Privacy-preserving location-based service protocols with flexible access

Shuyang Tang and Shengli Liu*

Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
Email: htftsy@sjtu.edu.cn
Email: liu-sl@cs.sjtu.edu.cn
*Corresponding author

Xinyi Huang

School of Mathematics and Computer Science, Fujian Normal University, Fujian, China
Email: xyhuang@fjnu.edu.cn

Zhiqiang Liu

Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
Email: liu-zq@cs.sjtu.edu.cn

Abstract: We propose an efficient privacy-preserving, content-protecting location-based service (LBS) scheme. Our proposal gives refined data classification and uses generalised ElGamal to support flexible access to different data classes. We also make use of pseudo-random function (PRF) to protect users’ position query. Since PRF is light-weighted primitive, our proposal enables the cloud server to locate position efficiently while preserving the privacy of the queried position.

Keywords: location-based services; LBS; outsourced cloud; security; privacy preserving; public-key cryptosystems; pseudo-random function; PRF; ElGamal cryptosystem.


Biographical notes: Shuyang Tang is an undergraduate student of the Department of Computer Science and Engineering, Shanghai Jiao Tong University. His research interests include cryptocurrency, public-key cryptography, post-quantum cryptography and block chain.

Shengli Liu is a Professor at Shanghai Jiaotong University. She received her Bachelor’s, Master’s and PhD degrees from Xidian University in 1995, 1998 and 2000 respectively. From 2000 till 2002, she continued her research on cryptography and received another PhD from Technische Universiteit Eindhoven, the Netherlands. Her research interests include public-key cryptosystems and applications.

Xinyi Huang received his PhD from the School of Computer Science and Software Engineering, University of Wollongong, Wollongong, NSW, Australia, in 2009. Currently, he is a Professor with the Fujian Provincial Key Laboratory of Network Security and Cryptology, School of Mathematics and Computer Science, Fujian Normal University, Fuzhou, China. His research interests include cryptography and information security.

Zhiqiang Liu received the BS and MS degrees in Mathematics, and PhD in Cryptography from Shanghai Jiao Tong University in 1998, 2001, and 2012, respectively. Currently, he is an Associate Professor in the Department of Computer Science and Engineering, Shanghai Jiao Tong University. His research interests include symmetric-key cryptography, post-quantum cryptography, cryptocurrency, block chain and DAG-based distributed system.
1 Introduction

With the development of wireless technology and ubiquitous computing, the last decade has seen the flourish of intelligent portable devices, like smart phones, electronic pads, etc. Numerous internet service products of a great variety come to be available, such as information searching, on-line games, ticket ordering, electronic maps, and so on. As one important class, location-based services (LBS) has been exclusively researched and implemented with the aim of providing information, utility, entertainment services based on geographical locations for mobile device users.

With a smart phone at hand, it is easy for a person to search the places like ‘the nearest cinema’, or ‘the pubs near the city centre’ on the internet. Current maturity of cloud technology makes it possible for service provider to outsource their data and computation to cloud so as to lighten their managements. However, security issues emerge with an outsourcing database, causing a considerable threat to the privacy and integrity of users’ information and the outsourced database.

In this paper, we focus on the security issues of LBS. An LBS system consists of three parties: the LBS provider, the cloud server (CS) and users. The LBS provider sends outsourcing database to the CS, and grants access keys to users. Authorised users interact with the CS so as to attain LBS services.

The security requirements for a privacy-preserving LBS system are as follows.

- **Privacy of dataset**: the CS cannot learn dataset of the LBS provider. This requires that the LBS provider encodes dataset before outsourcing it to CS.

- **Authorisation of users**: only users who register in the LBS system can enjoy the LBS services.

- **Anonymity of users**: when a user issues a query \((x, y)\) for services, the identity of the user should be unknown to either the LBS provider or CS.

- **Privacy of position**: the position queries \((x, y)\) by users should be unknown to either the LBS provider or CS.

- **Privacy of location-based data**: only the user who issues the query \((x_i, y_i)\) can decode the reply of CS and obtain the corresponding data cell \(d_{cell}(x_i, y_i)\) stored in dataset.

1.1 Related works

Many research efforts have been dedicated to the design of efficient privacy-preserving LBS schemes (see Wikipedia, 2016b; Amar and Boumerdassi, 2007; Magkos, 2011; Jagdale and Bakal, 2013; Barboza-García et al., 2014). The first solution was proposed in Beresford and Stajano (2003), in which the anonymity of users is realised by changing users’ pseudo-identities at a constant frequency. Multiple techniques have leveraged to devise privacy-preserving, content-preserving and efficient location-based service schemes in various scenarios (Khoshgozaran and Shahabi, 2010; Guo et al., 2016; Chen et al., 2016; Li et al., 2016; Zhu et al., 2016). The technique of \(k\)-anonymity (Wikipedia, 2016a) was used to obtain user location privacy, which ensures that the query record of any user could not be distinguished from the other \((k – 1)\) records. Note that a trusted third party (TTP) is necessary to implement \(k\)-anonymity (Khoshgozaran and Shahabi, 2007; Chow et al., 2009). However, assuming the existence of TTP is not reasonable since no one is immune to attack especially in an open setting like internet of things (IoT). Later, dummy locations (Kido et al., 2005), secret circular shift (Lien et al., 2013) were introduced to avoid the reliance on TTP. Unfortunately, most of the previous constructions impose heavy communications or computations on users, leading to much burden on mobile devices. Hence a TTP-free, efficient location-preserving LBS scheme is desirable. Recently, a LBS scheme named FINE was proposed in Shao et al. (2014), which was indeed free of TTP and built from attribution-based encryption (ABE). Thanks to ABE, this scheme supports fine-grained access control. However, the technique of ABE imposes a large amount of keys on CS, and CS has to deal with complicated partial decryption, resulting in a great computational complexity.

1.2 Our contributions

We focus on LBS schemes which are not only privacy preserving, but also support flexible access to different classes of data. More specifically, we build a privacy-preserving LBS scheme, making use of ElGamal public key encryption and pseudo-random function (PRF).

1.3 Paper organisation

This paper is organised as follows. We first give building blocks of our scheme and present the framework in Section 2. In Section 3, we give a basic scheme and analyse its security. In Section 4, we build a more efficient scheme based on the basic one. Section 5 concludes the paper.

2 Preliminaries

In this paper, let \([n]\) denote the set \([1, 2, \cdots, n]\). We use bold lowercase symbol (e.g., \(\mathbf{v}\)) to represent a vector, and bold uppercase symbol (e.g., \(\mathbf{M}\)) to represent matrix. Denote by \(\mathbf{v}^T\) the transpose of vector \(\mathbf{v}\). Let \(s_1, \cdots, s_t \overset{\$}{\leftarrow} S\) denote picking elements \(s_1, \cdots, s_t\) uniformly from set \(S\). Let \(\mathbf{A} = \mathbf{B}^T\) denote the value of \(\mathbf{B}\) is assigned to \(\mathbf{A}\) or \(\mathbf{A}\) is defined as \(\mathbf{B}\).

The multiplication and division between two vectors of same size are defined as follows. For any \(\mathbf{u} = (u_1, u_2, \cdots, u_t)^T\), and \(\mathbf{v} = (v_1, v_2, \cdots, v_t)^T\):
The functionality of a naive system (without considering security issues) is as follows:

- The LBS provider possesses a collection of data, called dataset, which can be considered as an array of data cells. Each data cell datacell\((x,y)\) is uniquely indexed by a coordinate \( (x,y) \). In other words, dataset = \{datacell\((x,y)\)\}. The LBS provider will outsource the dataset to CS.
- Users aims to query a position \((x,y)\), and access the corresponding data datacell\((x,y)\).
- CS saves dataset to cloud database, and will delegate the LBS provider to provide services to users. Given a query \((x,y)\), CS will search its database and responds with datacell\((x,y)\) to the user.

2.2 Generalised ElGamal encryption

The ElGamal Encryption (Gamal, 1985) is one of most commonly used public encryption schemes, proposed by Elgamal in 1985, it consists of four algorithms.

1. ElGamal.In\((\kappa)\) → \(pp = (\mathbb{G}, p, g)\): initialisation algorithm takes the security parameter \(\kappa\) as input and returns public parameter \((\mathbb{G}, p, g)\), where \(\mathbb{G}\) is a group of prime order \(p\), where \(p\) is of bit length \(\kappa\), and \(g\) is a generator of group \(\mathbb{G}\).

2. ElGamal.Kg\((pp)\) → \((pk, sk)\): key generation algorithm takes the public parameter \(pp\) as the unique input and outputs a pair of secret key \(sk\) and public key \(pk\).
   - let \(k \leftarrow \mathbb{Z}_p\) and compute \(h = g^k\)
   - set \(sk = k\), and \(pk = h\)
   - return \(((pk, sk) = (h, k))\).

3. ElGamal.Enc\((pk, m)\) → \(c\): the encryption algorithm takes the public key \(pk\) and a message \(m \in \mathbb{G}\) as input, and returns a ciphertext \(c\).
   - choose \(r \leftarrow \mathbb{Z}_p\)
   - compute \(c = (g^r, h^r \cdot m)\)
   - return \(c\).

4. ElGamal.Dec\((sk, c)\) → \(m\): the decryption algorithm takes the secret key \(sk\) \((sk = k)\) and ciphertext \(c = (\delta_0, \gamma_0)\) as input, and outputs message \(m\).
   - parse \(c = (\delta_0, \gamma_0)\)
   - compute \(m = \frac{\gamma_0}{\delta_0^k}\)
   - return \(m\).

The correctness of ElGamal scheme can be guaranteed by

\[
\frac{\gamma_0}{\delta_0^k} = \frac{h^r \cdot m}{(g^r)^k} = \frac{h^r \cdot m}{h^r} = m.
\]

Remark: We assume that a \(\kappa\)-bit string can be efficiently encoded to (and efficiently decoded from) an element in \(\mathbb{G}\). If the message is much longer than \(\kappa\), i.e., \(|m| > \kappa\), we will divide the message into several blocks and encrypt each block with the ElGamal encryption algorithms.

Generalised ElGamal encryption (GE) allows simultaneous encryption on multiple messages with different key pairs. It consists of four algorithms.
1. GE.In(1^κ) → pp: the initialisation algorithm takes security parameter κ as input and outputs a multiplicative group \( G \) of prime order \( p \), and a generator \( g \) of this group. Similarly to the Elgamal scheme, it returns \( pp = (G, p, g) \) as the public parameter.

2. GE.Kg(pp) → (pk, sk): the key generation algorithm takes public parameter as input and generates \( n \) pair of keys \((pk_i, sk_i) = (h_i, k_i) = (g^{k_i}, k_i)\), where \( k_i \leftarrow \mathbb{Z}_p \). Set \( pk = (pk_1, pk_2, \ldots, pk_n)^T = (h_1, h_2, \ldots, h_n)^T \) and \( sk = (sk_1, sk_2, \ldots, sk_n)^T = (k_1, k_2, \ldots, k_n)^T \). Return \((pk, sk)\).

3. GE.Enc(pk, M) → C: to encrypt a message matrix \( M \in (\mathbb{G})^{l \times n} \) of size \( l \times n \) under the public key vector \( pk = (pk_1, pk_2, \ldots, pk_n)^T \), it encrypts the \( n \) items in each row of \( M \) with the ElGamal encryption under \( n \) public keys \((pk_1, pk_2, \ldots, pk_n)\) but using a shared randomness. More precisely, given \( M_{l \times n} = \begin{pmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,n} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ m_{l,1} & m_{l,2} & \cdots & m_{l,n} \end{pmatrix} \) 

\[
\begin{align*}
\text{pk} &= (pk_1, pk_2, \ldots, pk_n)^T = (h_1, h_2, \ldots, h_n)^T, \\
\text{choose } r &= (r_1, r_2, \cdots, r_l)^T \leftarrow Z_p^l, \\
\text{and compute the ciphertext matrix as}
\end{align*}
\]

\[
C_{(x(n+1))} = \begin{pmatrix} g^{r_1} h_{r_1}^{t_1} \cdot m_{1,1} & h_{r_1}^{t_2} \cdot m_{1,2} & \cdots & h_{r_1}^{t_n} \cdot m_{1,n} \\ g^{r_2} h_{r_2}^{t_1} \cdot m_{2,1} & h_{r_2}^{t_2} \cdot m_{2,2} & \cdots & h_{r_2}^{t_n} \cdot m_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ g^{r_l} h_{r_l}^{t_1} \cdot m_{l,1} & h_{r_l}^{t_2} \cdot m_{l,2} & \cdots & h_{r_l}^{t_n} \cdot m_{l,n} \end{pmatrix}
\]

\[
= \begin{pmatrix} h_{r_1}^{t_1} \otimes m_{1,1} & h_{r_1}^{t_2} \otimes m_{1,2} & \cdots & h_{r_1}^{t_n} \otimes m_{1,n} \\ h_{r_2}^{t_1} \otimes m_{2,1} & h_{r_2}^{t_2} \otimes m_{2,2} & \cdots & h_{r_2}^{t_n} \otimes m_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ h_{r_l}^{t_1} \otimes m_{l,1} & h_{r_l}^{t_2} \otimes m_{l,2} & \cdots & h_{r_l}^{t_n} \otimes m_{l,n} \end{pmatrix}
\]

4. GE.Dec(sk, C) → M: parse \( sk = (sk_1, sk_2, \cdots, sk_n)^T = (k_1, k_2, \cdots, k_n)^T \) and \( C \) as a \( l \times (n+1) \) matrix over \( G \), i.e., 

\[
C_{(x(n+1))} = \begin{pmatrix} c_0 & c_1 & c_2 & \cdots & c_n \end{pmatrix}
\]

\[
= \begin{pmatrix} g^{t_1} h_{t_1}^{k_1} \cdot m_{1,1} & h_{t_1}^{k_2} \cdot m_{1,2} & \cdots & h_{t_1}^{k_n} \cdot m_{1,n} \\ g^{t_2} h_{t_2}^{k_1} \cdot m_{2,1} & h_{t_2}^{k_2} \cdot m_{2,2} & \cdots & h_{t_2}^{k_n} \cdot m_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ g^{t_l} h_{t_l}^{k_1} \cdot m_{l,1} & h_{t_l}^{k_2} \cdot m_{l,2} & \cdots & h_{t_l}^{k_n} \cdot m_{l,n} \end{pmatrix}
\]

\[
= \begin{pmatrix} c_1 & c_2 & \cdots & c_n \end{pmatrix}
\]

Return \( M = \begin{pmatrix} c_1 & c_2 & \cdots & c_n \end{pmatrix} \)

2.3 Symmetric encryption

A symmetric encryption scheme SE (AES is recommended) consists of three algorithms:

1. SE.Kg(1^κ) → ek: the key generation algorithm takes a security parameter as input and returns a key ek. In this paper, the symmetric key is regarded as an element uniformly randomly chosen from \( \mathbb{G} \), and can also be explained as a random bit string.

2. SE.Enc(ek, m) → c: the encryption algorithm takes the symmetric key \( ek \) and a message (i.e., plaintext) \( m \) as input and returns a ciphertext \( c \).

3. SE.Dec(ek, c) → m: the decryption algorithm takes the key \( ek \) and a ciphertext \( c \) as input and returns a plaintext \( m \).

2.4 Pseudorandom function family

Pseudorandom Function \([\text{PRF} \text{ (Katz and Lindell, 2007)}] \) family \( \{\text{PRF}_{k'}(\cdot)\}_{k' \in \{0,1\}^\kappa} \) is a family of Functions indexed by \( k' \in \{0,1\}^\kappa \) and the function domain and codomain are both \( \{0,1\}^n \times \{0,1\}^n \). It satisfies the following properties:

1. For any \( k' \in \{0,1\}^\kappa \), \( \text{PRF}_{k'}(\cdot) \) should be efficiently computable.

2. For any probabilistic polynomial-time (PPT) distinguisher \( D \) having access to either a PRF \( \text{PRF}_{k'}(\cdot) \) or a random function \( f_k(\cdot) \), there exists a negligible function \( \text{negl}(\cdot) \) such that:

\[
| \text{Pr}[D^{\text{PRF}_{k'}(\cdot)}(1^\kappa) = 1] - \text{Pr}[D^{f_k(\cdot)}(1^\kappa) = 1] | \leq \text{negl}(\kappa)
\]

where \( k' \in \{0,1\}^\kappa \) is chosen uniformly at random and \( f_k \) is chosen uniformly at random from the set of all functions mapping \( 2\kappa \)-bit strings to \( 2\kappa \)-bit strings.

In practice, block cipher encryption (e.g., DES or AES) can be regarded as Pseudorandom Permutation, which is naturally PRF (Katz and Lindell, 2007). As a result, the evaluation of PRF is much more efficient than a public-key encryption algorithm.

3 The basic construction and security analysis

In this section, we firstly introduce the framework of LBS scheme in Subsection 3.1. After that, we present the basic construction in Subsection 3.2 and give an analysis in Subsection 3.3.
3.1 The framework

The dataset is divided into \( w \times s \) cells. Each data cell \( \text{DataCell}\{i, j\} \) is indexed by its geographical position \((i, j)\) where \( i \in \{1, 2, \ldots, w\}, j \in \{1, 2, \ldots, s\} \). It consists of different classes of data, for instance hospital-only, restaurant-only, or all-except-hospital. Let \( R \) be the number of different classes. Without loss of generality, we assume that each class consists of \( l \) group elements (otherwise padded with \( \perp \)), i.e., \( \text{DataCell}\{i, j\} = (m_{1,u}, m_{2,u}, \ldots, m_{l,u})^R \), where \( u \in \{1, 2, \ldots, R\} \). Therefore,

\[
\text{DataCell}\{i, j\} = \{(i, j), M^{(i,j)}\},
\]

and

\[
M^{(i,j)} = [m_1, m_2, \ldots, m_R] = \begin{bmatrix}
m_{1,1} & m_{1,2} & \cdots & m_{1,R} \\
m_{2,1} & m_{2,2} & \cdots & m_{2,R} \\
\cdots & \cdots & \cdots & \cdots \\
m_{l,1} & m_{l,2} & \cdots & m_{l,R}
\end{bmatrix}.
\]

See Table 1.

<table>
<thead>
<tr>
<th>Class 1</th>
<th>Class 2</th>
<th>\ldots</th>
<th>Class R</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{DataCell}{i, j}</td>
<td>{(i, j), M^{(i,j)}}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We present the framework of our LBS scheme. It consists of the following four parts.

- **System initialisation (SI)**: set up the system and publish the public parameters of the system.

- **Service data creation (SDC)**: the LBS provider encodes its dataset and sends the encoded version \( \text{Encode} (\text{dataset}) = (\text{Encode} (\text{DataCell}\{(i, j)\}))_{(i, j) \in [w] \times [s]} \) to CS.

- **User grant (UG)**: a user registers to the LBS provider and the LBS provider issues a token to the user.

- **User revocation (UR)**: the LBS provider revokes a user so that the user cannot use the LBS service.

- **LBS service (LBS)**: a user makes use of his token to encode his location-based query \((i, j)\) to CS, and CS locates the corresponding encoded data \( \text{Encode} (\text{DataCell}\{(i, j)\}) \) in its database. Then the user uses his token to decode \( \text{Encode} (\text{DataCell}\{(i, j)\}) \) and recover the data \( \text{DataCell}\{i, j\} \) that he demands.

As all previous LBS schemes, we assume that the CS is **honest but curious**, i.e., it strictly follows the protocol, but it is curious about the LBS provider’s dataset and users’ queries.

3.2 Construction 1: The basic one

3.2.1 System initialisation

To setup the system, both the LBS Provider and CS will generate their keys and publish the public parameters.

- **LBS Provider**: LBS calls the PRF key generation algorithm to obtain a PRF key and calls the generalised ElGamal key generation algorithm to generate \( R \) key pairs:

\[
\begin{align*}
&1. \text{ randomly choose a PRF key } k' \leftarrow \{0, 1\}^s, \\
&2. \text{ run GE.In}(1^s) \rightarrow (G, g, p); \\
&3. \text{ GE.Kg}(1^s, (G, g, p)) \rightarrow (pk, sk), \text{ where } pk = (h_1, h_2, \ldots, h_R), \text{ and } sk = (k_1, k_2, \ldots, k_R). \\
&4. \text{ publish } \{(G, g, p), pk\}.
\end{align*}
\]

- **Cloud server**: generate its public/secrets key pair with ElGamal. \( \text{Kg}(1^s, (G, g, p)) \rightarrow (pk_c, sk_c) \). It sets userlist = \( \emptyset \). Then publish the public key of userlist.

The public parameter of the system is parameter = \( \{(G, g, p), pk, pk_c\} \).

**Figure 2** System initialisation SI

![SI System Initialisation](image)

3.2.2 Service data creation

During Service Data Creation, the LBS provider encodes its own dataset dataset under the PRF under the PRF key \( k \). In formula, \( \text{index}_{i,j} = \text{PRF}_{k'}(i, j) \).

- **Coordinate encoding**: the coordinate \((i, j)\) is encoded with the PRF under the PRF key \( k \). In formula, \( C^{(i,j)} = \text{GE.Enc}(pk, M^{(i,j)}) \).

- **Data encoding**: the data item \( M^{(i,j)} \) in data cell \( \text{DataCell}\{(i, j)\} = (i, j, M^{(i,j)}) \) is encrypted with generalised ElGamal scheme under the public key \( pk \), i.e., \( C^{(i,j)} = \text{GE.Enc}(pk, M^{(i,j)}) = \text{GE.Enc}(pk, (m_1, m_2, \ldots, m_R)) \).
Table 2 Notations for basic construction

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\kappa$</td>
<td>Security parameter</td>
</tr>
<tr>
<td>$p$</td>
<td>A large prime of bit-length $\kappa$</td>
</tr>
<tr>
<td>$\mathbb{G}$</td>
<td>A multiplicative group of order $p$</td>
</tr>
<tr>
<td>$g$</td>
<td>A generator of group $\mathbb{G}$</td>
</tr>
<tr>
<td>$M^{(i,j)}$</td>
<td>Dataset for cell $(i,j)$, in which $M^{(i,j)} = [m_1, m_2, \ldots, m_R]$</td>
</tr>
<tr>
<td>$C^{(i,j)}$</td>
<td>Encryption of $M^{(i,j)}$ through GE, $C^{(i,j)} = [c_0, c_1, c_2, \ldots, c_R]$</td>
</tr>
<tr>
<td>${\text{PRF}_h}$</td>
<td>A pseudorandom function family</td>
</tr>
<tr>
<td>$k'$</td>
<td>Key for PRF</td>
</tr>
<tr>
<td>$\text{id}$</td>
<td>Identity of user</td>
</tr>
<tr>
<td>$\text{pid}$</td>
<td>Pseudo-identity of user, granted by the LBS Provider</td>
</tr>
<tr>
<td>$\text{pk}$</td>
<td>Vector $\text{pk} = (pk_1, pk_2, \ldots, pk_R)$, i.e., public keys for $R$ classes</td>
</tr>
<tr>
<td>$sk$</td>
<td>Vector $sk = (sk_1, sk_2, \ldots, sk_R)$, i.e., secret keys for $R$ classes</td>
</tr>
<tr>
<td>$(pk_{cs}, sk_{cs})$</td>
<td>Public/private key pair of cloud server</td>
</tr>
<tr>
<td>$(pk_{pid}, sk_{pid})$</td>
<td>Public/private key pair of user with pseudo-identity $\text{pid}$</td>
</tr>
<tr>
<td>$e_k$</td>
<td>Key for symmetric encryption</td>
</tr>
<tr>
<td>$\text{index}^{(i,j)}$</td>
<td>Index of cell $(i,j)$</td>
</tr>
<tr>
<td>$k''$</td>
<td>Session key in Diffi-Hellman’s key exchange scheme</td>
</tr>
<tr>
<td>$\text{datacell}^{(i,j)}$</td>
<td>$\text{datacell}^{(i,j)} = (i,j, M^{(i,j)})$</td>
</tr>
<tr>
<td>$\text{dataset}$</td>
<td>$\text{dataset} = {\text{datacell}^{(i,j)}}_{(i,j)\in[w] \times [s]}$</td>
</tr>
<tr>
<td>$\text{Encode}(\text{datacell}^{(i,j)})$</td>
<td>$\text{Encode}(\text{datacell}^{(i,j)}) = (\text{index}^{(i,j)}, C^{(i,j)})$</td>
</tr>
<tr>
<td>$\text{Encode}(\text{dataset})$</td>
<td>$\text{Encode}(\text{dataset}) = {\text{Encode}(\text{datacell}^{(i,j)})}_{(i,j)\in[w] \times [s]}$</td>
</tr>
</tbody>
</table>

Figure 3 Service data creation SDC

```
LBS Provider
For $i = 1$ to $w$, $j = 1$ to $s$
deal with $\text{datacell}^{(i,j)}$ as follows.
$\text{PRF}_{\alpha}(i,j) \rightarrow \text{index}^{(i,j)}$
$\text{datacell}^{(i,j)} = (i,j, M^{(i,j)})$
$M^{(i,j)} = [m_1, m_2, \ldots, m_R]$
$p = (pk_1, pk_2, \ldots, pk_R)^T = (h_1, h_2, \ldots, h_R)^T$
$r = (r_1, r_2, \ldots, r_l)^T \overset{\$}{\leftarrow} \mathbb{Z}_p$
$C^{(i,j)} = [g^{r_1} h_1^{r_1} \cdot m_1, \ldots, g^{r_l} h_1^{r_l} \cdot m_1, g^{r_1} h_2^{r_1} \cdot m_2, \ldots, g^{r_l} h_2^{r_l} \cdot m_2, \ldots, g^{r_1} h_s^{r_1} \cdot m_s, \ldots, g^{r_l} h_s^{r_l} \cdot m_s]$
Send to Cloud Server $\{\text{index}^{(i,j)}, C^{(i,j)}\}$
(1 ≤ $i \leq w$, 1 ≤ $j \leq s$)

Cloud Server
Receive and store $\{\text{index}^{(i,j)}, C^{(i,j)}\}$
(1 ≤ $i \leq w$, 1 ≤ $j \leq s$)
```

As a result, the encoded data cell consists of the encoded index $\text{index}^{(i,j)}$ and the encrypted data item $C^{(i,j)}$.

The LBS Provider sends all the encoded data cells $\{\text{index}^{(i,j)}, C^{(i,j)}\}$ (1 ≤ $i \leq w$, 1 ≤ $j \leq s$) to CS.

3.2.3 User grant and revocation

With User Grant, the LBS Provider issues a token to a newly registered user, and with User Revocation, it revokes users.

- **User grant**: suppose that a user id registers to the system, applying for a token which is used to access data of $u$ classes $i_1, i_2, \ldots, i_u$, where $i_j \in [R]$ and $j = 1, 2, \ldots, u$. We assume that there is a secret and authenticate channel between the LBS provider and users during this phase. The user picks $\alpha \overset{\$}{\leftarrow} \mathbb{Z}_p$, and calculates $h_1 = g^\alpha$, then it sends $(\text{id}, h_1, (i_1, i_2, \ldots, i_u))$ to the LBS Provider through a secure channel. Then the LBS Provider picks $\beta \overset{\$}{\leftarrow} \mathbb{Z}_p$ and calculates $h_{\text{pid}} = h_1^\beta$. It picks pseudo-random identity $\text{pid} \overset{\$}{\leftarrow} \{0, 1\}^{\kappa}$, and then sends to user $(\text{pid}, \beta, h_{\text{pid}}, (sk_{i_1}, sk_{i_2}, \ldots, sk_{i_u}), k')$. After that, the user calculates its own secret key $sk_{\text{pid}} = \alpha \cdot \beta$, and sets his public key as $pk_{\text{pid}} = h_{\text{pid}}$. The user sets his own token as $\text{token}_{\text{pid}} = (sk_{\text{pid}}, (sk_{i_1}, sk_{i_2}, \ldots, sk_{i_u}), k')$. Meanwhile, the LBS Provider sends to Cloud server $(\text{pid}, h_{\text{pid}}, (i_1, i_2, \ldots, i_u))$ and Cloud server adds $(pk_{\text{pid}}, h_{\text{pid}}, (i_1, i_2, \ldots, i_u))$ into its userlist.

- **User revocation**: to revoke user id, the LBS Provider sends the corresponding $\text{pid}$ to Cloud server, and Cloud server locates it and deletes the corresponding item from userlist.
### 3.2.4 Location-based service

When a user uses location-based service, he expects to obtain the data of classes \([j_1, j_2, \ldots, j_v]\) of data in a datacell with respect to an geographic location \((x, y)\) from CS.

1. **User** computes index of its point-of-interest (POI) \((x, y)\) and gets \(\text{PRF}_k(x, y) \rightarrow \text{index}\). He also prepares the types of data, denoted by \([j_1, j_2, \ldots, j_v]\), that he takes interests in. Then he utilises ElGamal encryption to get \(\text{ElGamal.Enc}(pk_{cs}, \text{index}) \rightarrow c_I\), \(\text{ElGamal.Enc}(pk_{cs}, \text{pid}) \rightarrow c_P\), and \(\text{ElGamal.Enc}(pk_{cs}, (j_1, j_2, \ldots, j_v)) \rightarrow c_T\). Finally, he sends \((c_I, c_P, c_T)\) to Cloud server.

2. **Cloud server** decrypts the received \((c_I, c_P, c_T)\) to obtain \(\text{index}, \text{pid}, (j_1, j_2, \ldots, j_v)\). It looks up the userlist to check whether there exists an item \((\text{pid}, h_{\text{pid}}, t = (i_1, i_2, \ldots, i_u))\) which is consistent to \(\text{pid}, \text{and } (j_1, j_2, \ldots, j_v) \subseteq t\). It refuses to provide services if not. Otherwise, it calculates a Diffie-Hellman session key \(k'' = h_{sk_{cs}}^{pk_{cs}}\). After that, Cloud server looks up table to locate \((\text{index}_{(i,j)}, C^{(i,j)})\) with \(\text{index}_{(i,j)} = \text{index}\), where \(C^{(i,j)} = [c_0, c_1, \ldots, c_R]\), then encrypts \((c_0, c_1, c_j, \ldots, c_{j_v})\) under the session key \(k''\) and sends the ciphertext to the user.

3. **User** receives responses from Cloud server, and decrypts it with session key \(k'' = pk_{cs}^{sk_{cs}}\) to recover \([c_0, c_1, c_j, \ldots, c_{j_v}]\). Then with \(sk_{j_1}, sk_{j_2}, \ldots, sk_{j_v}\) in its token, user can get \(m_{ij} = c_{j_i}/(s_{k_{j_i}})^t\) \((1 \leq i \leq v)\).

More details are shown in Figure 5.
3.3 Security analysis

We present security analysis for our basic scheme.

- **Privacy of dataset**: the dataset of the LBS provider is encrypted with ElGamal public-key encryption (PKE) scheme. Since the security of ElGamal PKE can be reduced to the decisional Diffie-Hellman (DDH) assumption. We claim that the dataset is unknown to the CS unless the CS can break the DDH problem. Any unauthorised user cannot learn any dataset since CS will not provide any services to him.

- **Authorisation of users**: the access control of users in the system is implemented by CS. Recall that CS is honest but curious, CS will always provide services to authorised users.

- **Anonymity of users**: when a user id registers to the system, the LBS provider will issue a pseudo-name pid to user id. Later, user id will always use pid in the system. Therefore, CS knows pid instead of id. The LBS provider knows the map between id and pid. However, the communication between CS and user id is encrypted by ElGamal PKE and symmetric encryption scheme, hence, the LBS provider does not know which user is asking service to CS. As a result, the anonymity of the user is preserved.

- **Privacy of position and location-based data**: the position \((x, y)\) queried by users is encoded with Pseudo-Random Function PRF. Recall that the output of PRF is pseudo-random. Consequently, the index \(\text{index} = \text{PRF}_{k_{\text{CS}}}(x, y)\) looks random to CS, and CS will not learn any information from index. Recall that the LBS provider does not know the secret key \(sk_{\text{pid}}\) of user pid. As a result, the encrypted communication between CS and user id guarantees that LBS and other users cannot learn \((x, y)\) and the Location-Based Data that CS replies to user id.

4 Construction 2: A more efficient one

The basic construction only supports one position \((x, y)\) per query from users. There are two subtleties:

1. If a user wants to ask for several data-cells with different positions, he has to implement LBS protocol several times. The number of rounds and computational complexity of the user will increase linearly to the number of queries.

2. Recall in our basic construction, the user has to carry on the decryption algorithm of ElGamal that is much more complicated than any symmetric encryption like AES. So the user has to do \(\Theta(n)\) times of ElGamal decryption if he asks \(n\) positions, which is quite a computational burden on the user.

Based on the above observations, we propose the following improvements:

- **Support of area-query**: we allow the users to query a circle area of a central position \((x, y)\) and radius \(d\), denoted by
  \[
  \delta(x, y, d) = \{(u, v) \in \mathbb{Z}_p^2 : \| (x, y) - (u, v) \|_2 \leq d \}.
  \]
  To avoid too much computation and bandwidth burden on mobile devices during user’s inquiry into each cell within its interested area, the whole position-based dataset is divided into blocks, each block consisting of several cells. For example, one block may include 100 \times 100 cells. Each time that user sends request \(\delta(x, y, d)\) to Cloud Server to fetch all data cells in the blocks that \(\delta(x, y, d)\) covers.

- **Efficiency improvement**: to avoid huge amount of computation overhead imposed to the LBS provider as well as the users, we use hybrid encryption technique. The LBS provider will use ElGamal PKE to encrypt a symmetric key for each class of data, and then uses the symmetric key to encrypt the class of data in all data cells. Symmetric encryption will make the encryption and decryption much more efficient.

The dataset is divided into multiple blocks \(B_{i,j}\), where \(i \in I\) and \(j \in J\). Each block may contains many data cells. There is a geographical central data cell \((x_i, y_j)\) in each block \(B_{i,j}\). We collect all the data in data cells in the block and denoted by \(M^{(i,j)}\). To facilitate our description, we denote a block by a tuple \(B_{i,j} = ((x_i, y_j), M^{(i,j)})\). Similar to our basic construction, the data in each block is divided into several classes, i.e., \(M^{(i,j)} = [m_1, m_2, \ldots, m_R]\), where the \(r\)-th column contains all location and detailed description information of objects that is open to class \(r\) in block \(B_{i,j}\). Likewise, \(l\) is the upper bound of data length for one class within one block.

Similar to the notations of previous construction, we suppose that the data set contains \(w \times s\) blocks, where \(1 \leq w, s \leq p\). By notation \((x_{i,j}, y_{i,j})\), we denote the coordinates of the data cell in the central position of block \((i, j)\), similarly, \(1 \leq x_{i,j}, y_{i,j} \leq p\).

Now instead of encrypting all LBS dataset with public key encryption scheme, we only use ElGamal Encryption to encrypt \(R\) symmetric keys for data of each block, and massive data are encrypted with the corresponding symmetric key.

4.1 System initialisation

System initialisation directly follows that of basic construction.

4.2 Service data creation

In SDC, we encrypt all data in blocks with symmetric keys, each class with a different symmetric key. Other parts of procedure SDC remain the same as the basic construction.
Figure 6  Dividing into blocks (see online version for colours)

Figure 7  System initialisation SI for new construction

Figure 8  Service data creation SDC for new construction

Each block \((i, j), B_{(i,j)}\) is encoded as follows:

- **Coordinate encoding**: only the coordinate \((x_{i,j}, y_{i,j})\) in the centre of block \((i, j)\) is encoded with the PRF under the PRF key \(k\). In formula, 
  \[\text{index}_{(i,j)} = \text{PRF}_k(x_{i,j}, y_{i,j})\], which is used to index the block.

- **Data encoding**: the data item is encrypted with symmetric encryption scheme under \(R\) keys \((ek_1, ek_2, \cdots, ek_R)\), one for each class, i.e.,
  \[c_r = \text{SE.Enc}(ek_r, m_r)\] for each \(1 \leq r \leq R\), where
  \[B_{(i,j)} = (x_{i,j}, y_{i,j}), M^{(i,j)}\], and
  \[M^{(i,j)} = [m_1, m_2, \cdots, m_R]\]. After that, each \(ek_r\) is encrypted with ElGamal key \(pk_r\), and regarded as the head of each class. Formally,
  \[\text{ElGamal.Enc}(sk_r, ek_r) \rightarrow ek_r\] and
  \[C^{(i,j)} = \begin{bmatrix} ek_1 & ek_2 & \cdots & ek_R \\ c_1 & c_2 & \cdots & c_R \end{bmatrix}\].

As a result, the encoded data blocks consists of encoded index of centre position \(\text{index}_{(i,j)}\) and the encrypted data item \(C^{(i,j)}\). The LBS Provider sends all the encoded block cells \(\{(\text{index}_{(i,j)}, C^{(i,j)})\}\) \((1 \leq i \leq w, 1 \leq j \leq s)\) to CS.

4.3 User grant and revocation

User Grant and User Revocation in this construction follows that of basic construction.

4.4 Location-based service

1  Suppose that a user asks for data of class \(t = (j_1, j_2, \cdots, j_r)\) within the area \(\delta(x, y, d)\). The user finds all blocks intersecting with \(\delta(x, y, d)\), then adds redundant blocks to form a list, then arranges centre coordinates of blocks in list as \(\{(x_{i,j}, y_{i,j})\}_{i \in [t]}\), where \((x_{i,j}, y_{i,j})\) is central coordinates of \(i\)-th block in query list. It calculates \(\text{PRF}_k(x_{i,j}, y_{i,j}) \rightarrow \text{index}_i\) for each \(i \in [t]\). Then, it arranges
target = (index1, index2, ..., indexv) and gets
ElGamal.Enc(pkcs, target) → c1
ElGamal.Enc(pkcs, pid) → cp and
ElGamal.Enc(pkcs, t) → ct. After that, user sends
(pid, c1, cp, ct) to the cloud.

2 Cloud server decrypts (c1, cp, ct) and checks whether
there exists an item of identity pid in userlist that
contains t = (j1, j2, ..., jv), and rejects service if
not. Otherwise, it calculates a Diffi-Hellman session
key k' = h^pid. After that, Cloud server looks up
table to locate all (index(i,j), C(i,j)) with
index(i,j) ∈ target, where

\[ C(i,j) = \begin{bmatrix} e^{k_1} & e^{k_2} & \cdots & e^{k_r} \\
                        e_1 & e_2 & \cdots & e_r \end{bmatrix}, \]

then sends (e^{k_1}, e^{k_2}, ..., e^{k_r}) and
(e_{j1}, e_{j2}, ..., e_{jv}) to user after symmetric encryption
with session key k''.

3 For each received data block, the user receives the response
from Cloud server, and decrypts it with
session key k'' = pk^{k_{skd}}, to recover

(e^{k_1}, e^{k_2}, ..., e^{k_r}) and (e_{j1}, e_{j2}, ..., e_{jv}). Then
with sk_{j1}, sk_{j2}, ..., sk_{jv}, in its token, user gets
(e^{k_1}, e^{k_2}, ..., e^{k_r}) to decrypt (e_{j1}, e_{j2}, ..., e_{jv})
and recovers (m_{j1}, m_{j2}, ..., m_{jv}).

More details are shown in Figure 9.

5 Security and performance analysis

5.1 Security analysis

The constructions of our newly proposed schemes are
based on pseudorandom functions, symmetric encryption,
ElGamal encryption, as well as the generalised ElGamal
encryption. The pseudorandom functions, symmetric
cryptography can be instantiated with AES, whose security
has been thoroughly investigated. The security of the
generalised ElGamal Encryption scheme can be proved based
on the well-known DDH problem.

---

**Figure 9** Location-based service LBS for new construction

<table>
<thead>
<tr>
<th>LBS User</th>
<th>Cloud Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine t blocks covered by b(x, y, d)</td>
<td>Receive (c1, cp, ct);</td>
</tr>
<tr>
<td>Determine t block centers {(x_i, y_i)}_{i\in[t]}</td>
<td>For each i ∈ [t]:</td>
</tr>
<tr>
<td>For each i ∈ [t]:</td>
<td></td>
</tr>
<tr>
<td>PRF' (x_i, y_i) → index_{x_i, y_i}</td>
<td>index_{x_i, y_i} = \left[ \frac{pk^{g^{(i)}}<em>{\text{index}</em>{x_i, y_i}}}{g^{(i)}}, \frac{pk^{g^{(i)}}<em>{\text{index}</em>{x_i, y_i}}}{g^{(i)}}, \frac{pk^{g^{(i)}}<em>{\text{index}</em>{x_i, y_i}}}{g^{(i)}} \right],</td>
</tr>
<tr>
<td>e_j^{(i)} ∈ Z_p^*</td>
<td>target = {index_{x_i, y_i}}_{i\in[t]}</td>
</tr>
<tr>
<td>c_j \leftarrow \left( g^{(i)}, pk^{x_{c_j}}<em>j \cdot \text{index}</em>{x_i, y_i} \right)</td>
<td>pid = \frac{pk^{g^{(i)}}<em>{\text{index}</em>{x_i, y_i}}}{g^{(i)}},</td>
</tr>
<tr>
<td>c_p \leftarrow \left( g^{(i)}, pk^{x_{c_p}}_j \cdot \text{pid} \right)</td>
<td>t = \frac{pk^{g^{(i)}}<em>{\text{index}</em>{x_i, y_i}}}{g^{(i)}},</td>
</tr>
<tr>
<td>t \leftarrow \frac{1}{t}</td>
<td>Find item (pid, ypid, t') in userlist;</td>
</tr>
<tr>
<td>c_T \leftarrow \left( g^{(i)}, pk^{x_{c_T}}_j \cdot t \right)</td>
<td>If not found or t' ≤ t, abort service;</td>
</tr>
<tr>
<td>Send to cloud (c1, cp, ct);</td>
<td>k'' = h^pid</td>
</tr>
</tbody>
</table>
| | Find all (index(i,j), C(i,j)) that index(i,j) ∈ target, for each:
| | | |
| Receive tuple pairs \{(c_{k1}, c_{k2}, ..., c_{k_r}, (x_{c_{k1}}, x_{c_{k2}}, ..., x_{c_{k_r}}))\}, |
| for pair each of them: | | |
| do triple decryption |
| k'' = pk^{k_{skd}} |
| SE.Dec(k'', e^{k_1}) → e_{j1} (1 ≤ θ ≤ v) | SE.Enc(k'', e^{k_2}) → e_{j2} (1 ≤ θ ≤ v, j_0 = 0) |
| SE.Dec(k'', e^{k_3}) → e_{j3} (1 ≤ θ ≤ v) | SE.Enc(k'', e_{j1}) → e_{j1} (0 ≤ θ ≤ v, j_0 = 0) |
| ElGamal.Dec(sk_{j1}, e_{j2}) → c_{j2} (1 ≤ θ ≤ v) | Send to user (e^{k_1}, e^{k_2}, e_{j3}) |
| SE.Dec(e^{k_1}, e_{j2}) → m_{j2} (1 ≤ θ ≤ v) | Send to user (e^{k_1}, e^{k_2}, e_{j3}) |
| | Send to user (e^{k_1}, e^{k_2}, e_{j3}) |
Table 3  Notations for performance analysis

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\kappa$</td>
<td>Security parameter for the scheme</td>
</tr>
<tr>
<td>$l$</td>
<td>Number of items in dataset for each class in each block</td>
</tr>
<tr>
<td>$R$</td>
<td>Number of classes</td>
</tr>
<tr>
<td>$r$</td>
<td>Maximal number of queried classes in one query</td>
</tr>
<tr>
<td>$t$</td>
<td>Maximal number of blocks in one query after redundant blocks appended</td>
</tr>
<tr>
<td>$N$</td>
<td>Total number of blocks for the entire map</td>
</tr>
<tr>
<td>$n'$</td>
<td>Number of attribute values in Shao et al.'s system (2015)</td>
</tr>
<tr>
<td>$M$</td>
<td>Maximal number of users registered and not yet revoked in one class</td>
</tr>
<tr>
<td>$m$</td>
<td>Number of users to be revoked</td>
</tr>
<tr>
<td>$p$</td>
<td>Order of group $G$, which is prime number with length $\kappa$</td>
</tr>
<tr>
<td>$T_{Ge}^X$</td>
<td>Timing of one multiplication in group $X$</td>
</tr>
<tr>
<td>$T_{Gd}^X$</td>
<td>Timing of one division (i.e., multiplication with one inversion) in group $X$</td>
</tr>
<tr>
<td>$T_e$</td>
<td>Timing of one exponentiation in group $X$</td>
</tr>
<tr>
<td>$T_p$</td>
<td>Timing of one bilinear pairing in Shao et al.'s scheme</td>
</tr>
</tbody>
</table>

5.2  Performance analysis

In the following parts of this section, we will show computational complexity of Construction II, and compare it with that of Shao et al.'s newest data-sharing model (Shao et al., 2015, INFOCOM 2015) related to this construction. Since most of the time is cost on multiplication, exponentiation, division and pairing over group $G$ or $G_T$ (for Shao et al.'s scheme), we ignore all computational cost other than them. Also, we consider all calculations over $G_T$ and $\mathbb{Z}_q$ of Shao et al.'s work the same as that of $G$ for simplicity. Compared with public-key encryption, symmetric encryption and PRF evaluation have only negligible computation cost, so we will not count them into computing cost. In the analysis for Shao et al.'s scheme, we would consider the number of data files in one block as $R$. Our analysis focuses on main computation, including multiplication, division and exponentiation over $G$, $G_T$ and pairing. Performance analysis follows Shao et al.'s scheme and the notations are listed in Table 3.

One reason for Shao et al.'s scheme unsatisfactory efficiency is that, in order to fit in the scenario of attribution-based encryption (ABE), heavy computation has to be implemented for ABE encryption and decryption. More computational effort is made in order to blur out attribute sets, since Shao et al.'s scheme makes use of anonymous ABE so as to achieve $k$-anonymity for the contributes.

5.2.1  User

In Shao et al.'s construction, two formats of data record is provided; we take the average cost of two formats to simplify out analysis.

We can see from Table 4 that our construction has the same computational complexity as the Shao et al.'s scheme.

Table 4  Comparison on computing cost of LBS on user side

<table>
<thead>
<tr>
<th>Phase</th>
<th>Our construction</th>
<th>Shao et al.'s construction</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDC</td>
<td>$T_e + T_m^G$</td>
<td>$n'T_m^G$</td>
</tr>
<tr>
<td>UG</td>
<td>$T_e$</td>
<td>$n'T_m$</td>
</tr>
</tbody>
</table>

5.2.2  LBS provider

Comparing the computing cost on the LBS provider side for service data creation and user grant with that of Shao et al.'s scheme, we can see our construction has a smaller computational cost on provider side. As shown in Table 5, Shao et al.'s ABE-based scheme required $n'$ multiplications for data creation of one item and $n'$ multiplications for one user grant. However, our scheme requires only one exponentiation and one multiplication for each item's data creation and one exponentiation for each user grant.

Table 5  Comparison on computing cost on user side

6  Conclusions

In this paper, we propose a privacy-preserving location-based service scheme with flexible access. Our construction achieves the privacy of dataset of the LBS provider, maintains the anonymity and privacy of users. Compared with the recent LBS scheme named FINE, our construction is more efficient.
Acknowledgements

This work has been supported by the National Natural Science Foundation of China (No. 61672346, 61672347, and 61373153).

References


