on that location.

Protecting against denial-of-service

By allowing anonymous posting and queries based only on group names, denial-of-service (DoS) attacks become (slightly) easier than when forcing users to log in. This can be mitigated by making queries cacheable and therefore open to using a CDN (such as Akamai). In the current implementation, clients only query "static" HTTP URLs that encode their location area and the group they are interested in. Responses can be regularly pre-generated and pushed into a world-wide distribution network that would be resilient against DoS.

5. Implementation

We have developed a specific implementation of our attribute-based messaging architecture "Air-Writing". It is a private, mobile, group based, spatial messaging

service and currently implements all attributes listed above. Additionally, the first mobile game application "scavenger hunt" has been implemented in the form a of group as explained in section "scavenger hunt game 1".

Scalable server

As mentioned in section "protecting against DoS", "protecting against DoS" attacks can be achieved by using a CDN. The usage of CDNs to store and retrieve messages also enables the system to handle more concurrent user requests and hence increases the total system performance. But there is a small problem with this approach. As Air-Writing is evolving, we discover more and more interesting attributes to offer. Some of these attributes require server side calculations and state manipulation, in order to be retrieved correctly. One simple example for this is the pick attribute. A message which has the pick attribute can only be received as many times as the amount defined in the pick attribute (see Section 3 for more details on the pick attribute). Once a user receives a pick attributed message, the counter of the message should be decreased by one. This processing must be performed achieved on the server side, and most of todays CDNs are capable of computing such simple calculations. However, some complex attributes may require more computation that cannot be achieved by established CDNs.

For this reason, we have developed a second scalability concept which distributes the server load to other clusters. We exploit the fact that Air-Writing is a location-based service and users in one location will mostly interact with messages for that location. By default, messages are saved on one server that has been associated to the respective geographic region. If we notice that some region of the world is generating more traffic, we can create a new server only for that location and all the messages for this specific area are automatically transferred.

The implementation of this set of scalability features also requires the clients to be updated. In previous versions of Air-Writing (as described in our earlier conference paper (Mayrhofer *et al.*, 2010)), the clients always polled one server for the messages. With this new scalability approach, clients need to know which server contains the messages for their region. Clients may query the main server for a list of servers in their region upon startup. This list is saved in-memory on the client and checked before every request to determine which server to communicate with. Therefore, in the new scalability concept, there are two types of servers. One main server where all the central data is stored and managed and which may be clustered and load balanced using traditional high-performance computing techniques supported by standard application servers. Then there are the zone servers which are responsible for a specific geographical region. Zone servers have a scope which is a rectangular area in the GPS space.

We can examine the whole scalability perspective of Air-Writing by looking at the data managed by the system. Data stored on Air-Writing servers can be divided in three categories "user data", "group data" and "message data".

User data. User data contains all the information about users like user-name, password hash, e-mail, etc. User data is not accessed very often by the clients. Users of the system create a user account on the system only once and can log into this account whenever they want. Although it is possible to update the account, this is rarely the case.

Air-Writing only allows to update user account information online via the web portal and the client application does not provide any means for updating account

information directly. Consequently, user data is often read but rarely updated. Hence our new scalability concept does not provide any scalability for writing user data, allowing user data to still be updated on the main server.

On the other hand, user data is read every time a user logs in, including the password hash that is required to authenticate the credentials of the users. Placing this data only on the main server may therefore introduce a bottleneck, but mirroring it to all other servers may not be helpful in terms of scalability because all the servers must be updated once the user updated some parts of their data.

Users often reside in one geographical location and use the system from there. Keeping copies of the user data on unrelated zone servers is thus not necessary and Air-Writing neither copies the user data automatically to all zone servers nor keeps it only on the main server.

Instead, the zone server retrieves the required data on demand from the main server and caches it. As the client applications only interact with the zone servers and the users mostly reside in one region, this approach helps to reduce the traffic on the main server. The next time the user logs in, the respective password hash will be available on the zone server, and the login will be achieved without interacting with the main server.

Group data. Groups are also not created very frequently, allowing the implementation of write operations to be "heavy" in terms of computational effort and/or data transfer. On the other hand, reading of group data is a common task in the system and has to be lightweight in terms of performance.

Groups are created on the main server and also copied to all the zone servers that are within the scope of the group. This ensures that the users immediately see the new groups in their zone and can write to them. Because all the group information is synchronously saved on the zone servers, reading group data does not create any traffic on the main server.

Message data. Messages form the most frequently created and updated data on the system. Therefore, they are created directly on the zone servers. If the scope of a message exceeds the limits of the zone server, the message is copied to the neighboring zone server(s). This means that the zone servers must be capable of communicating among each other without needing the main server. The main server is not involved in creation and retrieving of any messages (Figures 3 and 4).

All client message polling has to be adapted for this approach. Before every poll, the client application has to verify its current location and find the right zone server to communicate with. If the client location is near to the border of the responsibility area of a zone server, then the client application also has to poll the neighboring zone server for messages.

Server backend

The server backend of Air-Writing is responsible for the following operations:

- The backend acts as a communication port for mobile clients. Mobile clients query the server for messages at their location. They may also send messages to the server. All the information such as user accounts, groups, and messages are managed by the server.
- It provides a portal for web-based access.

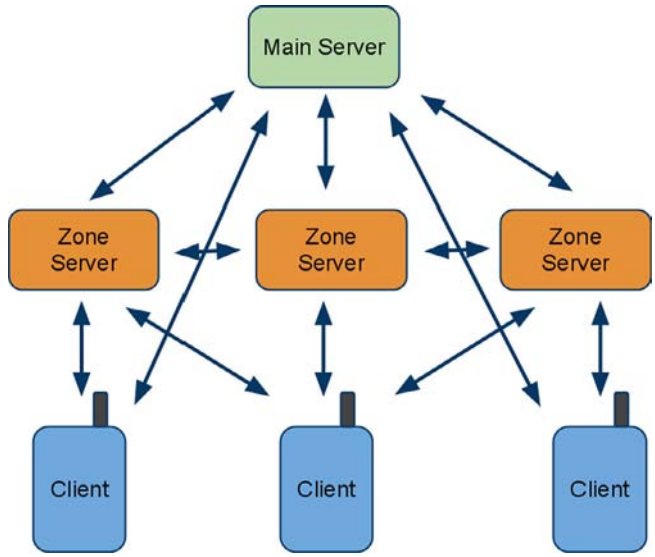

Figure 3.
Main and zone servers

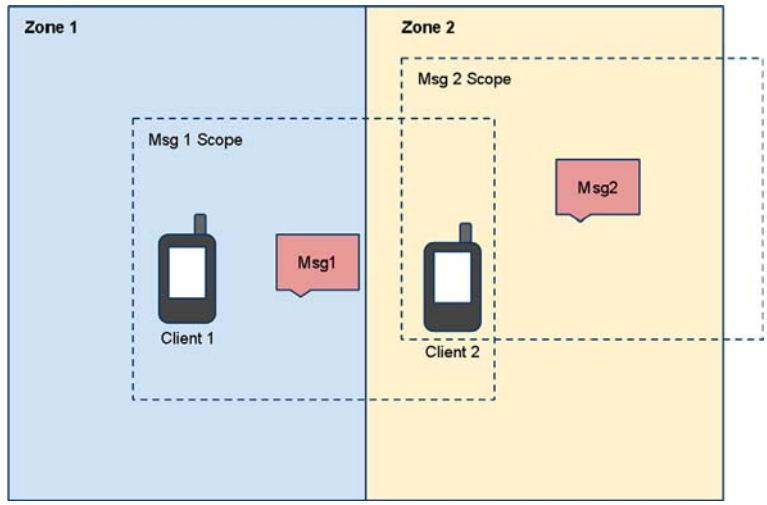

Figure 4.
Zone pooling

- It offers services and aggregated statistical data to third parties (note that even if desirable by some third parties, violation of user privacy is impossible even by server operators due to the design principles explained above).

As shown in Figure 5 several open source frameworks were combined to realise the server backend: PostgreSQL[7] is an open source relational database system. Air-Writing makes use of PostgreSQL on the database layer with the PostGIS[8] extension. PostGIS enables PostgreSQL to operate on geographic objects. Air-Writing saves the scope

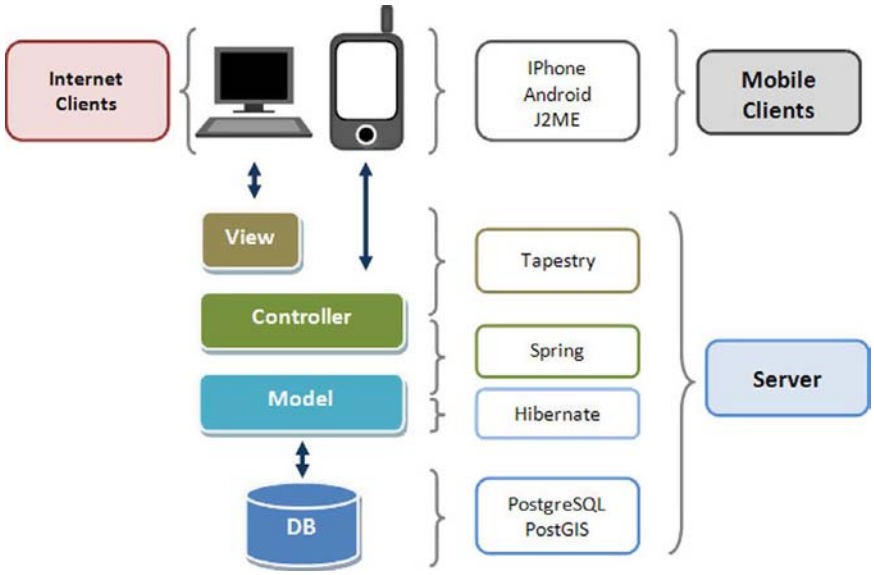

Figure 5.
Infrastructure of air-writing

of messages as geometries in the database. Hibernate[9] is used to map application objects to relational database tables, and business logic resides in service classes which are managed by the Spring Framework[10]. These service classes form the controller layer of the server application and use Hibernate to interact with the database layer. The view layer utilises the Tapestry5 Framework[11] which is a new component based Web Framework. Tapestry5 builds upon the Java Servlet API, and thus runs within any servlet container. For Air-Writing, we currently use Tomcat as the servlet container.

Clients
The client software has been tested on Nokia's E50, N73, N95, Apple's IPhone 3G, 3Gs, 4G and on the Android Dev Phone 1. Screenshots of all platforms are shown in Figure 6(a)-(c) in different application states to highlight comparability of all rich client implementations.

6. Analysis
For analyzing the general Air-Writing architecture and our server and client implementation in particular, we have implemented two games and evaluated them with 26 initial users. The objective of this analysis is to evaluate if the overall system seems intuitive and usable and if our intended privacy guarantees can be met based on real-world data sets. Game 1 was tested exclusively on Nokia Series60 (Symbian) phones, while game 2 was tested with Apple iPhone 3G and 4G devices.

Scavenger hunt game 1
The rules and challenges of the game "scavenger hunt" as shown in Table III are simple:

- All participants in the game have to unravel a mystery question as quickly as possible.

(a) **(b)** **(c)**

Figure 6.
Screenshots of various client platforms

Notes: (a) Reading a message with an Android prototype; (b) setting the attributes with the J2ME prototype; (c) writing a message with the Android prototype

Table III.
The scavenger hunting game 1

Sequenced instructions	
(1) Welcome to Air-Writing at Technical University Vienna! This game will take one hour at maximum and is placed on a one-square kilometer gaming field. You will receive a notification if you cross the borders. Beware, if you cross them more than 3 times, you will loose this game. To win the game, you have to unravel one of two mystery questions which are located at this starting point (so you have to return to your current location at the end) – Lets start, your first quest is simple: find the lord for your next instruction! Hint: Follow the right wing of the owl	
(2) The Lord: Hello Airwriter, you will come to heaven if you win this game. If not, you will burn in hell – sorry [. . .] there is not enough space for everyone in heaven. But don't care, your next quest is simple: say a prayer at the next visible church!	
The Lord	*The Devil*
(3a) The Lord: Good prayer! You will be an angel soon. but first, go to the Karlsplatz church and pray again!	(3b) The Devil: stop praying for the lord and don't go to Karlsplatz church but find some lost souls for me. The next is [. . .] Hm [. . .] Just 3 mins away!
(4a) The Lord: Good prayer! You will win this game as an angel, I will give you the first part of your password: holy – but to win, you have to finish your last quest: find a lost wounded soul in the park!	(4b) The Devil: you have found a lost soul. Yeah! Bring it to Naschmarkt! This is the first part of your password: hell
(5a) The Lord: you found the wounded soul! Bring it back to the owl! The second part of the password is soul. Run back to the owl	(5b) The Devil: My well obeying servant! Thank you for this soul. Your second part of the password is fire. Hurry back to the owl!
(6a) Password: holysoul. Good prayer! You won the game. I will give you wings to fly!	(6b) Password: hellfire. Hahahah! you won the game! come and join me in hell!

- For that reason, the gameplay owner has to provide one locked (encrypted) and some unlocked messages to the gaming field, an area of approximately one square kilometer.
- The unlocked messages act as pieces for solving the puzzle question.

- Additional help messages define the gaming field borders as "border fence is reached".

Gameplay scenario. Maria and Judas are students at the Technical University Vienna. They both have heard about a location-based game close to their university published at Karlsplatz and are interested in playing it. They visit the page with their mobile clients, download, and install the client software. After starting the Air-Writing client software, a group "scavenger hunt" becomes visible. They join it and receive three messages. Message (1) as the welcome and unlocked message and two locked messages (6a and 6b) they cannot yet read. Both start following the "right wing of the owl" order. After a few meters, they arrive at the stone statue and receive a message (2). Maria says to Judas: "This is easy, the church is straight ahead!" They keep on moving and receive two messages at the church (3a and 3b). Maria decides to pray further (go to the respective location), and Judas to find the lost soul (another location). Judas is looking around and receives a warning "border fence is reached" and turns back to the church. He takes another route and finds the next message (4b) and with it the first part of the password "hell". At the same time, Maria arrives at Karlsplatz church and also receives her first part of the password "holy" (4a). Maria is told to look for a soul and starts to walk around again. Judas arrives at Naschmarkt and receives his last order and part of the password "fire" (5b). At the same time, Maria finds the soul (5a). Both start running back to the owl. Maria is first, unlocks one of the two locked messages and wins.

Scavenger hunt game 2

The second game (Table IV) was similar to game 1, but was adapted to the city of Graz. In this game, only iPhone clients were used.

Gameplay scenario. Maria and Judas are students at the University of Graz. Both were invited to take part in a usability test. Maria and Judas are starting the game with their iPhones at a starting point close to the Mensa. When Maria and Judas arrive, they both receive message 1 and 2. Maria does not really understand the question of message 2, but Judas does. He is able to answer the quiz question and receives the hint. Both testers are heading to the University and on the way, Maria receives the pick message (3), which is

Message	Attributes
(1) Welcome Player, your next message is waiting at the mensa!	Text
(2) This was an easy challenge. You will also find a quiz message here somewhere, which will help you in winning the game. The next message is somewhere in front of Uni Graz	Text
(2a) Question: "What is the special menu today offered by the devil?" Answer: "Diavolo Pizza" Text: "If you are in front of Uni Graz, go 10 meters left – and there will be the next important message"	Text, encryption
(3) You have picked a message! This message will help you in winning the game more easily. If you are in front of Uni Graz, go 10 meters left – and there will be the next important message	Text, pick
(4) Honor this scientific place with the right emotion, and you will find the next message	Text
(5) Only crazy people can win this game! Hurry back to the starting place and win!	Text, emotion (crazy)

Table IV.
The scavenger hunting game 2

also telling her that she should move 10 meters to the left when being in front of the university. Maria and Judas are receiving message (4). Maria is faster in getting the "right emotion" (crazy) and therefore is getting message (5) earlier. Maria wins win the game.

Usability analysis

Procedure. The scavenger hunt game 1 was tested with 20 users within a time period of ten days in the city of Vienna. Each test was designed to take one hour at maximum. At the beginning, the idea and application was explained (5 minutes). Afterwards, users received a pre-configured mobile device (Nokia N95, Nokia E50) and started to play at the technical university of Vienna. All important activities were logged on the client and server. After finishing the game, all logs from the client were sent to the server by the test instructor. Finally, the users were asked to report their experiences with a short survey of about 20 questions.

The scavenger hunt game 2 was tested with six users within a time period of two days in the city of Graz. Each test was designed to take half-an-hour at maximum. For this test, pre-configured iPhones were used. The users in Graz reported their experiences verbally only.

Participants. The participants in this study were sampled from a group of students aged between 20 and 55 years. All of them are using the internet and their mobile phone regularly.

Results of the scavenger hunting game 1. This test was focusing the functionality. In total, 19 of the 20 test users (95 percent) would use Air-Writing if it would be available on their mobile phone, even as a permanent background application (88 percent). The satisfaction factor for the usability of the alpha version is moderate to good. About 80 percent liked the alpha prototype but 29 percent had sometimes problems in using it. Most testers would use it both for gaming and writing messages (62 percent), only 8 percent would use it just for gaming purposes. Another 30 percent would just write with it. About 97 percent of the users would use it both on the client and the online platform, the rest just on the client. We also asked the participants to put a monetary value on the security/privacy aspect (which Air-Writing implements). The maximum values were EUR 150 per year, EUR 2 per month, or EUR 0.3 per message. The lowest values were EUR 0.3 per message with no yearly or monthly cost. Some users seemed reluctant to pay for such a service, which suggests advertisement-based models. However, the large differences in the willingness to pay indicate different expectations and should be analysed further.

We also asked our subjects for potential new attributes, attribute mixtures or groups. The most interesting answer was a feature which realises mutually exclusive messages (therefore, if a user is reading message A on place X it is not possible anymore to read message B on place Y). The preferred times for setting up an individual group were 5 minutes (31 percent), 10 minutes (54 percent), 20 minutes (8 percent), the rest decided for the optional answer that the time depends how funny it is creating a group.

An exploratory analysis of server and client logs showed several interesting points; first, that more than 50 percent of the users have restarted the Air-Writing client at least once, with a maximum restart count of 4. The reason for restarting the client lay supposedly not in client errors, but due to the exit button being next to the option button of the N73 model, which could easily be pushed by accident. Second, that the average time

for receiving messages after entering a location was 2.7 (standard deviation 2.7) seconds. Third, that users were switching to different views about 32 times (standard deviation 22.6). Because there are only seven views (Menu, Login, Write, Read, Options, Setup, Exit) this value is rather high and was not expected. For each view switch, approximately 4.5 clicks are needed. This means that 144 clicks were done on average for view switches by the user. We will take this value as a reference for further improvements.

Results of the scavenger hunting game 2. In this test, the focus was on the usability of the iPhone clients. Five out of six participants criticized the "overloaded screen" (Figure 7(a)). The two main menu buttons ("read" and "write") were too big and were interrupting the clicks on the clouds. The problem was that the buttons are – in general – defined as rectangles, but the image of these buttons had a round and "cloud like" shape (shown in Figure 7(a) by the red highlighted rectangle A and B). If a user was trying to click on such a cloud, which was "under" a transparent (and therefore invisible) corner of a read or write button, the read or write clicking action was invoked instead of the action to open the message. This seemed very confusing to many participants. Also, three out of six participants did not fully understand the accessory

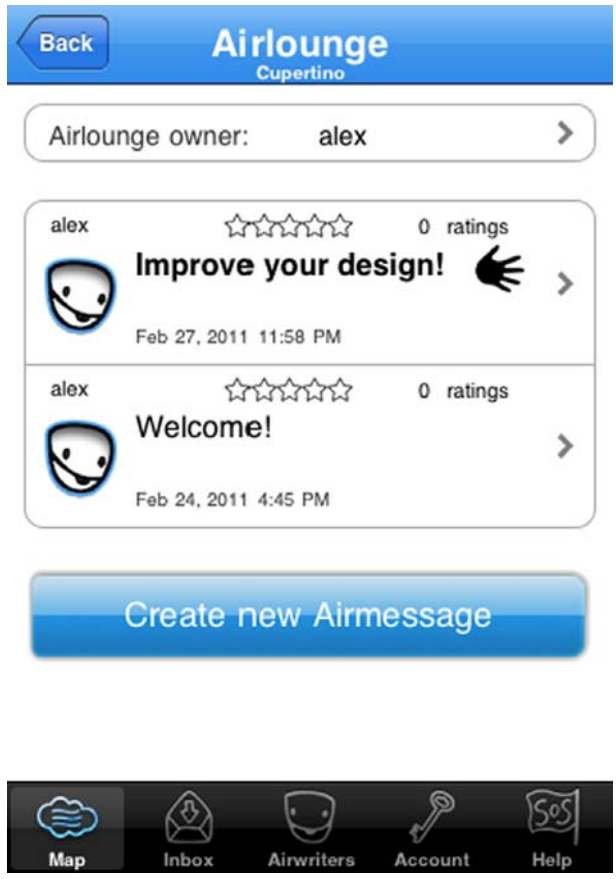

Figure 7.
Pick messages visually highlighted

view concept (Figure 7(a)). This concept is implemented at many popular location applications with map views such as Google and Microsoft maps; if the user clicks on a map object, a preview appears instead of loading the full object information or directly invoking some object specific actions. The reason for this intermediate step is to ensure that all objects are clickable. If two map objects are at the same position, one object is overlapping the other. By clicking on the overlying object a second time, the accessory view of the underlying one becomes visible. We believe that accessory views are the best way to handle overlapping objects, as long as they are not overlapped by some other objects, as it has happened in this user study. Based on these results, a new design concept (Figure 7(b) and (c)) was presented in the form of prototypes to the users and gained an acceptance rate of 100 percent. This new design concept also moves the menu bar to the top, which in the opinion of three participants, was too close to the tab bar menu (shown in Figure 7(a) by the red line labeled C).

Another problem was the pick attribute. Four out of six users were unable to differentiate between normal and pickable messages. It was confusing:

- that an already picked message could disappear; and
- that pick messages are not visually highlighted.

Two out of six users explained in their interview that they suspected client errors and therefore assumed the client not to receive all messages. They were not aware that previous users had already "picked" all available messages within their range, which indicates further usability problems of the more complex attribute types that still need to be studied in more detail. A further design concept, which is indicating pick messages with a special symbol (Figure 8) gained an acceptance rate of 100 percent, although after explaining the concept of messages to the same user group and therefore not being representative in terms of a usability study.

(a) (b) (c)

Figure 8.
Usability improvements

Notes: (a) The original Airwriting iPhone Startscreen; (b) the rearranged Airwriting iPhone Startscreen; (c) the rearranged Airwriting iPhone Startscreen with the hide option enabled

The functionality of the Air-Writing platform strongly depends on a working internet connection. However, user feedback indicates that this dependency should not affect the client as strongly as it does in the actual version from a usability perspective. If there is no internet connection available, the client is currently "useless", as it does not allow any interaction besides checking whether the connection is working or not. Three out of six users complained about that issue.

Some users (two out of six) also experienced problems in receiving messages at all. An analysis based on the recorded data shows that the reason for this fatal problem was a radius value of messages that was set too small. Minimum, maximal, and recommended optimal radius values need to be determined in future empirical studies based on a significantly larger data set from daily use.

Privacy analysis

The subjects in our initial user study intentionally used different values for poll interval and poll range. For a good compromise between privacy protection, guaranteed message delivery, and transfer volume, these parameters depend on the number of messages in the area, the speed of the user and how far the messages are distributed. The primary goal is not to let users pass any messages without receiving them while still protecting their privacy.

Every poll request reveals the rough location of the user to the server (unless this query is a randomised "cloaking" query as explained above). During a request, the location information on the client is sent to the server after it has been randomised (or rounded to be able to use CDNs[12]. Using shorter poll intervals results in sending the location information more often and is therefore disadvantageous in terms of privacy. Longer poll intervals result in better privacy protection, but require the use of wider ranges in order to prevent users from leaving the cached areas and thus missing some messages. The problem with wider ranges is that with every request more messages will be transferred to the client. Mobile devices still have limited resources (memory, CPU, etc.); a large amount of messages may therefore cause performance problems on the client side as well as larger transfer volume on the wireless internet connection (and may thus lead to higher cost).

In our privacy analysis, we have taken the path recorded in our user study (Figure 9(a)) and simulated multiple instances of using the Air-Writing protocol with different combinations of poll intervals and poll ranges. We also generated two extra paths (shown in Figure 9(b)) to analyse indistinguishability of these paths based on the data the server would be able to record in each of the simulations. The resulting measure is comparable to k-anonymity: when all three paths are indistinguishable given the server log, then perfect privacy has been reached in this scenario. In practice, we expect many more concurrent paths and thus better user privacy with even lower poll intervals and ranges. One of these paths leads from north-east down to south-west and the other one from north-west down to south-east. Message positions as used in the game "scavenger hunt" are also indicated in Figure 9(a) and (b). In each simulated instance, the aim was for users to receive all of these messages as soon as they pass the respective location.

Results of the simulation are summarised in Table V for cases in which no messages were missed by the clients. Ideally, every message should be polled only once in order to save bandwidth and client resources. However, because the ranges between two subsequent queries may overlap, some messages may be requested and transferred

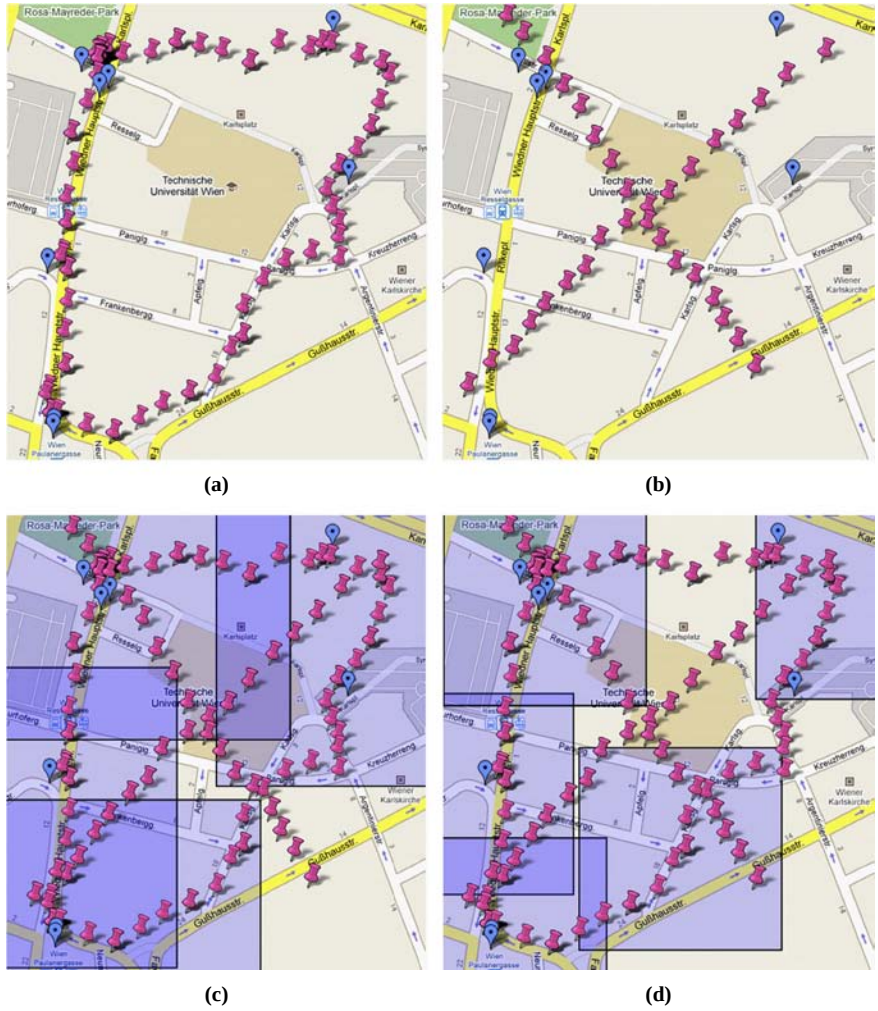

Figure 9.
Logged and simulated location traces

Notes: (a) Messages and real path; (b) messages and simulated paths; (c) polls with range 300 meters and interval 350 s; (d) polls with range 200 meters and interval 300 s

Table V.
The privacy level depends on pool interval and range, but these also influence client resource usage and bandwidth

Range (m)	Pool interval (s)	Overhead	Largest pool	Privacy
700	1,000	0	8	3/3
700	350	19	8	3/3
400	350	6	6	2/3
300	350	2	3	1/3
45	30	14	2	1/3
200	300	0	3	1/3

more than once. Column "overhead" shows the total number of redundant message transfers during the run, while column "largest poll" represents the amount of messages transferred during the biggest poll in that run, which is the query where the highest number of messages have been transferred and indicates the maximum resource consumption on the client.

Finally, column "privacy" shows the level of privacy achieved during the run, represented by the number of paths which an attacker cannot distinguish from the actual user path, given full access to the server logs or by eavesdropping on the communication. With one real and two simulated paths, a privacy value of 3/3 indicates that looking at the communication logs between the client and the server none of the three paths can be distinguished from the others. Figure 9(c) and (d) show the communication log in form of the queried areas in each query and may help to more intuitively optimise parameters in practical settings.

As seen in the first two lines of Table V, using wide ranges for poll requests ensures location privacy, but causes more messages to be transferred within a single poll. The bottom three lines of the table show that we can reduce bandwidth usage by shortening the poll range, but we also lose the privacy protection. This shows a clear trade-off between the privacy level and the required bandwidth or device capacity, which can be configured by each user to account for personal preferences. In the current Air-Writing implementation, group moderators have the responsibility for keeping the number of messages in a given region below a specified limit. This guarantees that poll responses do not exceed the limited capabilities of mobile devices while privacy can be protected using wide poll ranges.

7. Conclusion and future outlook

In this article, we propose to enhance textual messaging services by introducing an abstract, attribute based messaging service architecture. We demonstrate a working implementation with various attributes for three different client device platforms (J2ME, Android, iPhone) with special support for the highly important location attribute, thus supporting spatial messaging applications. Both our architecture and the specific implementations have been designed to protect users' privacy from the start: by polling messages anonymously based on message groups and supported by client caching and filtering as well as active randomisation in time and space, strong privacy protection is enabled by default. Even potential adversaries on the level of mobile internet service providers or server operators would be unable to distinguish locations traces from different users. The results of an initial user study indicate that the concept of our architecture seems clear and accessible, even if the analysed logs show that there is room for many improvements in terms of user experience. One important future extension is the implementation of various forms of client managed attributes for further evaluation studies and more rapid prototyping of additional application scenarios.

Notes

1. http://blog.nielsen.com/nielsenwire/online_mobile/in-us-text-messaging-tops-mobile-phone-calling/ (2008).
2. http://mobithinking.com/sites/mobithinking.com/files/dotMobi_and_AKQA_Mobile_Usage_and_Attitudes_Study.pdf (2008).

3. www.bdnooz.com/lbsn-location-based-social- networking-links (2009).
4. ZYB, a social platform, was recently bought by Vodafone, www.gomonews.com/vodafone-acquires-100-of-zyb/ (2008).
5. www.plazes.com/ (2009).
6. www.jotyou.com/ (2009).
7. www.postgresql.org/ (2008).
8. http://postgis.refractions.net/ (2008).
9. www.hibernate.org/ (2008).
10. www.springframework.org/ (2008).
11. http://tapestry.apache.org/tapestry5/ (2008).
12. For achieving global scalability, we propose the use of CDNs. Messages can be stored and retrieved under standard HTTP URLs which reflect the rounded location in a lattice with hierarchically structured resolution. Rounding is necessary to keep the number of these URLs manageable.

References

Ballagas, R., Kratz, S.G., Borchers, J.O., Yu, E., Walz, S.P., Fuhr, C.O., Hovestadt, L. and Tann, M. (2007), "REXplorer: a mobile, pervasive spell-casting game for tourists", *Proceeding CHI '07 Extended Abstracts on Human Factors in Computing Systems*, ACM Press, New York, NY, pp. 1929-34.

Banerjee, S., Agarwal, S., Kamel, K., Kochut, A., Kommareddy, C., Nadeem, T., Thakkar, P., Trinh, B., Youssef, A., Youssef, M., Larsen, R.L., Udaya Shankar, A. and Agrawala, A. (2002), "Cover feature: rover: scalable location-aware computing", *IEEE Computer*, Vol. 35 No. 10, pp. 46-53.

Boesen, J., Rode, J.A. and Mancini, C. (2010), "The domestic panopticon: location tracking in families", *Proceedings of the 12th ACM International Conference on Ubiquitous Computing, Copenhagen*, ACM Press, New York, NY, pp. 65-74.

Bowen, C.L., Raymond, D.R. and Martin, T.L. (2008), "Location privacy for users of wireless devices through cloaking", *HICSS*, IEEE Computer Society, Washington, DC.

Brush, A.J.B., Krumm, J. and Scott, J. (2010), "Exploring end user preferences for location obfuscation, location-based services, and the value of location", *Proceedings of the 12th ACM International Conference on Ubiquitous Computing, Copenhagen*, pp. 95-104.

Buhan, I., Doumen, J., Hartel, P. and Veldhuis, R. (2007), *Secure Ad-hoc Pairing with Biometrics*, SAfE, University of Twente, Centre for Telematics and Information Technology, Enschede.

Burrell, J. and Gay, G. (2002), "E-graffiti: evaluating real-world use of a context-aware system", *Interacting with Computers*, Vol. 14 No. 4, pp. 301-12.

Counts, S. (2007), "Group-based mobile messaging in support of the social side of leisure", *Work, Comput. Supported Coop*, Kluwer, Norwell, MA.

Dingledine, R., Mathewson, N. and Syverson, P. (2004), "Tor: the second-generation onion router", *13th USENIX Security Symposium*, pp. 303-20.

Froehlich, P., Simon, R., Muss, E., Stepan, A. and Reichi, P. (2007), "Envisioning future mobile spatial applications", *People and Computers XXI, British HCI*, British Computer Society, Swinton.

Gedik, B. and Liu, L. (2008), "Protecting location privacy with personalized k-anonymity: architecture and algorithms", *IEEE Trans. Mob. Comput.*, Vol. 7 No. 1.

Gruteser, M. and Grunwald, D. (2003), "Anonymous usage of location-based services through spatial and temporal cloaking", *MobiSys., USENIX*.

Hazas, M., Kray, C., Gellersen, H., Agbota, H., Kortuem, G. and Krohn, A. (2005), "A relative positioning system for co-located mobile devices", *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*, ACM Press, New York, NY, pp. 177-90.

Kido, H., Yanagisawa, Y. and Satoh, T. (2005), "Protection of location privacy using dummies for location-based services", paper presented at ICDE Workshops.

Krumm, J. (2007), "Inference attacks on location tracks", paper presented at Pervasive Computing, 5th International Conference.

Mayrhofer, R. (2007), "Towards an open source toolkit for ubiquitous device authentication", *5th {IEEE} International Conference on Pervasive Computing and Communications* (*PerCom 2007*), White Plains, NY, pp. 247-54.

Mayrhofer, R., Sommer, A. and Saral, S. (2010), "Air-writing: a platform for scalable, privacy-preserving, spatial", *Proc. üWAS2010: 12th International Conference on Information Integration and Web-based Applications & Services*, ACM.

Narzt, W., Pomberger, G., Ferscha, A., Kolb, D., Müller, R., Wieghardt, J., Hörtner, H., Haring, R. and Lindinger, C. (2007), "Addressing concepts for mobile location-based information services", paper presented at the 12th International Conference on Human-computer Interaction HCI, Beijing, China.

Peng, G. (2004), "CDN: content distribution network", *CoRR*, Vol. cs.NI/0411069.

Persson, P., Espinoza, F., Fagerberg, P., Sandin, A. and C-ster, R. (2002), "GeoNotes: a location-based information system for public spaces", *Readings in Social Navigation of Information Space*, Springer, Berlin.

Rahemtulla, H.A., Haklay, M. and Longley, P.A. (2008), "A mobile spatial messaging service for a grassroots environmental network", *J. Locat. Based Serv.*, Vol. 2 No. 2.

Scipioni, M.P. and Langheinrich, M. (2010), "I'm here! Privacy challenges in mobile location sharing", Second International Workshop on Security and Privacy in Spontaneous Interaction and Mobile Phone Use (IWSSI/SPMU 2010).

Toch, E., Cranshaw, J., Drielsma, P.H., Tsai, J.Y., Kelley, P.G., Springfield, J., Cranor, L., Hong, J. and Sadeh, N. (2010), "Empirical models of privacy in location sharing", *Proceedings of the 12th ACM International Conference on Ubiquitous Computing, Copenhagen*, ACM Press, New York, NY, pp. 129-38.

About the authors

Rene Mayrhofer currently holds a Full Professorship for Mobile Computing at University of Applied Sciences Upper Austria. Previously, he held a Guest Professorship for Mobile Computing at University of Vienna, Austria, during which he received his Venia Docendi for Applied Computer Science. His research interests include computer security, ubiquitous computing, and machine learning, which he brings together in his research on intuitive and unobtrusive techniques for securing spontaneous interaction. He received his Dipl.-Ing. (MSc) and Dr techn. (PhD) degrees from Johannes Kepler University Linz, Austria, and subsequently held a Marie Curie Fellowship at Lancaster University, UK. Rene Mayrhofer is the corresponding author and can be contacted at: rene@mayrhofer.eu.org

Alexander Sommer is a Mobile Software Developer, who is developing mobile applications for some of the biggest companies in Austria. He is working at Evolaris Next Level GmbH, an independent research and development institution for interactive media in Austria. He received his Dipl. -Ing. (MSc) degree from Technical University Vienna, Austria. His research interests are mobile social applications, interactive media and business model design for mobile applications. Besides working at the project "Airwriting", Alexander Sommer is also developing an interactive eBook reader application (www.scrollstory.com).

Sinan Saral is a Software Developer living in Austria, Vienna. He is a member of the Technical University of Vienna, Austria and received his Bachelor (BSc) degree from this university. He is an active web and mobile applications developer and runs one of the biggest car-sharing web sites in Turkey. He primarily works on mobile applications, web applications and functional programming languages.