Efficient Oblivious Augmented Maps: Location-Based Services with a Payment Broker

Markulf Kohlweiss[1], Sebastian Faust[1], Lothar Fritsch[2], Bartek Gedrojc[3], Bart Preneel[1]

[1] K.U. Leuven ESAT/COSIC, Kasteelpark Arenberg 10
3001 Leuven, Belgium
[2] Johann Wolfgang Goethe-Universität
60054 Frankfurt am Main, Germany
[3] TU Delf ICT, Mekelweg 4
2628 CD Delft, The Nederlands

Abstract. Secure processing of location data in location-based services (LBS) can be implemented with cryptographic protocols. We propose a protocol based on oblivious transfer and homomorphic encryption. Its properties are the avoidance of personal information on the services side, and a fair revenue distribution scheme. We discuss this in contrast to other LBS solutions that seek to anonymize information as well as possible towards the services. For this purpose, we introduce a proxy party. The proxy interacts with multiple services and collects money from subscribing users. Later on, the proxy distributes the collected payment to the services based on the number of subscriptions to each service. Neither the proxy nor the services learn the exact relation between users and the services they are subscribed to.

1 Introduction

Does my electronic map need to know where I am – or that I am looking at it? Electronic maps can be augmented with information provided by location-based services (LBS). This way subscribed users can find what they need fast. With LBSs, location privacy is at stake. To reach good privacy, it is advisable to limit access to identity information and access to location information. Even the regular observation of service usage patterns might reveal private information.

Today LBSs are provided in one of two ways. Either all the service specific data is made available to the user who computes the result locally—this is the case for car navigation systems; or the service is provided remotely. The latter is the dominant approach for providing LBSs in mobile communication networks. Such LBSs can be seen as a by-product of the GSM system (Global System for Mobile Communications), as the location of subscribers is already used for mobility management [17]. Given these constraints, we aim at achieving privacy for mobile subscribers who want to use LBSs. We presume that the location will be gathered from a mobile network, while the service will be provided by external service providers.

Our approach uses cryptographic protocols to ensure privacy: Oblivious transfer and homomorphic encryption. By developing a framework where the user's location and subscription are processed in the encrypted domain, we achieve privacy for certain classes of LBSs in a new way. Unlike classic approaches using Mixes or anonymous credentials [19], our approach achieves the following strong privacy properties: First, the mobile operator learns nothing in addition to what he already knows except for the set of users that are at all interested in using LBSs. Thus, no usage profiles can be collected. Second, the service providers only learn the number of subscribers to their service. Thus, service providers do not learn the users' location.

Our protocol offers the additional privacy property of service unobservability. Even the service providers do not know whether a user is accessing their service or not. By introducing a privacy trustee we are able to preserve service unobservability in case of service/operator collaboration.

The privacy provided is optimal: the operator needs to know the set of LBSs users such that he can localize and charge them, and services are payed by the operator depending on the number of subscribers they were able to attract.

Efficiency consideration. Despite the strong security requirements, our protocol scales well: The initialization is independent of the number of users. Subscriptions are linear both in effort and in size in the number of services. This is unavoidable, as all services need to be involved to guarantee service unobservability. The mobile user as the party with the most restricted resources has to receive and decrypt only a single message.

Related Work. Various privacy enhancing technologies (PET) have been proposed for LBSs. Most of these techniques focus on providing pseudonymity and anonymity for LBSs. Federath *et al.* [13] proposed the use of a trusted fixed station and Mixes [9] for hiding the relation of real world identities to location data in GSM networks. While our protocol can be adapted to such a privacy enhanced GSM network by letting the fixed station localize the user (and take over some of the responsibilities of the mobile operator), we explicitly focus on the less privacy friendly but more practical setting where a third party knows the user's location.

Researchers started to develop LBS specific PETs called mix-zones (see [5] and [16]). Mix-zones allow users to switch pseudonyms associated to their location in an unobservable way. Kölsch *et al.* [19] use pseudonymization techniques in the following realistic setting. A network operator (or a party connected to multiple operators) knows the user's location, while the LBSs are provided by independent service providers that know the user only under short lived pseudonyms. Basically, as location information is inherently attached to a person's life, reidentification is often easy [15]. Location needs to be hidden, not anonymized.

Structure of the paper. Section 2 considers cryptographic tools. We describe the properties and high level implementation of our privacy protecting LBS scheme in Sec. 3. The detailed construction is given in Sec. 4. We analyze the efficiency and security of our solution in Sec. 5 and finally conclude in Sec 6.

2 Tools

Zero-Knowledge Proofs of Knowledge. A *zero-knowledge* proof is an interactive proof in which the verifier learns nothing besides the fact that the statement proven is true. Zero-knowledge proofs-of-knowledge protocols exist for proving various statements about discrete logarithms in groups of known and hidden order [4, 6, 8, 26]. These techniques allow to prove statements about cryptographic primitives that operate in these groups, for instance that two commitments contain the same value, or that a value was verifiably encrypted. Given a statement $\mathsf{Alg}(x) = y$ and $\mathsf{Alg}'(x') = y'$ about two algorithms, with secret input x, x' and public output y, y', it is possible to prove AND and OR relations of these statements. Such protocols can be made non-interactive by applying a cryptographic trick called the Fiat-Shamir heuristic [14]. We write in a short form notation, e.g., for AND

$$\pi = PK\{(x, x') : \mathsf{Alg}(x) = y \wedge \mathsf{Alg}'(x') = y'\}.$$

Homomorphic Encryption. Homomorphic encryption is a form of malleable encryption. Given two ciphertexts, it is possible to create a third ciphertext, with a plain text that is related to the first two. For (additive) homomorphic encryptions, the encrypted plain texts fulfill the following relations:

$$\mathsf{Enc}_h(m_1) \oplus \mathsf{Enc}_h(m_2) = \mathsf{Enc}_h(m_1 + m_2), \qquad c \otimes \mathsf{Enc}_h(m) = \mathsf{Enc}_h(c \cdot m).$$

We speak of additive homomorphic encryption because $+$ corresponds to the addition operation of a ring. We write $c \otimes \mathsf{Enc}_h(m)$ to denote the c times homomorphic addition of $\mathsf{Enc}_h(m)$. Note that for Damgård-Jurik Encryption [12] $c \otimes \mathsf{Enc}_h(m)$ corresponds to $\mathsf{Enc}_h(m)^c$ and can be implemented efficiently.

Damgård-Jurik Encryption. The Damgård-Jurik cryptosystem is a generalization and adaptation of the Pallier cryptosystem [23] based on the decisional composite residuosity assumption. It allows for the encryption of arbitrary long messages without the need to generate new keys. It preserves the homomorphic property of Pallier. They also describe the zero-knowledge proofs and threshold decryption techniques required by our protocol.

Threshold encryption. In a distributed decryption protocol a private key is shared among a group of parties, where only a qualified subset of the parties is allowed to decrypt a ciphertext c, whereas fewer parties learn nothing on the secret nor on the decryption of c. In our scheme we use the special case of a distributed 3-out-of-3 threshold encryption scheme, which could be implemented, e.g., with the threshold protocol presented in [12].

Oblivious Transfer. Oblivious transfer (OT) was first introduced by Rabin [25]. The primitive captures the notion of a protocol by which a sender sends some information to the receiver, but remains oblivious as to what is sent.

Adaptive OT from Blind Signatures. For LBSs, we are not so much interested in single executions of OT, but want to query the same database multiple times at different indices (for different users and changing locations). Adaptive OT protocols were proposed in [11, 20, 21]. Camenisch et al. [7] recognized that the schemes in [11, 21] are based on a common principle to construct adaptive OT from unique blind signature schemes (UBSS). The first UBSS is described by Chaum in [10].

We briefly sketch the basic idea of adaptive OT schemes based on UBSS. First, all messages $m_1, \ldots, m_n$ are symmetrically encrypted using the hashed signature of the index i, $1 \leq i \leq n$, as the key. Thus $C_i = \mathsf{Enc}_s(m_i; H(\mathsf{Sign}(i; sk)))$. H is a symmetric hash function, Enc_s is a secure symmetric cipher, Sign is the signature algorithm corresponding to the UBSS, and sk is the signing key of the sender that will be used for creating the blind signature. The ciphertexts $C_1, \ldots, C_n$ are transferred to the receiver. To obtain message $m_{\hat{\imath}}$, the receiver runs a blind signature protocol with the sender on message $\hat{\imath}$ to obtain the symmetric key (signature on $\hat{\imath}$).

OT using Homomorphic Encryption. It is a known property of additive homomorphic encryption that given an encryption $Q = \mathsf{Enc}_h(1)$ it is possible to compute an encryption of a message m as $m \otimes Q = \mathsf{Enc}_h(m \cdot 1) = \mathsf{Enc}_h(m)$. However, if $Q = \mathsf{Enc}_h(0)$, the same operation does not change anything, i.e., $m \otimes \mathsf{Enc}_h(0) = \mathsf{Enc}_h(0)$ [22].

Given the semantic security of the encryption, the party trying to encode the message cannot distinguish the two cases above. Based on this observation an OT can be constructed by using a vector $\boldsymbol{Q} = (Q_1, \ldots, Q_\ell)$. To request message $m_{\hat{\jmath}}$, $Q_{\hat{\jmath}} = \mathsf{Enc}_h(1)$ and $Q_j = \mathsf{Enc}_h(0)$ for $j \neq \hat{\jmath}$. Zero-knowledge proofs can be used to prove the correct construction of Q. The communication complexity of the protocol can be reduced by computing $E = \bigoplus_{j=1}^{\ell} m_j \otimes Q_j$, and transferring only E to the recipient.

3 Privacy Protected LBS Scheme: Security Definition and Solution Sketch

3.1 Definition

Parties. Our protocol involves a user $\mathcal{U}$ who accesses LBSs over her mobile device. Her goal is to obtain location specific information on topics of her interest. This information is collected and served by service providers $\mathcal{L}_1, \ldots, \mathcal{L}_\ell$. A third party that knows the user's location and stands in a financial relationship with the user acts as a proxy $\mathcal{P}$ between users and services — this could be the mobile operator of the user or an organization associated with it. The proxy is responsible for the security of the location information and assists in the payment transaction. We assume that the number of users connected over a proxy is much higher than the number of services. Finally, we assume the existence of an independent party without any commercial interests: a privacy protection organization $\mathcal{T}$ that can be offline for most of the time. We refer to all parties except users as organizations.

Security and Privacy Requirements. A secure and privacy friendly LBS protocol should protect the assets and interests of all involved parties. The assets that need to be protected are: the user's location, the user's subscription, the topic specific databases of the $\mathcal{L}_j$, and the payment. We consider the following requirements:

– *Location privacy:* The protocol does not reveal the user's location to the service.
– *Service usage privacy:* Even when the proxy and the LBSs collude, the secrecy of the user's subscription remains protected. This includes message privacy; i.e., only the users can decrypt the messages of services.
– *Database secrecy:* The user and the proxy get no information about the topic specific database of $\mathcal{L}_j$. A user gets only the information for the locations she requested. This property must hold even if the proxy and the proxy collude.
– *Fairness:* It is guaranteed that either the user receives the expected data for the requested location and the LBS receives his expected payment, or the cheating party can be uniquely identified. In order to preserve service usage privacy, the user reveals the cheating party only to the trustee $\mathcal{T}$.

Protocol phases. In the *Setup* phase the involved parties generate their keys. During the *Service Update* phase, each service $\mathcal{L}_j$ encrypts its topic specific database and transfers it to the proxy. In the *Subscription* phase a user $\mathcal{U}$ creates an encrypted subscription for a service, sends it to the proxy, and is charged the subscription fee. In the *Data Retrieval* phase the proxy runs a protocol with every service $\mathcal{L}_j$ and obtains an encrypted result. The proxy combines them into a single encrypted result for the user such that she only receives the data of the subscribed service. The fair allocation of the money collected in the subscription phase takes place in the *Settlement* phase under the supervision of the trustee $\mathcal{T}$.

Remarks. The database of a service $\mathcal{L}_j$ is represented as a one-dimensional vector with one element for each location. We assume that the number of locations n is the same for all services. Further, we assume that services only update the whole database at once. In the current version of our protocol a user is only subscribed to a single service. Service usage privacy is guaranteed with respect to the total number of users that subscribed during a subscription period. A subscription period is defined as the time between two settlement phases. Finally, we assume that parties communicate over secure channels and that $\mathcal{P}$, $\mathcal{L}_j$, and $\mathcal{T}$, are able to authenticate communication, and to sign messages using their identity.

3.2 High-Level Approach and First Sketch

We follow a constructive approach in the description of our protocol. We use building blocks from Sec. 2, put them into place, and describe their function in our construction. Some of the security requirements can be fulfilled by the functionality provided by individual building blocks; others require a complex interplay between building blocks. As a consequence the mapping from building blocks to the sub-protocols of our solution is not one-to-one. We will sketch the sub protocols (cf. Fig. 1) as they get assembled from their building blocks.

Our main building blocks are two variants of OT and a threshold encryption scheme. Homomorphic encryption and zero-knowledge protocols serve as sub - building blocks in the previous schemes, but are also used to glue them together in a secure way. The two OT protocols are specifically selected for their good performance under repetition of input data. The blind signature based OT scheme (cf. Sec. 2) is optimized for the case that the input database remains fixed, while the index varies. The homomorphic encryption based OT is efficient in the opposite case; it is efficient for fixed indices.

During the protocol execution, a single proxy interacts with a multitude of users and multiple services. The first building block we put into place is a blind signature based OT protocol. It is executed with the proxy acting as the requester and one of the services as the sender. It allows the proxy to retrieve location specific information $m_{\hat{\imath},j}$ for a user at location $\hat{\imath}$ without service L_j learning the user's location. This guarantees *location privacy.* The proxy executes this sub-protocol with all offered services. This assures *service privacy* at the service side. In this way the proxy obtains an information vector $m_{\hat{\imath},1}, \ldots, m_{\hat{\imath},\ell}$.

Our second building block is a homomorphic encryption based OT protocol (cf. Sec. 2). It is run with the proxy acting as the sender (using the aforementioned vector as input) and the user acting as the requester (using the index of the service $\mathcal{L}_{\hat{\jmath}}$ she subscribed to as input). The protocol allows the user to learn $m_{\hat{\imath},\hat{\jmath}}$ without the proxy learning the user's subscription; we achieve full *service privacy.*

Note how the choice of OT protocols is crucial for the performance of our protocol. In the first OT, the same database is queried by the proxy for all users (and different locations as they move about). The database needs to be encrypted and transferred to the proxy only once (cf. Fig. 1.2). For the second OT between user and proxy, the subscribed service is invariant for the duration of a subscription period and it is sufficient to send the first (and expensive) message of the homomorphic OT only once (cf. Fig 1.3 ①). Consequently we split off these operations as sub-protocols which have the semantic of a service update and a user's subscription.

This gives us a first instantiation of the first 4 protocol phases. The outline of the protocol is depicted in Fig. 1. Note that some of the sub protocols are not yet implemented. For ease of presentation we use a simplified notation. We refer the reader to App. A for a more formal definition of our LBS scheme. The detailed protocol description is given in Section 4.

Setup. (cf. Fig. 1.1: ① KeygenU, ② KeygenL) Every user generates a key-pair for a homomorphic encryption scheme ①. These keys are used for the OT based on homomorphic encryption. Every service generates a key-pair (skB, pkB) that is used for OT based on blind signatures ②.

Service Update. (cf. Fig. 1.2: ② EncryptData) The database of the LBS $\mathcal{L}_j$ consists of the n elements $m_{(1,j)}, \ldots, m_{(n,j)}$ ①. Each of the elements is encrypted with its own symmetric key $H(k_i)$ that is computed by hashing the signature $k_i = \mathsf{Sign}(i; skB)$ of the index ②. The encrypted database $DB_j = (C_1, \ldots, C_n)$, with $C_i = \mathsf{Enc}_s(m_i, H(k_i))$ is sent to the proxy ③.

Subscription. (cf. Fig. 1.3: ① Subscribe) A user's subscription ① consists of ℓ elements $S_{(U,1)}, \ldots, S_{(U,\hat{\jmath})}, \ldots, S_{(U,\ell)}$, one for each service ②. Each element contains a ciphertext Q of the homomorphic encryption scheme. Q decrypts to 1 for the service $\mathcal{L}_{\hat{\jmath}}$ the user subscribes to and to 0 otherwise. To ensure the security of the OT the user proves in zero-knowledge that all $S_{(U,j)}$ are correctly constructed.

Data Retrieval. (cf. Fig. 1.4: ① Request, ② Combine, ④ Decrypt) In the data retrieval phase a user obtains location-specific data from her subscribed service. The proxy is involved since he is aware of the user's location and stores the encrypted databases of the services. Recall that these databases are encrypted using hashed signatures as keys. The proxy acts on the user's behalf and can request decryption of individual items without revealing the location of the user. To guarantee service usage privacy the proxy has to repeat the following steps for every service $\mathcal{L}_j$ ①:

The proxy blinds the location $\hat{\imath}$ and sends the blinded value $\mathsf{Blind}(\hat{\imath}; b, pkB)$ to the service. The service replies with the blinded signature $\langle k_{\hat{\imath}} \rangle_{blind}$. The proxy computes $m_{\hat{\imath},j} = \mathsf{Dec}_s(C_{\hat{\imath}}; H(\mathsf{Unblind}(\langle k_{\hat{\imath}} \rangle_{blind}; b, pkB)))$. This completes the first OT. The proxy collects $m_{\hat{\imath},1}, \ldots, m_{\hat{\imath},\ell}$ and continues with the second OT (the user's first message is taken from her subscription). The proxy takes the Q corresponding to $S_{(U,j)}$ and computes $E_j = m_{\hat{\imath},j} \otimes Q$ for all $1 \leq j \leq \ell$.This corresponds to an encryption of $m_{\hat{\imath},\hat{\jmath}}$ for $\mathcal{L}_{\hat{\jmath}}$ and an encryption of 0 otherwise.

As a last step the proxy combines the E_j by homomorphically adding all the encryptions (not knowing which of them contain the message) ②. This way all encryptions of 0 cancel out. The result is transferred to the user ③. She decrypts E to obtain $m_{\hat{\imath},\hat{\jmath}}$ ④.

The two main flaws of this construction are (1) the fact that the proxy learns the $m_{i,j}$ vector for the locations of all users. This is a compromise of *database secrecy* and (2) the lack of a fair payment infrastructure.

3.3 First Revision: Database Secrecy

We address the lack of *database secrecy* by intertwining the first OT with the second. To this end we let the proxy pass on $S_{(U,j)}$ to $\mathcal{L}_j$. Now (after agreeing on who sends which bit range) both $\mathcal{L}_j$ and the proxy can act as senders in the second OT without learning each others inputs. This is made possible by the properties of homomorphic encryption, which lets everyone manipulate encrypted data. Informally, the last message of the first OT will be transferred as part of the encrypted payload of the second OT. This guarantees that only the user with her secret decryption key can obtain the results of both protocols.

More concretely the following changes have to be made in the subscription and data retrieval phases.

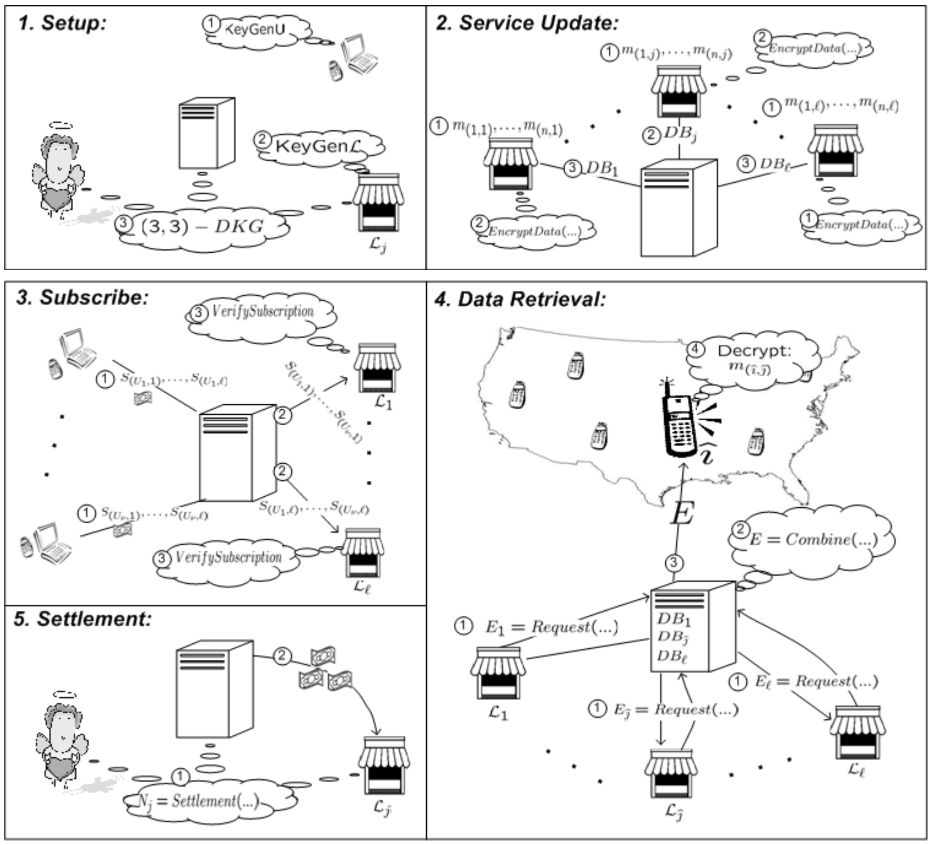

Fig. 1. Setup and Service Update, Subscription, Data Transfer, and the Settlement phase: Subscription $S_{(\mathcal{U},j)}$, encrypted database DB_j, service result E_j, combined result E, location-specific message $m_{(\hat{\imath},\hat{\jmath})}$, number of subscriptions N_j, location $\hat{\imath}$, and the subscribed service $\hat{\jmath}$.

Subscribe. The $S_{(U,j)}$ are now also sent to the services ②.

Data Retrieval. During Request ① the proxy blinds the location $\hat{\imath}$ and sends the blinded value $\mathsf{Blind}(\hat{\imath}; b, pkB)$ to the service. To ensure that only the user (and not the proxy) can decrypt $C_{\hat{\imath}}$, the service encrypts the blinded signature $\langle k_{\hat{\imath}}\rangle_{blind}$. This is done with an additive homomorphic encryption scheme. Remember that during subscription the user (through the proxy) provided the service $\mathcal{L}_{\hat{\jmath}}$ with an encryption $Q = \mathsf{Enc}_h(1)$. The service computes $E_{\hat{\jmath}} = \langle k_{\hat{\imath}}\rangle_{blind} \otimes Q = \mathsf{Enc}_h(\langle k_{\hat{\imath}}\rangle_{blind} \cdot 1) = \mathsf{Enc}_h(\langle k_{\hat{\imath}}\rangle_{blind})$. The result is sent to the proxy who uses a similar approach to add b and $C_{\hat{\imath}}$ to $E_{\hat{\jmath}}$. These requests are done for all services, including those the user did not subscribe to. The latter however received $Q = \mathsf{Enc}_h(0)$ during Subscribe and all the operations result in $E_j = \mathsf{Enc}_h(0)$, for $j \neq \hat{\jmath}$.

As a last step the proxy computes the homomorphic sum of all encryptions—not knowing which of them contain the unblinding information, the encrypted message, and the blinded signature ②. This way all encryptions of 0 cancel out. The result is transferred to the user ③. She decrypts E, obtains $b \| C_{\hat{\imath}} \| \langle k_{\hat{\imath}} \rangle_{blind}$, and computes $m_{\hat{\imath}\hat{\jmath}} = \mathsf{Dec}_s(C_{\hat{\imath}}; H(\mathsf{Unblind}(\langle k_{\hat{\imath}} \rangle_{blind}; b, pkB)))$ ④.

3.4 Second Revision: Payment Infrastructure

The core idea for the payment infrastructure is to bind the request of the second OT (the subscription) to a vote. Now revenues can be fairly distributed between services by anonymously counting the number of times users voted for (subscribed to) a service. We use ballot counting techniques based on homomorphic encryption and threshold decryption. We make the following changes to the setup and subscription phase, and we provide an implementation for the settlement phase.

Setup. (cf. Fig. 1.1: ③ PaymentSetup) Each LBS $\mathcal{L}_j$ runs a distributed key generation protocol together with the proxy and the privacy trustee ③. This results in a key pair with a secret key shared according to a $(3, 3)$-threshold scheme. The shared key is needed in the settlement phase to jointly compute the payment result.

Subscription. (cf. Fig. 1.3: ① Subscribe, ③ VerifySubscription) A user's subscription ① consists of ℓ elements $S_{(U,1)}, \ldots, S_{(U,\hat{\jmath})}, \ldots, S_{(U,\ell)}$, one for each service ②. Each element contains two ciphertexts Q and P of the homomorphic encryption scheme, where the first is encrypted with the user's public key and the latter with the payment key. Both Q and P decrypt to 1 for the service $\mathcal{L}_{\hat{\jmath}}$ the user subscribes to, and to 0 otherwise. To ensure the security of the OT and the payment, $\mathcal{U}$ proves in zero-knowledge that Q and P are constructed correctly. The service providers check these proofs before providing the service ③.

Settlement. (cf. Fig. 1.5: ① Settlement) The technique used in the Settlement phase is similar to a technique used in electronic voting protocols. The non-interactive zero-knowledge proof sent by the user in the subscription ensures that P and Q encrypt the same value (either 1 or 0). The homomorphic property of the ciphertexts allows to anonymously sum up the content of all different P values. The trustee $\mathcal{T}$ ensures that only the homomorphic sums (and not individual subscriptions) are decrypted in a 3-out-of-3 threshold decryption ①. Based on the result the proxy divides the subscription money received from the users during subscription in a fair way ②.

4 Our Multi-Party Proxy LBS Scheme

Notation. We write cryptographic primitives as $\mathsf{Alg}(x; k)$, where x denotes the processed inputs of the algorithm and k denotes keys, randomness, or public parameters. When it is clear from the context k is omitted. We use the

$\mathsf{Alg}(\mathcal{E}_1(x_1; k_1), \mathcal{E}_2(x_2; k_2))$ to denote an interactive algorithm between entities $\mathcal{E}_1$ and $\mathcal{E}_2$ with the respective inputs and keys.

Length Parameters. Let κ be a security parameter that determines the key sizes of the underlying cryptographic schemes. We use $l_N \in \Theta(\kappa)$ to denote the length of the RSA modulus N used for Damgård-Jurik encryption. The length of a ciphertext for a plaintext of length N^s is N^{s+1}. For simplicity we use ciphertexts of length $l_N(s+1)$, to encode plaintexts of length $(l_N - 1)s$. We use l_H to denote the length of the plaintext for the homomorphic encryption scheme. Let l_B, l_b, and l_D be the length of a blinded signature, the blinding factor, and an encrypted database entry respectively. We require $l_B + l_b + l_D \leq l_H$.

Setup. $\mathsf{PaymentSetup}(\mathcal{L}_j(1^\kappa), \mathcal{P}(1^\kappa), \mathcal{T}(1^\kappa))$ is executed for each service $\mathcal{L}_j$. The privacy trustee $\mathcal{T}$, the proxy $\mathcal{P}$ and the LBS $\mathcal{L}_j$ run a distributed key generation protocol to generate a public payment key pkS_j. The secret key skS_j is shared according to a $(3, 3)$-threshold encryption scheme such that only the three parties together can reconstruct the key. This results in three secret shares $skS_{(\mathcal{L}_j, j)}$, $skS_{(\mathcal{P}, j)}$, and $skS_{(\mathcal{T}, j)}$, which are included in the secret key of $\mathcal{P}$, $\mathcal{L}_j$, and $\mathcal{T}$ respectively.

LBS Key Generation. Every service $\mathcal{L}_j$ runs $\mathsf{KeygenL}(1^\kappa)$ to generate the keypair $(pk_{\mathcal{L}_j}, sk_{\mathcal{L}_j})$ that contains amongst others a key pair (pkB_j, skB_j) for a unique blind signature protocol.

Proxy Key Generation. For our construction the proxy does not need to generate keys on his own. The public key $pk_\mathcal{P}$ of $\mathcal{P}$ results from adding the public payment keys of the services. Hence $pk_\mathcal{P}$ contains $pkS_1, \ldots, pkS_\ell$. His secret key $sk_\mathcal{P}$ contains the corresponding secret shares $skS_{(\mathcal{P},1)}, \ldots, skS_{(\mathcal{P},\ell)}$.

User Key Generation. $\mathsf{KeygenU}(1^\kappa)$ generates a key pair $(pk_\mathcal{U}, sk_\mathcal{U})$ for an additive homomorphic Damgård-Jurik cryptosystem [12] used for homomorphic OT [22].

Service update. Each LBS $\mathcal{L}_j$ encrypts its location specific information using algorithm $\mathsf{EncryptData}(m_{(1,j,v)}, \ldots, m_{(n,j,v)}, v; sk_{\mathcal{L}_j})$. The value v denotes the version number of the data update. Note that $m_{(i,j,v)}$ contains i,j, and v and is signed by $\mathcal{L}_j$ to allow for checks of authenticity. A service $\mathcal{L}_j$ uses his secret key skB_j to compute $DB_{(j,v)} = (C_{(1,j,v)}, \ldots, C_{(n,j,v)}, v)$:

$$\begin{aligned} k_{(i,j,v)} &= \mathsf{Sign}(i \| v; skB_j) \\ C_{(i,j,v)} &= \mathsf{Enc}_s(m_{(i,j,v)}; H(k_{(i,j,v)})). \end{aligned}$$

The cryptographic hash function H is used for computing the symmetric key. Note that in Sign the values i and v need to be interpreted as fixed length bit strings. The resulting database $DB_{(j,v)}$ is sent to the proxy.

Subscription. User $\mathcal{U}$ must subscribe to a service $\mathcal{L}_{\hat{\jmath}}$ to receive location related information from him. This is done by running the protocol $\mathsf{Subscribe}(\mathcal{P}(sk_\mathcal{P}, pk_\mathcal{U}), \mathcal{U}(\hat{\jmath}; sk_\mathcal{U}, pk_\mathcal{P}))$ together with the proxy. The proxy's public key is parsed as $pk_\mathcal{P} = (pkS_1, \ldots, pkS_\ell)$ and the user proceeds as follows:

1. $\mathcal{U}$ uses her public key $pk_{\mathcal{U}}$ to compute subscription elements $S_{(\mathcal{U},j)}$, $1 \le j \le \ell$:

$$S_{(\mathcal{U},j)} = (Q_{(\mathcal{U},j)}, P_{(\mathcal{U},j)}, \pi_j) \text{ where}$$

$$Q_{(\mathcal{U},j)} = \begin{cases} \mathsf{Enc}_h(1; pk_{\mathcal{U}}) & \text{if } j = \hat{\jmath} \\ \mathsf{Enc}_h(0; pk_{\mathcal{U}}) & \text{otherwise} \end{cases}, \quad P_{(\mathcal{U},j)} = \begin{cases} \mathsf{Enc}_h(1; pkS_j) & \text{if } j = \hat{\jmath} \\ \mathsf{Enc}_h(0; pkS_j) & \text{otherwise} \end{cases}$$

$$\begin{aligned}\pi_j = SPK\{(r_1, r_2) :\\ &(Q_{(\mathcal{U},j)} = \mathsf{Enc}_h(1; r_1, pk_{\mathcal{U}}) \wedge P_{(\mathcal{U},j)} = \mathsf{Enc}_h(1; r_2, pkS_j)) \vee \\ &(Q_{(\mathcal{U},j)} = \mathsf{Enc}_h(0; r_1, pk_{\mathcal{U}}) \wedge P_{(\mathcal{U},j)} = \mathsf{Enc}_h(0; r_2, pkS_j)\}\end{aligned}$$

The $Q_{(\mathcal{U},j)}$ are used to request the location specific information from the LBS $\mathcal{L}_{\hat{\jmath}}$ and the $P_{(\mathcal{U},j)}$ are used for the oblivious payment.
2. The resulting $S_{(\mathcal{U},1)}, \dots, S_{(\mathcal{U},\ell)}$ are sent to the proxy together with the payment for the subscription, e.g., by using a credit card.
3. Additionally, the user proves in zero-knowledge that the homomorphic sum of the values $Q_{(\mathcal{U},j)}$ is an encryption of 1. This can be done using standard techniques from [12]. See also Sec. 2.
4. If the verification of the the last proof and of the individual π_j proofs succeeds, the proxy adds a time stamp to each $S_{(\mathcal{U},j)}$ and signs it.

The proxy passes each subscription $S_{(\mathcal{U},j)}$ on to the respective service $\mathcal{L}_j$. He keeps a counter of the number of user subscriptions in this subscription period. Optionally the proxy may also retain all $S_{(\mathcal{U},j)}$.

Verify Subscription. Service $\mathcal{L}_j$ runs $\mathsf{VerifySubscription}(S_{(\mathcal{U},j)}, j; pk_{\mathcal{U}}, pk_{\mathcal{P}})$ to verify that $S_{(\mathcal{U},j)}$ is correct, i.e., that the content of the queries $Q_{(\mathcal{U},j)}$ and $P_{(\mathcal{U},j)}$ are equal and encryptions of 0 or 1, and that the proxy cannot deny that the user has subscribed. The first is done by verifying the proofs of knowledge, the latter by verifying the proxy's time stamp and signature.

If the algorithm succeeds, $S_{(\mathcal{U},j)}$ is added to a list of subscriptions S_j. The $P_{(\mathcal{U},j)}$ of S_j will later on be added up using the homomorphic property of the underlying encryption scheme Enc_h. If one of the verifications done by the services $\mathcal{L}_j$, $1 \le j \le \ell$ fails, they refuse to provide the information and the proxy has to refund the payment for the subscription to the user.

Data retrieval. The proxy runs $\mathsf{Request}(\mathcal{P}(DB_{(j,\hat{v})}, \hat{\imath}; sk_{\mathcal{P}}, pk_{\mathcal{L}_j}), \mathcal{L}_j(S_{(\mathcal{U},j)}; sk_{\mathcal{L}_j}, pk_{\mathcal{U}}, pk_{\mathcal{P}}))$ with $\mathcal{L}_j$ to request location specific information for user $\mathcal{U}$.

The input of the algorithm is the database $DB_{(j,\hat{v})}$ with most up-to-date version $\hat{v}$ and the current location of the user $\hat{\imath}$. The proxy's output is either an encryption of $m_{(\hat{\imath},j,\hat{v})}$ based on the location of the user or an encryption of 0 if $\mathcal{U}$ is not subscribed to $\mathcal{L}_j$.

1. The proxy chooses a random b and computes

$$\langle \hat{\imath} \| \hat{v} \rangle_{blind} = \mathsf{Blind}(\hat{\imath} \| \hat{v}; pkB_j, b).$$

The random blinding factor b hides the location $\hat{\imath}$ of the user in $\langle \hat{\imath} \| \hat{v} \rangle_{blind}$.
2. The proxy sends $\langle \hat{\imath} \| \hat{v} \rangle_{blind}$ to $\mathcal{L}_j$, $1 \le j \le \ell$, which computes

$$E_j = \langle k_{(\hat{\imath},j,\hat{v})} \rangle_{blind} \otimes Q_{(\mathcal{U},j)}, \quad \text{where } \langle k_{\hat{\imath},j,\hat{v}} \rangle_{blind} = \mathsf{Sign}(\langle \hat{\imath} \| \hat{v} \rangle_{blind}; skB_j)$$

3. Every service $\mathcal{L}_j$ sends E_j back to the proxy.
4. The proxy enriches E_j with $C_{(\hat{\imath},j,\hat{v})}$ and b. This is done by computing $E_j := E_j \oplus \big((C_{(\hat{\imath},j,\hat{v})} \| b) \ll l_B\big) \otimes Q_{(\mathcal{U},j)}$. Where $\ll$ denotes shifting to the left. This only changes the content of E_j if $Q_{(\mathcal{U},j)}$ is an encryption of 1.

Combining. After running $\mathsf{Request}$ with every $\mathcal{L}_j$ and receiving the corresponding E_j, the proxy runs $\mathsf{Combine}(E_1, \ldots, E_l, sk_{\mathcal{P}}, pk_{\mathcal{U}})$ to compute $E = \bigoplus_{j=1}^{\ell} E_j$.
Decrypting. The user decrypts E using $\mathsf{Decrypt}(E; sk_{\mathcal{U}}, pk_{\mathcal{L}_{\hat{\jmath}}})$:

$$\begin{aligned} C_{(\hat{\imath},\hat{\jmath},\hat{v})} \| b \| \langle k_{(\hat{\imath},\hat{\jmath},\hat{v})} \rangle_{blind} &= \mathsf{Dec}_h(E; sk_{\mathcal{U}}) \\ k_{(\hat{\imath},\hat{\jmath},\hat{v})} &= \mathsf{Unblind}(\langle k_{(\hat{\imath},\hat{\jmath},\hat{v})} \rangle_{blind}; b, pkB_{\hat{\jmath}}) \\ m_{(\hat{\imath},\hat{\jmath},\hat{v})} &= \mathsf{Dec}_s(C_{(\hat{\imath},\hat{\jmath},\hat{v})}; H(k_{(\hat{\imath},\hat{\jmath},\hat{v})})) \end{aligned}$$

Settlement. The proxy $\mathcal{P}$ can share the money collected during subscription fairly by counting the number of users that subscribed to service $\mathcal{L}_j$ in a given subscription period. However, this has to be done without revealing the user's service usage. First, $\mathcal{L}_j$ transfers S_j to the proxy and the privacy trustee. $\mathcal{P}$ and $\mathcal{T}$ check the signature and the time stamp of each $S_{(\mathcal{U},j)}$ to make sure that $\mathcal{L}_j$ does not add self generated subscriptions. Moreover, the proxy checks if $|S_j|$ corresponds to his subscription counter. This is needed to guarantee that services do not try to shrink the anonymity set of users. As the trustee is not online all the time he can only check the plausibility of the value. However, privacy savvy users can submit their encrypted subscriptions (or its hash) to the privacy trustee, which checks if their descriptions are considered during settlement.

The computation of $\mathcal{L}_j$'s fraction of the money is jointly done by the proxy, the privacy trustee and the service $\mathcal{L}_j$. First, they compute $\bigoplus_{\mathcal{U}} P_{(\mathcal{U},j)}$. The result is an encryption of the number of users having subscribed to service $\mathcal{L}_j$. Since $\bigoplus_{\mathcal{U}} P_{(\mathcal{U},j)}$ is encrypted with pkS_j, all three parties have to participate in a distributed $(3,3)$-threshold decryption. The output of $\mathsf{Settlement}(\mathcal{L}_j(S_j; sk_{\mathcal{L}_j}, pk_{\mathcal{P}}), \mathcal{P}(sk_{\mathcal{P}}, pk_{\mathcal{L}_j}), \mathcal{T}(sk_{\mathcal{S}_j,T}))$ is the total number of users subscribed to service $\mathcal{L}_j$. As long as not all parties collude, the service usage privacy of the users is guaranteed. See Section 5.2 for details.

5 Security and Efficiency

5.1 Efficiency analysis

For our efficiency analysis we focus on two main factors. The first is the limitation in computation and communication resources on small mobile devices. The other factor is scalability for the organizations with respect to location resolution, i.e. number n of map cells, number of services ℓ, and number of users. The costs for the setup of the payment and the service and proxy key generation are independent of the number of users and map cells. They are executed by unrestricted parties, and are thus ignored in the analysis. A similar argument holds for the settlement. We consider only public-key operations and use the length parameters from Section 4.

User. The costs incurred by the user are key generation, subscription, and decryption. *Computation:* Key generation involves the generation of a single RSA key. The decryption requires a single Damgård-Jurik decryption. The most relevant cost for the user is the generation of the subscriptions with 12 exponentiations. However, this cost is incurred only once per subscription period. In principle, the computed values can even be reused for multiple periods. *Communication:* A Damgård-Jurik ciphertext size is about $l_H + l_N$, where l_N is the size of the RSA modulus. The overhead in addition to the message length l_D is $l_B + l_b + l_N$. If we use RSA blind signatures, this is three times the size of an RSA modulus. The size of a subscription is about $12\ell(l_H + l_N)$. For small devices and slow communication channels, we suggest to do the key generation and subscription over the user's PC, and synchronize $sk_{\mathcal{U}}$ to the user's mobile device, or create the keys on the device, and move only $pk_{\mathcal{U}}$ to the PC for added security.

Organizations. The scalability of our service is nearly optimal. Key setup and database encryption are independent of the number of users. Database encryption is linear in the number of locations. Subscriptions are linear in the number of services—optimal as all services need to be involved to guarantee service privacy. Practically data transfer is independent of the number of locations and again only depends on the number of services.[1]

Computation: The most prominent cost incurred by the service is database encryption requiring one signature operation per location. *Communication:* The most dominant cost for $\mathcal{L}_j$ and $\mathcal{P}$ is the transmission of the encrypted database $DB_{(j,v)}$ which has length $n \cdot l_D$. Moreover, this costs incur whenever any of the m_i should be updated. This is a consequence of our strong database security definition. For a wide range of services it appears reasonable to relax this requirements. Updated encryptions C_i are computed like in $\mathsf{EncryptData}$ but only for a subset $U \subset \{1, \dots, n\}$ of updated locations.

5.2 Security Analysis

Our main goal is to implement LBSs without revealing additional information about the user's location and her service usage profiles. An adversary involved in our protocol should learn nothing except what he already could have learned by being involved in a scenario without LBSs. For the proxy this implies that he is allowed to know the user's location but should not learn anything about the user's service usage profiles. For service providers this implies that both the user's location and the user's service usage profiles have to be concealed. Note that in a scenario that includes payment mechanisms we have to diminish slightly from this strong security notion of no additional knowledge since a service can infer the number of subscribed users from the received amount of money.

[1] De facto the dependence is logarithmic as locations are included in the message; 4 billion different locations can be encoded using 32 bits.

As a further trust assumption we state that the proxy helps to solve fairness conflicts between users and service providers. This trust assumption is supported by the rationality of the proxy: his core competence is setting up the mobile infrastructure such that services and users can communicate in a fair way. Cheating or not cooperating in resolving fairness disputes decreases his reputation, thus decreasing his profit. Only in cases where accusing the cheater would endanger the users service privacy, we make use of the privacy trustee as an additional off-line trusted party. The assumption of a trusted third party to resolve fairness problems is common in the literature of fair exchange protocols. [2, 3] have shown that the problem of fairly exchanging data requires at least an off-line trusted party.

Location privacy. For the location privacy note first that in our protocol services only get in contact with location related information in the data retrieval phase. However, there the OT based on blind signatures protects location related information from being revealed unintentionally. The security of the OT scheme is based on the signature's blindness property. This property guarantees that for any malicious service $\widetilde{\mathcal{L}}_j$ the view of $\widetilde{\mathcal{L}}_j$ for a messages $M_0 = i\|v$ and for a message $M_1 = i'\|v'$ is computationally indistinguishable. As the user's location $\hat{\imath}$ is hidden in $\langle \hat{\imath}\|\hat{v}\rangle_{blind}$, the location privacy can be reduced to the blindness property of the used blind signature scheme.

Service usage privacy. is more challenging. Unfortunately, achieving service usage privacy First, the relationship user/service plays a role in different protocol stages; i.e. it is present during data retrieval and in the payment processes. Second, we consider a stronger, but realistic, adversary model and allow for a corrupted proxy, who possibly collaborates with any service. This implies that the service usage privacy cannot rely on the help of the proxy. We proceed by analyzing the relevant phases.

In the subscription phase, the user's subscription together with the proof of knowledge could reveal the user's service usage. In case of the first the crucial information is protected by the semantic security of the underlying homomorphic encryption scheme. The semantic security guarantees that two different subscriptions are indistinguishable. The zero-knowledge property of π_j ensures that no further information is revealed.

During the settlement, privacy is protected by using a joint decryption technique, i.e. a ciphertext can only be decrypted if the three parties, the $\mathcal{P}$, $\mathcal{L}_j$, and $\mathcal{T}$ work together. Hence, even if $\mathcal{P}$ and $\mathcal{L}_j$ are corrupted, there is no way for the adversary to force the decryption of a single subscription that would reveal the user/service relationship. This is only true as long as it is not possible to present faked subscriptions to the privacy trustee, which later on get accepted. This would reduce the size of the anonymity set $|S_j|$. The faked subscription attack reduces to $n-1$-attack [9], against which a full protection is impossible. However, to make it more difficult for an adversary, it could be mandatory for users to always submit a subscription (or hash of it) to the trustee. This technique leads to a lower bound for the size of the anonymity set $|S_j|$. To some extent the sensitivity to these attacks can be reduced by using authenticated channels such that subscriptions can be assigned to real users.[2]

Although one cannot assume the honesty of the proxy in general, it often seems more realistic to limit the adversaries' power to corrupt only services. This is further supported by our assumption of the proxy's rationality. In the case of the proxy's honesty, we have a stronger protection against the faking attack: first, because the proxy and the trustee verify the correctness of the signature and the time stamp and second, because the proxy checks for the equality of $|S_j|$ and his subscription counter. Both techniques help to protect against attempts in shrinking the anonymity set.

Database secrecy. In contrast to location and service usage privacy the database secrecy protects the interest of the LBS. It guarantees that a user gets no more than the requested information even if she collaborates with the proxy. The database secrecy of our scheme, relies on two aspects: First, the service encrypts his database before he sends it to the proxy. Second, as a result of the data retrieval phase the user only gets to know the requested data. This is due to the so called 'Database security' [21] of the underlying secure OT protocol.

Fairness. Our system is said to be fair if both the interests of the user and of the service are equally protected. With respect to our protocol this means that neither the user gets access to content without paying nor the service is able to cheat and receive a non-authorized payment without providing the information.

The fairness of our protocol significantly relies on the rationality and the consequential semi-honesty of the proxy. Furthermore we require a trade off between service privacy and fairness to protect against active attacks by services.

User fairness. From the user's perspective fairness is accomplished when the user receives the appropriate location information for her payment.

The proxy sends a request to each service. The service responds with a signed value E_j that is either an encryption of 0 (if the user did not subscribe to $\mathcal{L}_j$) or an encryption of the requested data. Should a service fail to provide any value E_j, the proxy forces it to pay a fee corresponding to the price of a service subscription and passes that money on to the affected user.

Now the proxy computes the user's final value E by combining all responses. The user decrypts E and checks whether it corresponds with her requested data. To facilitate this, the messages $m_{(\hat{\imath},\hat{\jmath},\hat{v})}$ contain $\hat{\imath}$,$\hat{\jmath}$, and $\hat{v}$ and is signed by $\mathcal{L}_j$. If a message is incorrect, the user can file a complaint.[3]

In this case the user has three choices: she can either choose full privacy and give up her money; she can file the complaint with the privacy trustee; or she can complain directly with the proxy. Complaints contain a proof of correct

[2] A powerful adversary will always find ways to forge subscriptions, even if it is just by convincing real users with money.

[3] Note that a complaining user acts as a decryption oracle. Together with the homomorphic property this can lead to the decryption of arbitrary messages. Consequently, user should not complain about random looking messages to an untrusted party.

decryption of a signed E. The recipient of the complaint verifies the proof and pays the money back. $\mathcal{T}$ will ask the proxy for the money given to users during conflict resolution in return for a list of bad services. These services will receive reduced payment or be sanctioned otherwise. Should the proxy refuse to pay the money to the trustee this can only be resolved legally.

Our protocol does not protect against denial of service attacks in which malicious services send random ciphertexts instead of $\langle k_{(\hat{\imath},j,\hat{v})}\rangle_{blind} \otimes Q_j$. However, this can be detected if users are willing to give up their privacy towards the proxy. We again rely on an optimistic strategy and the punishment of attackers upon detection. It is an open problem to propose an efficient data retrieval protocol based on zero-knowledge protocols that solves this problem.

Service fairness. means that service providers receive fair payment. In particular, a service provider must receive money for every user he serves.

This is ensured by the checks done in VerifySubscription. The algorithm checks that $Q_{(\mathcal{U},j)}$ and $P_{(\mathcal{U},j)}$ encode the same value $v \in \{0,1\}$. We refer to $P_{(\mathcal{U},j)}$ as the vote. Obviously if $v = 0$, no service is provided. Consider the case of $v = 1$: if (1) all votes are considered and (2) the votes are counted correctly, $P_{(\mathcal{U},j)}$ increases the subscription counter of $\mathcal{L}_j$ by 1. The first is ensured by the time stamp and signature of the proxy. The second property relies on the security of the homomorphic encryption scheme and the security of the distributed decryption against active adversaries.

6 Conclusion

We introduced the first privacy-preserving LBS framework based on cryptographic techniques, namely, on oblivious transfer and homomorphic encryption. The privacy of the user is protected by hiding the user's location from the services and by not revealing information on the user/service relationship. Additionally, we presented a system for subscription management including a fair yet anonymous payment scheme.

We have given strong intuitions on the different security properties of our scheme, however, it remains an open challenge to prove them in a formal context.

References

1. W. Aiello, Y. Ishai, and O. Reingold. Priced oblivious transfer: How to sell digital goods. In Pfitzmann [24], pages 119–135.
2. N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In *CCS '97: Proceedings of the 4th ACM conference on Computer and communications security*, pages 7–17, New York, NY, USA, 1997. ACM Press.
3. N. Asokan, V. Shoup, and M. Waidner. Asynchronous protocols for optimistic fair exchange. *sp*, 00:0086, 1998.
4. E. Bangerter, J. Camenisch, and U. M. Maurer. Efficient proofs of knowledge of discrete logarithms and representations in groups with hidden order. In Hutter and Ullmann [18], pages 154–171.

5. A. R. Beresford and F. Stajano. Location Privacy in Pervasive Computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003.
6. S. Brands. Rapid demonstration of linear relations connected by boolean operators. In *EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 318–333, 1997.
7. J. Camenisch, G. Neven, and A. Shelat. Adaptive oblivious transfer from blind signatures. Unpublished manuscript through personal communication, 2006.
8. J. Camenisch and M. Stadler. Proof systems for general statements about discrete logarithms. Technical Report TR 260, Institute for Theoretical Computer Science, ETH Zürich, Mar. 1997.
9. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, Feb. 1981.
10. D. Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203, 1982.
11. C.-K. Chu and W.-G. Tzeng. Efficient k-out-of-n oblivious transfer schemes with adaptive and non-adaptive queries. In S. Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 172–183, Les Diablerets, Switzerland, Jan. 23–26, 2005. Springer-Verlag, Berlin, Germany.
12. I. Damgård and M. Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *PKC '01: Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography*, pages 119–136. Springer-Verlag, 2001.
13. H. Federrath, A. Jerichow, D. Kesdogan, and A. Pfitzmann. Security in Public Mobile Communication Networks. In *IFIP TC 6 International Workshop on Personal Wireless Communications*, pages 105–116, 1995.
14. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.
15. L. Fritsch. Profiling and location based services. In M. Hildebrandt and S. Gutwirth, editors, *FIDIS D7.5: Profiling the European Citizen, Cross- disciplinary perspectives*, 2007.
16. M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of First International Conference on Mobile Systems, Applications, and Services (MobiSys'03)*, pages 31–42, 2003.
17. GSM Association: Location Based Services. Permanent reference document. Technical report, SE.23, 2003.
18. D. Hutter and M. Ullmann, editors. *Security in Pervasive Computing, Second International Conference, SPC 2005, Boppard, Germany, April 6-8, 2005, Proceedings*, volume 3450 of *Lecture Notes in Computer Science*. Springer, 2005.
19. T. Kölsch, L. Fritsch, M. Kohlweiss, and D. Kesdogan. Privacy for profitable location based services. In Hutter and Ullmann [18], pages 164–178.
20. M. Naor and B. Pinkas. Oblivious transfer with adaptive queries. In M. J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 573–590, Santa Barbara, CA, USA, Aug. 15–19, 1999. Springer-Verlag, Berlin, Germany.
21. W. Ogata and K. Kurosawa. Oblivious keyword search. *J. Complexity* , 20(2-3):356–371, 2004.
22. R. Ostrovsky and W. Skeith, III. Private searching on streaming data. In *CRYPTO 2005*, pages 223–240, 2005.

23. P. Paillier. Public-key cryptosystem based on composite degree residuosity classes. In J. Stern, editor, *Advances on Cryptology — EUROCRYPT 1999*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–228, Prague, Czech Republic, 2–6 May 1999. Springer-Verlag.
24. B. Pfitzmann, editor. *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, volume 2045 of *Lecture Notes in Computer Science*. Springer, 2001.
25. M. O. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.
26. C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.

A Formal Definitions

Notation. We write cryptographic primitives as $\mathsf{Alg}(x;k)$, where x denotes the processed inputs of the algorithm and k denotes keys, randomness, or public parameters. When it is clear from the context k is omitted. We use the $\mathsf{Alg}(\mathcal{E}_1(x_1;k_1),\mathcal{E}_2(x_2;k_2))$ to denote an interactive algorithm between entities $\mathcal{E}_1$ and $\mathcal{E}_2$ with the respective inputs and keys.

Algorithms. $\mathcal{U}$, $\mathcal{P}$, the $\mathcal{L}_j$, and $\mathcal{T}$ interact using the following algorithms:

– $\mathsf{KeygenL}(1^k)$, $\mathsf{KeygenP}(1^k)$, and $\mathsf{KeygenU}(1^k)$ generate the public/private-key pairs $(pk_{\mathcal{L}_j}, sk_{\mathcal{L}_j})$, $(pk_{\mathcal{P}}, sk_{\mathcal{P}})$, and $(pk_{\mathcal{U}}, sk_{\mathcal{U}})$ of the LBS service, the proxy, and the user respectively.

– $\mathsf{PaymentSetup}(\mathcal{L}_j(1^k),\mathcal{P}(1^k),\mathcal{T}(1^k))$ computes the public payment key pkS_j, and the secret settlement keys $skS_{(\mathcal{L}_j,j)}$, $skS_{(\mathcal{P},j)}$, and $skS_{(\mathcal{T},j)}$ that are private output of $\mathcal{L}_j$, $\mathcal{P}$, and $\mathcal{T}$ respectively. The key pkS_j is used for creating subscriptions; $skS_{(\mathcal{L},j)}$, $skS_{(\mathcal{P},j)}$, and $skS_{(\mathcal{T},j)}$ are only used during settlement.

To complete the key setup, pkS_j and $skS_{(\mathcal{E},j)}$ are added to $pk_{\mathcal{E}}$ and $sk_{\mathcal{E}}$, $\mathcal{E} \in \{\mathcal{P},\mathcal{L}_j\}$, respectively. Consequently $pk_{\mathcal{P}}$ contains the public payment keys of all services. Trustee $\mathcal{T}$ stores all $skS_{(\mathcal{T},j)}$ together as $sk_{\mathcal{T}}$. Key generation is required only once during the setup of the system.

– $\mathsf{Subscribe}(\mathcal{P}(sk_{\mathcal{P}}, pk_{\mathcal{U}}),\mathcal{U}(\hat{\jmath}; sk_{\mathcal{U}}, pk_{\mathcal{P}}))$ allows the user to subscribe to service $\mathcal{L}_{\hat{\jmath}}$. The output of the protocol on the proxy side is real money collected by the proxy and oblivious subscription information $S_{(\mathcal{U},j)}$ for every service, $1 \le j \le l$. $\mathcal{P}$ passes $S_{(\mathcal{U},j)}$ on to $\mathcal{L}_j$ respectively. For $j \neq \hat{\jmath}$, $S_{(\mathcal{U},j)}$ hides the fact that $\mathcal{U}$ is not interested in the service. However, $S_{(\mathcal{U},\hat{\jmath})}$ will allow $\mathcal{L}_{\hat{\jmath}}$ to provide the service (without even knowing that it is doing so). The $S_{(\mathcal{U},j)}$ values also contain the necessary information for the settlement.

– $\mathsf{VerifySubscription}(S_{(\mathcal{U},j)}, j; pk_{\mathcal{U}}, pk_{\mathcal{P}})$ checks that $S_{(\mathcal{U},j)}$ is a valid subscription for service j, and that it can be used for settlement. If $\mathsf{VerifySubscription}$ fails, the service is denied. In this case the proxy has to give back the money to the user. Otherwise, the service adds $S_{(\mathcal{U},j)}$ to its list of subscriptions S_j.

– $\mathsf{Settlement}(\mathcal{L}_j(S_j; sk_{\mathcal{L}_j}, pk_{\mathcal{P}}),\mathcal{P}(sk_{\mathcal{P}}, pk_{\mathcal{L}_j}),\mathcal{T}(sk_{\mathcal{T}}, pk_{\mathcal{P}}))$ outputs the number of users who subscribed to service j. collected during subscription. As long as not all three parties cooperate, none of them learns which users accessed the service.

To keep our protocol simple we restrict users to subscribe only once per subscription period. After all subscriptions have been made, the money collected by the proxy is distributed based on the number of times the different services were subscribed to. The proxy and the trustee need to check that the list of subscriptions is complete. A reduced list S_j would compromise the privacy of the user as a user could 'hide' between a smaller number of other users.

– $\mathsf{EncryptData}(m_{(1,j,v)},\dots,m_{(n,j,v)}, v; sk_{\mathcal{L}_j})$ encrypt the location specific service information of $\mathcal{L}_j$ with version v. To every location i, $1 \le i \le n$, corresponds one encrypted message. The output of the algorithm is the encrypted database $DB_{(j,v)}$.

The above operation is only required when a service is newly created or when the location specific data used for providing a service changes. The database $DB_{(j,v)}$ is transferred to the proxy over a high volume connection.

– $\mathsf{Request}(\mathcal{P}(DB_{(j,\hat{v})}, \hat{\imath}; sk_{\mathcal{P}}, pk_{\mathcal{L}_j}), \mathcal{L}_j(S_{(\mathcal{U},j)}; sk_{\mathcal{L}_j}, pk_{\mathcal{U}}, pk_{\mathcal{P}}))$ The proxy runs Request with service $\mathcal{L}_j$ giving $DB_{(j,\hat{v})}$ of verson $\hat{v}$ and the users current location $\hat{\imath}$ as input. The input at the service side is the user's subscription $S_{(\mathcal{U},j)}$. The proxy obtains an encrypted result E_j from the service. For services (and corresponding $S_{(\mathcal{U},j)}$) the user is not subscribed to E_j is an encryption of 0.
– $\mathsf{Combine}(E_1, \ldots, E_l; sk_{\mathcal{P}}, pk_{\mathcal{U}})$ After running Request with all services, algorithm Combine merges the individual results into a single result E containing an encryption of $m_{(\hat{\imath},\hat{\jmath},\hat{v})}$.
– $\mathsf{Decrypt}(E; sk_{\mathcal{U}}, pk_{\mathcal{L}_{\hat{\jmath}}})$ decrypts E to obtain the result $m_{(\hat{\imath},\hat{\jmath},\hat{v})}$.

The last three operations are triggered by the proxy and result in user $\mathcal{U}$ obtaining the service and location specific message $m_{(\hat{\imath},\hat{\jmath},\hat{v})}$.

Required security properties. In our security considerations we assume that parties communicate over secure channels and that $\mathcal{P}$, $\mathcal{L}_j$, and $\mathcal{T}$, are able to authenticate communication, and to sign messages using their identity. This can for instance be realized using a public key infrastructure. Note that we do not require client authentication and are thus able to avoid client-side certificates. A secure multi-service proxy LBS scheme needs to have the following properties:

Location privacy. As long as the proxy is honest, the location of the user is not revealed. adversary possibly corrupting any number of LBSs and other users and interacting with the honest proxy using protocols EncryptData, Request, ... , can choose two tuples (i, j, v) and (i', j, v'). Based on a random bit b, the attacker then runs the Request protocol with the proxy with one of these tuples as input, without knowing which. The advantage of the attacker in guessing this bit is negligible.

Service usage privacy. Even if the proxy and all the services collude, they cannot determine to which service a user subscribed. Subscribe protocol. The adversary cannot distinguish whether he is interacting with the real protocol or the simulated one. This includes message privacy; i.e., only the users can decrypt the messages of services.

Database secrecy. The user should only get the information she (together with the proxy) requested. ideal implementation simulation. This property must hold, even if users and proxy collude.

Fair payment. The LBS only provides the service if it gets payed for it. If a user pays for a service she can successfully blame others for the bad service.

Authenticity of result. The user is assured that her result is for the right location and of the right version, really originates from the LBS, and is the same as the one committed to in EncryptData.